

# Qualifying Observation Integrity for Selective Authorization in PostgreSQL

Aleksandar Mihajlovic

[amihajlovic1990@gmail.com](mailto:amihajlovic1990@gmail.com)

Singidunum University <https://orcid.org/0009-0009-0623-8080>

---

## Research Article

**Keywords:** authorization, PostgreSQL, cache invalidation, transaction integrity, translation validation, reproducibility

**Posted Date:** October 29th, 2026

**DOI:** <https://doi.org/10.21203/rs.3.rs-11532906/v1>

**License:**   This work is licensed under a Creative Commons Attribution 4.0 International License.  
[Read Full License](#)

**Additional Declarations:** The authors declare potential competing interests as follows: The author works as Associate Director - Forward Deployed Engineer at Modon Holding in Abu Dhabi, United Arab Emirates. The study does not report an employer deployment or use proprietary employer data. No employer funding or endorsement is claimed.

---

# Abstract

Selective authorization reuses an earlier policy decision when the database observations supporting that decision remain unchanged. Epoch counters can support this inference only when installed mutation code advances the intended identities and the executor faithfully observes those advances. This study formulates that producer-to-consumer contract and implements a restricted PostgreSQL installation qualifier. Two motivating counterexamples preserve apparently correct helper code or marker updates while breaking the inference: reordered formal parameter names redirect invalidation, and row-level security hides correctly maintained markers. The qualifier combines restricted code checking with function binding, control-storage, role and trigger-attachment checks. Separately authenticated writer and executor sessions provide a complementary behavioral oracle. A conditional composition argument explains safe reuse; a constructive counterexample shows why finite positive visibility probes cannot establish an unrestricted contract. On authenticated PostgreSQL 16.15, 48 valid catalog controls qualified and 48 incompatible variants were refused. A frozen 108-pair challenge produced zero unsafe qualifications in five unsafe configurations, versus two for a fixed structural lint, while refusing two safe unsupported forms. The purchase protocol passed 22 checks. In 540 measured cycles on PostgreSQL 16.15 under full-system CPU emulation, selective execution reduced the 10,000-row unchanged-state median cycle time by 24.0% relative to repeated native SQL evaluation with the same epoch maintenance; relevant mutations increased costs. The result is an executable qualification method with explicit correspondence obligations, reproducible failure evidence and measured support limitations.

## 1 Introduction

An approval based on database state can become invalid before its associated action executes. A purchase may have been permitted while no hold existed and sufficient budget remained, yet conflict with a newly inserted hold or another committed purchase. Repeating the complete policy query at execution is a straightforward solution. Selective reuse instead retains evidence about the observations used by an earlier decision and checks whether those observations changed. The potential benefit depends on the cost of maintaining and checking that evidence relative to fresh evaluation.

The central problem in this paper is the integrity of that evidence at a concrete deployment boundary. A change counter is useful only if a relevant write advances the right counter, the counter retains its history, and the admitting executor can read it. Checking a generated trigger body does not establish all three properties. Nor does demonstrating that an administrator can see an increment establish what an ordinary executor sees. These distinctions matter when the database combines named function parameters, security-definer triggers, effective privileges, row-level security and statement snapshots.

OntoCommit is a research prototype that checks a deliberately restricted relational invalidation profile before admitting selective reuse. Its checker recognizes row and equality-partition invalidation obligations, compares them with installed trigger code and attachment metadata, and examines the surrounding role and storage configuration. The present study extends the qualification boundary to the

entire observation path. Its motivating defects were found in a prior prototype: the qualifier accepted a helper whose first two formal parameter names were exchanged, and accepted an epoch table whose row-security policy concealed increments from the executor. Both changes were installed before qualification. They are configuration counterexamples, not evidence that an ordinary writer acquired administrative privileges.

The research question is therefore precise: **under which checked and trusted installation conditions may equality of executor-visible markers justify reuse of an earlier policy observation?** A second question concerns evidence: which defects are detected by a conventional structural catalog check, the restricted qualifier, and behavioral tests that cross actual database roles? A final experiment connects qualification to one complete purchase protocol and measures the cost against fresh native SQL evaluation under matched transaction obligations.

The contributions are a producer-to-consumer contract and conditional composition argument; a restricted installed-code and deployment checker with executable counterexamples; a frozen policy-observation challenge with an explicit finite-probe limitation; and a connected PostgreSQL implementation that combines freshness, current authority, prospective budget bounds and atomic effects. The method builds on translation validation, incremental maintenance and authorization consistency [1, 2, 3, 4]. Its distinctive focus is the checked correspondence between installed mutation behavior and executor-visible freshness evidence.

## 2 System model and motivating failures

### 2.1 Observations and trust boundary

A database observation is the content of a selected row, an equality partition, or a whole supported relation. Absence is observable content: inserting the first matching row changes an empty partition. A descriptor encodes observation kind, table, canonically ordered field names and typed values. For state  $S$ , let  $O_d(S)$  be the content identified by descriptor  $d$ , and let  $E_d(S)$  be its retained stored epoch. The executor's actual read is  $R_u(d, S)$ , which need not equal  $E_d(S)$  under an unsuitable visibility configuration.

A retained selective approval binds the action, principal, policy configuration, handler identifier and a sufficient set of observations. Sufficiency means that equal observation contents, with equal remaining policy inputs, preserve the decision. The invalidation checker establishes mutation coverage for declared observation types; it does not discover every semantic dependency of arbitrary application code. In the connected experiment, the fixed SQL policy reads an account row and the spend and hold partitions for the proposed scope. Its dependency set is specified directly and reviewed against that SQL.

The trusted components are the PostgreSQL engine and operating system, the catalog extractor and restricted checker, identity encoding, the fixed service/evaluator, trusted principal transport and deployment administration. An ordinary writer can perform permitted domain-table DML but cannot edit

authorization metadata, suppress the required triggers, assume the owner role or change the protected schema. The executor is a trusted service role, not arbitrary client-supplied SQL. A qualified deployment is assumed stable between qualification and use. Hostile owner changes, restored or cloned histories, arbitrary extensions and remote side effects are outside the implemented guarantee.

## 2.2 Producer binding can differ while the body matches

The intended helper has parameters `bump(p_kind text, p_table text, p_fields text[], p_values jsonb)`. The defective variant exchanges the names of its two same-typed leading parameters. The positional type signature and function body remain unchanged, but references to those names now bind different actual arguments. A partition update can consequently be stored under an identity beginning with the table name where the observation kind belongs. The intended marker remains unchanged.

PostgreSQL exposes argument types, names and modes separately in `pg_proc`; matching a type signature does not establish name-to-position correspondence [5]. The repaired profile requires the exact supported binding, input modes and built-in type identities, together with the function's execution-relevant attributes. It refuses unsupported signature variants instead of attempting general semantic equivalence.

## 2.3 Correct storage can coexist with an incorrect read

The second counterexample enables row-level security on the epoch table without an executor-read policy. The owner-controlled trigger stores the correct increment, while an ordinary executor sees no row. Interpreting that result as zero conflates true absence with hidden state. PostgreSQL's documented row-security rules explain this behavior: owners normally bypass policies, whereas ordinary roles are subject to them [6].

The restricted solution is to forbid row-level security and forced row-level security on all protected control tables. It also checks relation kind, owner, persistence, columns, constraints and effective access. Granting the executor elevated bypass privileges would change the trust boundary and is unnecessary. This refusal does not mean every RLS configuration is unsafe. A permissive policy can be safe for a particular observation contract but remain outside this qualifier's supported language.

## 3 Qualifying the complete observation path

### 3.1 Restricted mutation coverage

The row invalidator advances observation identities affected by inserts, deletes and updates. An update covers both departing and arriving identities, including key or partition changes. Whole-relation descriptors provide a conservative observation option. Counters are retained even after a row or partition becomes empty, preventing a delete-and-reinsert sequence from erasing change history. Counter overflow must abort the transaction rather than wrap. Data and marker writes occur in the same transaction.

The checker separately derives the obligations of the declared schema and parses the supported invalidation program. It compares the resulting obligations with the installed program and trigger attachments. The supported schema uses non-null text, signed 64-bit integers and booleans, fixed primary keys and declared equality partitions; text identity uses the specified collation. Supplementary S1 summarizes the supported profile and refusals; the complete grammar is defined by the parser and its accompanying artifact documentation. This is a restricted recognizer, not a verifier for arbitrary PL/pgSQL.

## 3.2 Binding and storage checks

For the helper, mutation gate and row-trigger functions, qualification inspects ownership, security-definer configuration, search path, language, return and argument type identities, argument names/modes, routine kind, strictness, volatility, parallel properties, defaults, set-returning behavior and other admitted attributes. Trigger checks bind the function to the intended table, timing, event set and firing mode. They refuse predicates, arguments and column-filtered update attachments outside the declared profile.

The control contract covers epochs, global\_clock, registry, tickets, effects and ledger. The qualifier verifies permanent ordinary heap relations under the expected owner, exact admitted column and constraint shapes, no RLS or forced RLS, and restricted table/column access. The profile intentionally rejects some harmless variations. For example, an equivalent constraint expressed in a form the checker does not recognize is not certified by semantic analogy. Such refusals are a support limitation, not evidence of an unsafe installation.

Effective-role checks examine the actual session identity and reachable permissions, rather than only the nominal login flags. They inspect schema and relation privileges, role memberships, privileged callable functions, trigger suppression paths and unsupported domain features. Genuine session authentication is a separate requirement: a privileged connection followed by SET ROLE is useful for embedded diagnostics but does not satisfy the deployment gate.

## 3.3 Behavioral challenges

The complementary observation test uses one authenticated executor to capture a semantic query result and its relevant markers, an authenticated writer to commit a controlled mutation, and the executor to reread the observation and markers. A changed semantic observation with unchanged visible markers falsifies the observation contract for that trace. An administrative read may explain whether the producer wrote a wrong identity or the consumer failed to see a correct identity, but it cannot replace the executor read.

The challenge uses disposable installations and independently specified SQL outcomes. It does not commit business effects after a rejected qualification. Its policy/marker pairs are observation witnesses; they are not counted as unauthorized purchases. Nor are sequential challenge reads evidence of concurrent admission correctness. The connected workflow experiment tests the admission snapshot separately.

The comparison has three executable methods. A fixed structural lint checks relation and role properties, types, keys, privileges, RLS and trigger presence/attachments. The restricted qualifier additionally checks the admitted invalidation semantics and function binding. A finite behavioral probe exercises a selected identity. The lint is a documented comparator, not a claim about every DBA review or existing validation product. A conjunction of static qualification and probes is reported without assuming that it improves on either method in the selected sample.

## 4 Conditional guarantees and a limit of probing

### 4.1 Reuse under an observation contract

Assume a sufficient observation set; exact producer identity and mutation coverage; atomic, retained, nondecreasing epochs without wrap/reset; faithful executor reads; and a snapshot obtained after acquiring a common gate held through the protected transaction. Also assume stable qualified code and roles. Faithful reads at capture and admission mean  $R_u(d, S_i) = E_d(S_i)$  for  $i \in \{0, 1\}$ . Consequently, equality of the values actually observed by the executor gives

$$(\forall d \in D, R_u(d, S_0) = R_u(d, S_1)) \Rightarrow (\forall d \in D, O_d(S_0) = O_d(S_1)).$$

To see why, suppose some recorded observation differs. At least one committed mutation between capture and admission changed it. Coverage advances the marker under the exact descriptor; retention prevents return to the old value. A faithful fresh read must therefore reveal a different marker, contradicting equality. Changes that later restore the original content may cause conservative revalidation, but do not invalidate the implication.

The gate excludes an intervening admitted writer after the freshness check. Sufficiency then transfers the earlier policy result. This reasoning concerns evidence freshness. Current principal/action binding, policy and handler identity, revocation, expiry, one-use consumption and prospective bounds remain separate obligations. The connected handler checks them and commits the local effect atomically. The argument is mathematical and conditional; neither the PostgreSQL implementation nor the checker has been proved correct in a theorem prover.

### 4.2 Indistinguishable reads and finite probes

A guard receiving identical complete inputs in two histories cannot distinguish them merely by computing a different function of those inputs. If one history permits the action and the other forbids it, a sound guard must obtain distinguishing evidence, perform sufficient fresh evaluation, or conservatively refuse. Wrong identity binding, hidden markers and stale snapshots can each create that indistinguishability when no other guard input records the change. This elementary observation explains the failure mechanism; it is not offered as a new general theorem.

Finite positive probes have a corresponding limitation. Let the probed identity set be  $Q$ , and choose an allowed identity  $d^*$  outside  $Q$ . A visibility rule can reveal all identities in  $Q$  while hiding  $d^*$ . The probe

transcript then matches a fully visible installation, although a mutation of  $d^*$  changes its stored marker without changing the consumer's apparent zero. The construction assumes an unrestricted rule language capable of distinguishing those identities. It does not defeat a static profile that forbids RLS, exhaustive checking of a genuinely finite closed domain, or a proof of the visibility predicate. Random probes can aid discovery without establishing universal completeness.

The supplied abstract model enumerates initial states, write sequences of length at most three and observed descriptors. It records both unsound reuse of changed observations and permitted-to-forbidden toy-policy cases. Each profile has the same denominator of 680 combinations. The complete abstract contract yields zero unsound reuses; removing premises produces witnesses. These bounded results corroborate the examples and are not engine failure probabilities or a mechanized proof of the unbounded argument.

## 5 A connected PostgreSQL purchase protocol

The fixed workflow has an account keyed by tenant, entity and period; a spend relation partitioned by that scope; and a hold relation with the same partition. A valid positive purchase amount  $a$  must satisfy currency and amount restrictions, absence of an active hold, nonnegative valid authoritative state, sufficient remaining balance, and the prospective cap

$$\text{spent}(S) + a \leq \text{cap}(S), \quad a \leq \text{remaining}(S).$$

The proposal is included in the cap check. For example, 140 existing spend plus an 80 purchase is refused under a cap of 150. Aggregation uses PostgreSQL numeric for SUM(bigint) and transports integers exactly; it does not use binary floating point for financial amounts.

Preparation acquires the common gate, reads the current registry, evaluates the fixed SQL policy, and stores an action-bound approval with expiry. Selective mode also captures three descriptors and the decision snapshot. Native mode retains no unused epoch evidence and repeats the same native SQL policy on each new commit. Both modes use identical indexes, maintained installation, authority ledger, debit, spend record and effect implementation.

Commit acquires the transaction-scoped advisory gate in its own statement. Later statements use fresh Read Committed snapshots, including after a lock wait [7]. The service checks the ticket and trusted principal binding. A previously consumed matching authority returns its historical receipt; a different action under that authority is refused. A receipt is not a new execution and does not require a new approval after revocation. For a new effect, the registry, policy/handler bindings, revocation and expiry are checked. Unchanged epochs permit selective reuse; otherwise the policy is reevaluated. Debit, spend, local effect and authority consumption then commit together.

All protected operations share one gate. This deliberately conservative design serializes admission and relevant raw DML and limits throughput. The experiment establishes neither fine-grained lock scalability

nor a distributed protocol. Trusted transport supplies the principal; this policy does not implement a separate tenant-membership ACL. Configuration fingerprints and handler identifiers are deployment bindings maintained by trusted administration, not executable-code attestation. Expiry is tested at the post-gate guard read, not guaranteed against every possible delay until the physical commit instant. Recovery tests concern client termination and lost responses within the same database history, not restoration of an old backup.

## 6 Evaluation methodology

### 6.1 Evidence design and environment

The development protocol distinguishes known regressions, frozen challenge cases, bounded models and connected effects. The two motivating failures were already known when the repairs were designed. A separate AI agent in the same project specified twelve SQL policy families and nine installation configurations before inspecting the repaired checker. Case definitions and runner hashes were frozen before execution, with documented pre-execution corrections. This separation is an internal cross-check, not independent external or human validation. The configurations are crossed with policy families; 108 pairs are not 108 independently sampled deployments.

Experiments used unmodified PostgreSQL 16.15 in a complete Linux 6.8 guest under QEMU 8.2.2 TCG emulation, with two virtual CPUs and 1,536 MiB guest memory. PostgreSQL ran as genuine guest UID 1000 on an ext4 virtual block disk. Psycopg 3.3.6 clients used TCP with SCRAM-SHA-256. The environment avoids substituting an elevated role-switch session for an ordinary login. Absolute latency is specific to this small emulated server; it is not an enterprise hardware benchmark.

The primary server configuration enables `fsync`, `synchronous_commit` and `full_page_writes`. Ordinary writer and executor sessions authenticate separately; their session and current roles agree. The artifact records restricted role attributes and a rejected wrong-password connection. PGLite runs are explicitly labeled catalog/protocol diagnostics. Their SET ROLE behavior is never substituted for ordinary server authentication.

### 6.2 Correctness measures and comparators

The catalog study pairs every intentionally incompatible installation with a freshly qualified valid control. It reports refusals rather than treating every out-of-profile attribute as a demonstrated exploit. The challenge assigns configurations to supported valid, unsafe or safe unsupported classes independently of the checker output. It measures accepted unsafe configurations, accepted supported controls, support refusals and concrete semantic/marker traces. A policy change after intentionally bypassing a rejected qualifier is a diagnostic, not a failure of the admitted service.

The purchase study exercises intervening relevant and unrelated writes, cap conflicts, revocation and version changes, principal/action binding, expiry, replay, concurrent consumers, lock-wait freshness and

abrupt client termination. Effects, spend, remaining balance and authority consumption are compared. The native path evaluates the same SQL predicate and makes the same protected transition. Finite case counts describe the exercised population; they do not estimate deployment incident rates.

## 6.3 Cost measurement

The fixed grid uses 100, 1,000 and 10,000 initial spend rows, three conditions (no intervening write, an unrelated write and a relevant no-op update), 30 measured repetitions per mode per cell and three warmups. Mode order is randomized within paired blocks with recorded seeds. Both paths retain the same indexes and effect logic. Fixtures are loaded in batches of 250 with maintenance active, and query plans are saved. The final run therefore contains 540 measured cycles. Client monotonic wall-clock measurements separate preparation, intervening mutation and commit. The first completed 540-cycle run was preserved before an implementation/planning investigation; the final measurement is a development-informed rerun of the unchanged workload, not a new independent performance holdout.

Qualification/provisioning and evidence maintenance are distinct from per-request service cost. The primary native comparator shares row maintenance with selective mode, isolating the cost of retained-evidence checking versus policy reevaluation. It is not a minimal native deployment without epochs. The paper therefore does not claim an overall resource advantage over an optimized native-only service. Earlier statement-batching measurements are retained as historical supplemental evidence, not relabeled as authenticated service measurements.

## 6.4 Research assistance and provenance

OpenAI Codex/ChatGPT assisted with study design, implementation, controlled configuration and test generation, literature retrieval, formal-argument drafting, experiment orchestration, analysis and manuscript preparation. Additional AI agents within the same project developed selected challenge cases and cross-checks. The artifact records source versions, frozen specifications, commands, actual outputs and environment limitations. No independent human reproduction or external peer review is asserted. The author supplied the research motivation, industry context and scope and is the sole human author. Research uses of AI are disclosed here because they extend substantially beyond writing assistance.

# 7 Results

## 7.1 Catalog qualification and server admission

All six original server tests passed without skips on ordinary PostgreSQL 16.15. They exercised actual executor qualification, seven denied writer operations, raw-writer invalidation, local effect/replay, concurrent gate blocking and COPY FROM STDIN. The repaired catalog experiment accepted **48/48** fresh valid paired controls and refused **48/48** deliberately incompatible variants. The same counts were reproduced in PGLite, whose embedded session correctly failed the strict authentication gate. Not all

incompatible attribute variants were shown to cause unsafe effects; the result is conformance to the supported profile.

The two original failure mechanisms remain visible in deliberately executed post-refusal diagnostics. Exchanging helper parameter names leaves the intended executor epoch at zero while the policy changes from allowed to denied. Enabling epoch RLS likewise hides the correctly stored increment. The valid control observes 0 to 1. The corrected service does not admit either rejected installation. The additional column-filtered gate regression is included in the 48 cases.

## 7.2 Frozen observation challenge

All nine installations completed with no setup or comparator errors, and all 108 policy transitions matched the specified TRUE-to-FALSE outcome. Table 1 uses configurations as its unit. Both supported controls qualified under every method and preserved change observations across all twelve families, giving 24 supported observation pairs. The restricted qualifier accepted none of the five unsafe configurations, but also refused both safe unsupported forms. The structural lint correctly rejected hidden-epoch and decoy RLS; its two false qualifications concern argument binding and insufficient partition coverage.

Table 1  
Qualification outcomes across nine authored configurations. Unsafe acceptance and support refusal are separate outcomes.

| Method               | Unsafe accepted | Supported refused | Safe unsupported refused |
|----------------------|-----------------|-------------------|--------------------------|
| Structural lint      | 2/5             | 0/2               | 1/2                      |
| Finite probes        | 1/5             | 0/2               | 0/2                      |
| Restricted qualifier | 0/5             | 0/2               | 2/2                      |
| Qualifier and probes | 0/5             | 0/2               | 2/2                      |

The unsafe configurations yielded 59 stale-policy witnesses among 60 policy/configuration pairs. Epoch-update suppression allowed the first marker insertion in one initially empty case, so that case did not produce a stale witness. No failure was invented to make this count 60. The decoy configuration passed its three positive probe cycles while producing twelve target-identity witnesses; the structural lint and restricted qualifier both refused it. Conjoining probes with the qualifier did not improve the measured acceptance/refusal counts. Its benefit here is explanatory and diagnostic evidence, rather than additional measured classifier accuracy.

## 7.3 Connected effects and recovery

All **22/22** authenticated purchase checks passed. Selective and native modes produced the expected atomic effect and replay behavior and both refused intervening spend of 140 followed by a proposed 80 under a cap of 150. Principal/action changes, revocation, policy and handler changes, positive-TTL

expiry, suspension and intervening holds were handled as specified. An unrelated writer mutation preserved reusable evidence.

The concurrency case used separate authenticated sessions for conflicting purchases; the final state and authority ledger matched the prospective bound. A separate case observed an executor waiting for the gate before allowing the writer to commit, then checked the executor’s fresh denial. Three injected pre-commit exceptions and three abrupt pre-commit client exits left no partial debit, spend, effect or consumption. A client exit after commit recovered by replaying the same receipt once. These are client interruption and transaction/replay observations, not server-power-loss or backup-restoration experiments.

## 7.4 Complete costs and lookup sensitivity

Table 2 reports the complete prepare–mutation–commit cycle. The mutation column describes controlled intervening writer work and is included in the cycle time; it is not server-only request latency. Per-stage samples, prepare-plus-commit values derivable from those samples, empirical tails and paired exploratory intervals are retained in the artifact. Thirty samples per cell do not establish a high-percentile service guarantee.

Table 2  
Median cycle time in milliseconds on the emulated PostgreSQL server. A ratio below one favors selective mode within this matched-maintenance experiment.

| Initial rows | Mutation  | Selective ms | Native ms | S / N |
|--------------|-----------|--------------|-----------|-------|
| 100          | none      | 29.66        | 28.53     | 1.039 |
| 100          | unrelated | 37.32        | 36.58     | 1.020 |
| 100          | relevant  | 38.85        | 35.92     | 1.082 |
| 1,000        | none      | 31.30        | 33.14     | 0.945 |
| 1,000        | unrelated | 37.37        | 39.59     | 0.944 |
| 1,000        | relevant  | 43.64        | 40.76     | 1.071 |
| 10,000       | none      | 66.29        | 87.27     | 0.760 |
| 10,000       | unrelated | 74.91        | 110.83    | 0.676 |
| 10,000       | relevant  | 99.11        | 96.44     | 1.028 |

Across unchanged and unrelated-mutation conditions, selective mode performed 0 commit-time policy evaluations versus 180 for native mode. Relevant writes forced reevaluation. Avoided query counts are not themselves a latency result. At 10,000 initial rows without an intervening mutation, the measured cycle medians were 66.29 ms and 87.27 ms respectively.

The first 540-cycle campaign used a JSON-array expansion joined to the epoch table. At 10,000 rows without an intervening mutation, its selective and native cycle medians were 197.18 and 100.44 ms. That adverse result is preserved with the exact initial source. Query-plan investigations found a hash-join/sequence-scan plan at a small metadata size, but a grown-fixture diagnostic instead used indexed lookups and did not reproduce the original large slowdown. The cause of the first campaign's size-dependent excess cost therefore remains unresolved.

The revised service exposes its three fixed descriptor keys directly in a parameterized IN predicate. Only the lookup changed; native policy semantics, effects, maintained installation and the declared timing grid were retained. The revised code was frozen and all 23 purchase unit checks, 22 server checks and 540 timed cycles were rerun. These measurements are development-informed, not an independent performance holdout, and are not pooled with the initial campaign.

In the final same-run comparison at 10,000 rows, selective cycle latency was 24.0% lower for unchanged state and 32.4% lower after an unrelated write. Relevant mutations instead made selective mode slower: 99.11 versus 96.44 ms at 10,000 rows. At 100 rows with no mutation it was also slightly slower, 29.66 versus 28.53 ms. These results identify a workload-dependent regime for this implementation, without attributing all before/after differences to a proven query-plan mechanism.

A separate component experiment used ten fresh connections against a warm server. Median connection establishment was 28.91 ms and direct qualification was 215.60 ms. A service constructor measured after that qualifier in the same session took 124.05 ms; this order-biased warm measurement is not an independent cold-start cost. Eighty indexed writer operations covered 1-row and 100-row inserts, no-op updates, scope moves and deletes with all row maintenance active. For 100-row operations, the median total DML-plus-maintenance-plus-commit costs were insert 235.16 ms, noop update 461.14 ms, partition move 463.89 ms, delete 211.61 ms. These are total write costs, not epoch overhead isolated against a no-maintenance writer. Provisioning, physical cold-cache behavior and production capacity were not benchmarked.

## 8 Related research

Matoi studies selective consequence-evidence revalidation, authority separation and incomplete mediation inventories [1]. The present study focuses on a narrower installed PostgreSQL boundary: formal parameter binding, retained storage and the actual executor's view of markers. The comparison is conceptual; Matoi's implementation was not reproduced here. Zanzibar addresses consistent authorization at global scale [8]. Its deployment evidence and consistency architecture are substantially broader than this single-instance experiment, and no performance comparison is made.

Translation validation provides the established architecture of checking individual generated results [2]. Certified graph maintenance and DBSP provide formal incremental-computation foundations [9, 3], while earlier work addresses trigger verification and state-dependent access control [10, 4]. OntoCommit

checks a restricted invalidation program and its operational prerequisites. It does not offer a more expressive general incremental query language or a stronger mechanized proof.

Proof-carrying authorization and revocable or use-once certificates already address reusable evidence and authority lifecycle [11, 12]. Cedar validation establishes properties about policy evaluation under schema-conforming inputs [13]; recent AutoCedar and SymCert work further develops verifier-guided synthesis and verified policy analysis [14, 15]. Those goals differ from checking whether the installed database observation machinery satisfies the premises of a freshness test. The policy evaluator and correspondence between declared observations and policy meaning remain trusted in this implementation.

Database testing methods such as pivoted query synthesis and query partitioning construct executable oracles [16, 17]. Metamorphic security testing and role-aware RLS testing also predate this study [18, 19]. The contribution of the cross-role challenge is its specific relation between a committed semantic change and a consumer-visible marker, together with a negative control exposing the limitation of finite positive tests. Simply using multiple database roles or mutation tests is not claimed as novel.

## 9 Discussion and limitations

The results support a deployment-specific observation contract. The important distinction is between a counter existing in storage and a reliable statement about what its reader observed. Producer semantics, storage retention and consumer visibility must compose. The motivating defects show why a theorem with correct premises can coexist with an implementation that fails to establish those premises. Explicit correspondence checks and executable traces make that gap assessable.

The qualifier's restricted language is both a strength and a limitation. Refusing RLS and unsupported function forms makes the accepted profile easier to inspect, but can reject safe installations. A broad enterprise deployment may rely on partitioned tables, configurable RLS, extensions or deployment practices outside this profile. The reported acceptance and refusal counts apply to authored cases, not a sampled enterprise population. The twelve policy families share a conservative observation partition; they are not twelve separately verified business handlers.

The formal argument assumes coverage, retention, faithful reads, sufficient dependencies and stable administration. The implementation has a substantial trusted computing base. Catalog queries, parser behavior, canonical encoding and transaction adapters are tested rather than machine verified. A newly discovered admitted configuration with a changed observation and unchanged visible marker would directly falsify the implementation claim. A finite successful test campaign cannot eliminate that possibility.

The connected workflow is one local purchase policy. It does not verify business intent, establish a general identity provider or tenant ACL, make arbitrary writer edits financially valid, or make remote effects atomic. It preserves the stated prospective restrictions for admitted purchases from the

specified authoritative state. Backup restoration, cloned histories and administrative rollbacks require a separate lifecycle design; no database-incarnation field is implemented. Client crash/reconnect evidence does not establish behavior under server power loss, storage failure or cross-region failover.

Performance depends on observation size, write activity, evidence construction, database indexes and deployment costs. An unchanged certificate can avoid a policy query without reducing total latency. Relevant mutations force reevaluation, and global gating constrains concurrency. Emulated-server measurements offer a controlled functional environment and within-environment comparison, not production capacity estimates. The matched native baseline does not charge selective mode the incremental cost of epochs relative to a native-only deployment. These boundaries are retained even if a particular measured case favors selective reuse.

External validation remains valuable. Same-project agents, frozen cases and source hashes improve traceability but do not constitute independent human replication. A useful next evaluation would test a broader schema/configuration corpus, a native-only deployment with equivalent business protection, other PostgreSQL versions and a less centralized admission mechanism. These are extensions, not unperformed results embedded in the current claim.

## **10 Conclusion**

Selective authorization requires an observation contract that reaches from installed mutation semantics to the admitting executor's actual read. OntoCommit makes that contract explicit for a restricted PostgreSQL profile and connects it to an atomic prospective-purchase workflow. In the frozen challenge, the qualifier refused all five unsafe configurations while retaining both supported controls and refusing two safe unsupported forms. Cross-role probes exposed concrete failures, but the decoy showed why positive probes alone cannot certify unrestricted visibility. The resulting contract and artifact make both the safety premises and their implementation limits directly assessable.

## **Declarations**

## **Competing interests.**

The author works as Associate Director - Forward Deployed Engineer at Modon Holding in Abu Dhabi, United Arab Emirates. The study does not report an employer deployment or use proprietary employer data. No employer funding or endorsement is claimed.

## **Ethics approval and consent to participate.**

Not applicable. The experiments use synthetic application data in disposable local research installations and do not involve human participants or private enterprise records.

## Consent for publication.

Not applicable. The study includes no identifiable participant information.

## Funding.

This research received no external funding. The author funded the study personally.

## Author contributions.

Aleksandar Mihajlovic is the sole human author and supplied the research motivation, industry context and scope. The research was undertaken in connection with doctoral studies at Singidunum University. Substantive AI assistance with research design, implementation, experiment orchestration, analysis and writing is described in Section 6.4.

## Availability of data and materials.

Supplementary S1 provides the qualification contract, proofs, controlled challenge definitions and reproduction boundaries. Supplementary S2, supplied as `OntoCommit_ResearchSquare_S2_Reproducibility.zip`, contains the implementation source, synthetic fixtures, frozen specifications, raw results, reproduction scripts and checksums. Both completed timing campaigns are retained separately. The artifact README identifies the results and reproduction commands. Third-party components retain their stated licenses.

## References

1. Zexun W (2026) *Matoi: Selective Revalidation of Consequence Evidence for Agent Authorization*. arXiv:2610.21394v1. <https://arxiv.org/html/2610.21394v1>
2. Pnueli A, Siegel M, Singerman E (1998) *Translation Validation*. TACAS 1998, LNCS 1384, 151–166. <https://doi.org/10.1007/BFb0054170>
3. Mihai Budiu T, Chajed F, McSherry L, Ryzhyk VT (2023) DBSP: Automatic Incremental View Maintenance for Rich Query Languages. PVLDB 16(7):1601–1614. <https://www.vldb.org/pvldb/vol16/p1601-budiu.pdf>
4. Lars E, Olson, Carl A, Gunter P, Madhusudan (2008) *A Formal Framework for Reflective Database Access Control Policies*. CCS 2008. <https://madhu.cs.illinois.edu/ccs08.pdf>
5. PostgreSQL Global Development Group (2026) *Catalog pg\_proc*. PostgreSQL 16 documentation. <https://www.postgresql.org/docs/16/catalog-pg-proc.html>
6. PostgreSQL Global Development Group (2026) *Row Security Policies*. PostgreSQL 16 documentation. <https://www.postgresql.org/docs/16/ddl-rowsecurity.html>

7. PostgreSQL Global Development Group (2026) *Transaction Isolation*. PostgreSQL 16 documentation. <https://www.postgresql.org/docs/16/transaction-iso.html>
8. Ruoming Pang Ramón, Cáceres M, Burrows Z, Chen P, Dave N, Germer A, Golynski K, Graney N, Kang L, Kissner JL, Korn A, Parmar CD, Richards MW (2019) *Zanzibar: Google’s Consistent, Global Authorization System*. USENIX Annual Technical Conference, 675–688. <https://www.usenix.org/conference/atc19/presentation/pang>
9. Angela Bonifati S, Dumbrava, Emilio Jesus Gallego Arias (2018). *Certified Graph View Maintenance with Regular Datalog*. arXiv:1804.10565. <https://arxiv.org/abs/1804.10565>
10. Hong Anh Le, Ninh Thuan Truong (2013) *Modeling and Verifying DML Triggers Using Event-B*. ACIIDS 2013, LNCS 7803, 539–548. [https://doi.org/10.1007/978-3-642-36543-0\\_55](https://doi.org/10.1007/978-3-642-36543-0_55)
11. Chaudhuri A (2009) Deepak Garg. *PCAL: Language Support for Proof-Carrying Authorization Systems*. CMU-CS-09-141. <https://people.mpi-sws.org/~dg/papers/cmu-cs-09-141.pdf>
12. Jamie Morgenstern D, Garg FP (2012) *A Proof-Carrying File System with Revocable and Use-Once Certificates*. STM 2011, LNCS 7170, 40–55. [https://doi.org/10.1007/978-3-642-29963-6\\_5](https://doi.org/10.1007/978-3-642-29963-6_5)
13. Cedar Project (2026) *Policy validation*. Cedar Policy Language Reference Guide. <https://docs.cedarpolicy.com/policies/validation.html>
14. Adarsh Vatsa S, Shome Y, Zhou WE (2026) *AutoCedar: An Agentic Framework for Verifier-Guided Access Control Policy Synthesis*. arXiv:2607.03656v1. <https://arxiv.org/abs/2607.03656>
15. Emina Torlak (2026) *SymCert: Verifying SMT-Based Policy Analyses*. FMCAD 2026, pp. 238–247; TU Wien Academic Press. <https://repositum.tuwien.at/handle/20.500.12708/230492>
16. Manuel Rigger ZS (2020) *Testing Database Engines via Pivoted Query Synthesis*. OSDI 2020. <https://www.usenix.org/system/files/osdi20-rigger.pdf>
17. Manuel Rigger ZS (2020) *Finding Bugs in Database Systems via Query Partitioning*. Proceedings of the ACM on Programming Languages, OOPSLA 2020. <https://www.manuelrigger.at/preprints/TLP.pdf>
18. Chaleshtari NB, Pastore F, Goknil A, Briand LC (2023) *Metamorphic Testing for Web System Security*. arXiv:2208.09505v2. <https://arxiv.org/abs/2208.09505>
19. Ariga Technologies (2025) *Testing PostgreSQL Row-Level Security (RLS) Policies with Atlas*. Official implementation documentation, 30 March 2025. <https://atlasgo.io/faq/testing-rls>

## Figures

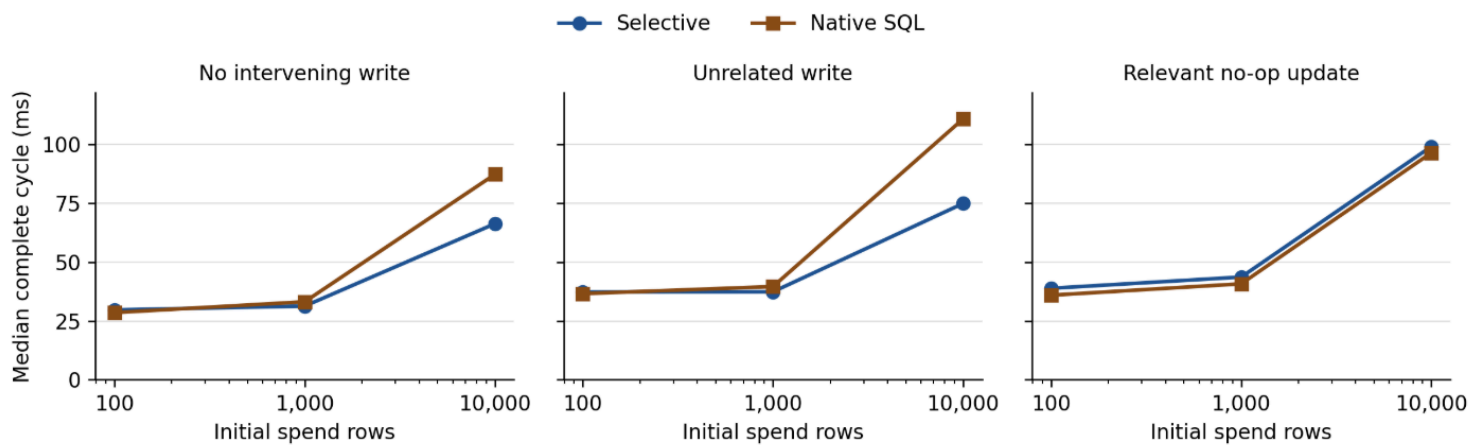


**Figure 1**

Same fixed workload and matched row-maintained installation. Each point is the median of 30 measured cycles after three warmups. The horizontal axis is logarithmic; the server uses full-system CPU emulation.

## Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [OntoCommitResearchSquareS1Methods.pdf](#)
- [OntoCommitResearchSquareS2Reproducibility.zip](#)