

SQLodin Working Group

Request for Discussion and Implementation Record

SOD 0003

Formal Specification & Theory

Mathematical Foundations and Safety Proofs for Multi-Master Consensus

STATE

COMMITTED

CREATED

2026-09-22

LAST UPDATED

2026-09-25

INTENDED STATUS

Committed

AUTHORS

Vikrant Rathore, with assistance from Ronak Rathore

DISCUSSION FORUM

Conditional safety arguments, explicit invariants and performance boundaries

TOPICS & LABELS

consensus proofs mathematics safety liveness invariants

Status of This Record

This document is a **SQLodin Discussion (SOD)** record authored in Typst and tracked in git. It follows an RFC-style specification structure so that architectural scope, rationale, trade-offs, proof obligations, and operational boundaries remain explicit. Documents using the placeholder number XXXXX are provisional drafts. Numbered SOD documents are permanent engineering records for discussion on improvement, architecture, and enhancements of SQLodin.

Contents

1	Abstract	2
2	Status and Implementation Boundary	2
3	Introduction	2
4	Terminology and Scope	2
5	Problem Statement	2
6	Goals and Non-Goals	2
6.1	Goals	2
6.2	Non-Goals	2
7	Design Overview	3
8	Detailed Design	3
8.1	Agreement and owner ballots	3
8.2	Durable history and application	4
8.3	Fresh reads and optimistic transactions	4
8.4	Recovery and retirement	4
8.5	Identifier and retry bounds	5
9	Security & Correctness Considerations	5
10	Operational Considerations	5
11	Validation and Acceptance Gates	5
12	Alternatives Considered	5
13	Open Questions	5

14 Discussion and Revision Notes	6
15 References	6

1 Abstract

This record states the mathematical obligations behind SQLodin's fixed-voter design. Agreement, durable application, read order and recovery are separate claims. Their composition depends on explicit host assumptions. Bounded model checking and inductive proofs support selected claims; neither is a machine-checked proof of the entire executable.

SOD stands for **SQLODIN Discussions** — versioned engineering records for discussion on improvement, architecture, and mathematical foundations of SQLodin.

2 Status and Implementation Boundary

Committed; reviewed 25 September 2026. The verification suite contains 59 bounded model configurations, including required negative controls, and 48 discharged inductive obligations. The release record binds the evidence. The models and proofs cover their declared abstractions; they do not verify the compiler, SQLite, TLS, filesystem or hardware. This record remains editable under the Committed lifecycle and does not claim a completed implementation-refinement proof.

3 Introduction

Models explore adverse orderings; induction establishes invariants over arbitrary abstract steps. Both depend on choosing the right state and connecting it to code. SQLodin therefore combines explicit invariants, executable models, negative controls and implementation fault tests.

4 Terminology and Scope

Let N be the fixed number of voters and $q = \lfloor \frac{N}{2} \rfloor + 1$ the majority size. A ballot is totally ordered. Each slot has at most one value per ballot. A chosen value has durable votes from a majority. An applied prefix contains no gaps. "Safety" excludes a bad result in every allowed execution; "liveness" requires progress under stated scheduling and availability assumptions.

5 Problem Statement

A proof of quorum intersection alone does not prove a database correct. The host can release a vote before sync, forget accepted values, apply nondeterministic SQL, acknowledge an uncommitted outcome or delete recovery evidence too early. The proof boundary must name each such obligation.

6 Goals and Non-Goals

6.1 Goals

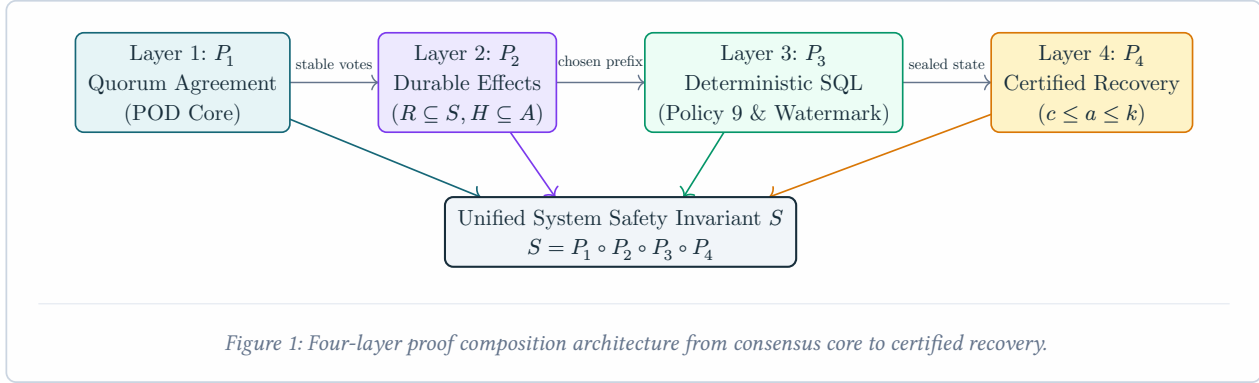
- Establish agreement, durable acknowledgement, deterministic prefix application, fresh reads, retry fencing and safe recovery publication.
- State the additional assumptions required for system liveness and progress.
- Keep every abstract action traceable to a host contract and targeted regression tests.

6.2 Non-Goals

- No claim of Byzantine safety, asynchronous wait-free progress, arbitrary SQL determinism, physical device correctness or a mechanized refinement from Odin instructions to TLA+.

7 Design Overview

The argument composes four distinct layers: quorum agreement fixes values; durable effects preserve the voting evidence; deterministic application fixes outcomes; certified recovery preserves a prefix and its suffix. A read or retry claim then refers to that same history. Performance is a separate measurement, not a consequence of agreement.



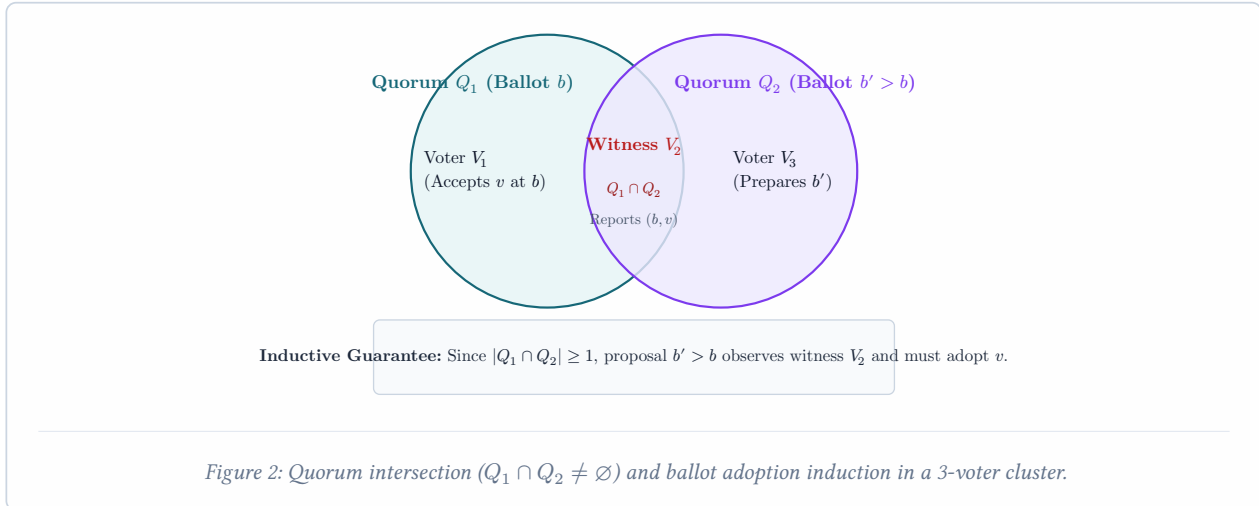
8 Detailed Design

8.1 Agreement and owner ballots

For majority quorums Q_1 and Q_2 ,

$$|Q_1 \cap Q_2| \geq |Q_1| + |Q_2| - N \geq 2q - N > 0.$$

Suppose value v is chosen at ballot b . A higher ballot prepares through a quorum that intersects the choosing quorum. Its witnesses report their highest accepted votes. The induction on higher ballots requires those votes to preserve any previously chosen value. Adopting the highest reported vote preserves that property; durable promises exclude subsequent lower-ballot acceptance by the preparing quorum. Within one ballot, value stability prevents equivocation. Intersection, adoption and persistent promises are all needed.



INVARIANT – Quorum Intersection & Value Stability

For any two majority quorums Q_1 and Q_2 over N voters, $|Q_1 \cap Q_2| \geq 1$. If a value v is chosen at ballot b , any subsequent proposal at ballot $b' > b$ that prepares through quorum Q_2 must encounter at least one voter that accepted (b, v) and must adopt value v .

The initial owner ballot can omit prepare only because it is reserved for that slot's owner, there is no valid earlier competing proposal, and restart cannot reuse the ballot for a different value. A higher promise can reject it. Recovery then uses ordinary higher-ballot preparation.

8.2 Durable history and application

Let S be stable protocol records, R released dependent effects, K chosen slots, A applied slots and H acknowledged slots. The durable-history abstraction requires:

$$R \subseteq S, \quad H \subseteq A \subseteq K.$$

Every chosen slot also has a durable voting quorum. Application commits user changes, the request outcome and the applied watermark atomically. A crash before that boundary cannot justify acknowledgement. A replay after it must not execute the effect twice.

INVARIANT – Durable History Frontiers

$$R \subseteq S \quad \text{and} \quad H \subseteq A \subseteq K$$

- $R \subseteq S$: No packet or effect is released to peers or clients before its prerequisite transitions are flushed to stable storage.
- $H \subseteq A \subseteq K$: A request is acknowledged (H) only after it is applied locally (A), which in turn requires it to be chosen by a quorum (K).

For database state D_0 , ordered values $v_1, \dots, v_k$ and admitted deterministic transition T ,

$$D_k = T(T(\dots T(D_0, v_1), \dots), v_k).$$

Equal starting state and equal values give equal logical state by induction on k . This claim requires a compatible schema and SQLite build, the SQL policy, and matching parameter bytes. Consensus does not supply those premises.

Grouped application has the additional obligation $G_i = R_i$: the observable state and outcome after request i match the individual-commit reference. Deferred constraints are checked at each request boundary. A whole-transaction rollback uses the reference fallback before acknowledgement.

8.3 Fresh reads and optimistic transactions

A fresh read joins its cohort before marker allocation. After the marker applies, the local prefix includes writes that must precede the read. An old marker cannot justify a new invocation. An optimistic transaction validates the database revision at its ordered commit. An unchanged revision preserves the preview's reads, including predicates; a changed revision rejects the commit. This is conservative and can reject nonconflicting work.

8.4 Recovery and retirement

Let c be the certified cut, a the applied cursor, k the chosen frontier and h the highest acknowledged position. Prefix recovery maintains:

$$c \leq a \leq k, \quad h \leq k, \quad (c, k] \subseteq \text{retained tail}.$$

The abstraction resets the cursor to the certified cut on crash and replays the suffix. Readiness requires the recovered prefix. A real host must also validate the certificate, seal and generation.

INVARIANT – Certified Image Prefix Recovery Boundary

$$c \leq a \leq k \quad \text{with} \quad (c, k] \subseteq \text{retained tail}$$

On restart, the storage engine mounts the certified image at cut c , verifies its seal, and replays exactly the uncompactd suffix $(c, k]$ up to the chosen watermark k . No uncommitted transaction is applied, and no chosen transaction is omitted.

Publication first makes a private generation durable, then changes the durable catalog pointer. Retirement protects the active generation and its exact predecessor. Filesystem deletion becomes durable

before ownership inventory is forgotten. These ordering rules make interrupted work recoverable without treating unrelated directories as disposable storage.

8.5 Identifier and retry bounds

For bounded fields, $f(t, i, s) = t \cdot 2^{22} + i \cdot 2^{12} + s$ is injective: division and remainders recover each component. This does not prevent reusing the same tuple. Unique voter identities, validated ranges and durable reservation frontiers supply the missing condition. Retry epochs similarly require a durable fence before old outcome entries can be discarded.

9 Security & Correctness Considerations

Assume non-Byzantine authenticated voters, fixed identities and membership, intact checked messages, compatible deterministic execution, and storage that honors successful sync. Loss, duplication, reorder, process crashes and partitions are allowed. An unknown storage or execution error fails closed. Checksums detect accidental corruption; they do not prove a voter honest.

10 Operational Considerations

Safety does not need a responsive majority. Progress does: a majority must eventually communicate, storage operations must finish, service must be fair, and some recovery exchange must eventually avoid continual preemption. Bounded demand prevents unbounded idle-slot chasing. It does not prove a wall-clock deadline. One failed voter in a three-voter configuration leaves a majority; recovery must preserve the prefix while that majority continues service.

11 Validation and Acceptance Gates

`specs/multimaster-refinement.typ` maps protocol and host actions to the implementation. TLC checks bounded state spaces and required counterexamples when protections are removed. TLAPS discharges 12 durable-history and 36 prefix-recovery obligations. The wider model suite covers ownership, read order, retries, grouping, generation publication, retirement and bounded scheduling.

Targeted implementation checks kill processes at durability boundaries, inject storage failures, exercise one-voter loss, and compare grouped execution with the reference. Passing models are necessary evidence for their claims, not a substitute for those checks. A changed protocol or host contract must revisit the affected model, proof premises and corresponding regression. No elapsed soak duration or arbitrary database-size campaign is an additional gate.

12 Alternatives Considered

Validation Approach	Scope	Primary Disqualification
Soak-Only Stress Testing	Runtime observation	Cannot systematically exercise rare reordering and split-brain states
Single Protocol Proof	Paxos core	Ignores filesystem sync, SQLite execution, and crash recovery boundaries
End-to-End Mechanized Refinement	Full executable stack	Tractable for verified microkernels, intractable for SQLite C runtime

13 Open Questions

Mechanized refinement of the complete host and stronger liveness proofs remain possible future work. They are disclosed limits of the current evidence, not claims already established or newly introduced release blockers. Membership changes require their own quorum-transition argument.

14 Discussion and Revision Notes

DECISION — 22 September 2026: Mathematical Seam Definitions

The original record contained conditional Paxos and identifier proof sketches. The review corrected the assumption that quorum intersection alone proves agreement and the assumption that an injective ID encoding prevents tuple reuse after restart.

DECISION — 24 September 2026: Verification-First Qualification

A failed cluster campaign motivated verification-first qualification. A timed-out read and a lagging returning voter were distinguished from loss of majority service. Mandatory 8-hour, 24-hour and seven-day durations were removed as approval criteria. Counterexamples became targeted regressions.

DECISION — 25 September 2026: Scoped Release Decision

Bounded ownership, durable-history, prefix-recovery and host-seam evidence now support the scoped release decision. Historical failures and tool output remain in the evidence reports.

DECISION — 25 September 2026: SOD 0005 Additions (in discussion)

SOD 0005 adds OwnedSkip (seven configurations, three negative controls), JournalCache (three, two negative) and QuorumRead (three, two negative) models. It also adds OwnedSkipProof, 41 TLAPS obligations proving that an owner's round-zero no-op fixes the slot for unbounded ballots and values. The counts above describe the qualified release candidate; the additions cover the post-release performance mechanisms only.

15 References

- SOD 0002: Architecture; SOD 0004: Host decisions.
- specs/multimaster-refinement.typ: The code map and refinement obligations.
- DurableHistoryProof.tla and PrefixRecoveryProof.tla: Induction proofs.
- tools/check_formal.py and tools/check_proofs.py: Formal evidence reproduction.
- docs/releases/2026-09-25.typ: Implementation qualification record.