

R1.2 grouped and serial application

The reference transition is one complete ordered request in a FULL SQLite transaction: request fence lookup, SQL execution or classified rejection, outcome and session metadata, revision and applied watermark. The application group optimization applies at most sixteen adjacent requests under one outer transaction. It does not change their log order or SQL contract.

Assume a deterministic SQL transition under R1.1. At each request boundary, the private grouped state equals the reference state after the same prefix. Fence lookup observes preceding requests in the group, so duplicate identities, gaps, expired epochs and read-version conflicts have the same result. A savepoint isolates each request. A classified error rolls back its data effects before its rejection outcome is recorded. The deferred-foreign-key counter must be clear at each successful boundary, preventing a later request from repairing an earlier request that the reference would have rejected. Thus successful staging preserves the prefix relation by induction on the finite group length.

SQL ROLLBACK conflict actions and rollback triggers can abort the outer transaction. The implementation recognizes this only after classifying the SQL rejection, discards the uncommitted grouped attempt, and replays through individual reference transactions. Unknown storage, allocation or execution errors fail closed instead of becoming semantic fallback. No result from the discarded attempt is acknowledged. Session epochs are explicit ordered controls and preserve their own atomic fence transition rather than being treated as user SQL.

RELEASE ends a request's savepoint but never acknowledges it. Only a successful outer commit advances the published applied watermark. The durable host also requires the chosen journal frontier before application: in the separated store the application commit uses WAL NORMAL and the FULL journal barrier that precedes it provides durability (SOD 0005 M4, `JournalCache.tla`). If the process dies before acknowledgement, retry/recovery uses durable request identities and outcomes. Prefixes committed by reference fallback are recoverable even if a later request fails. This argument assumes SQLite savepoint/transaction semantics and storage honoring synchronization; the fault tests check implementation boundaries, not physical power-loss behavior.

Evidence map

`test_application_group_matches_individual_history` compares 192 requests through grouped and individual commits in each of the combined and separated storage formats. It compares every outcome, table/trigger data, generated-key counters, retry fences, revision and epoch, including periodic reopen. Histories mix immediate and deferred constraints, SQL expression/policy failures, rollback conflict modes, duplicates, identity collisions and sequence gaps.

`test_fts_group_and_reference_have_identical_logical_snapshot_state` compares the complete logical image and all outcomes for 32 FTS/vector/trigger transactions including updates, deletes and row reuse. Session-retirement tests add full session capacity, grouped controls, retry fences and restart. Optimistic-transaction tests validate every revision within an application group. ORM integration exercises commit, rollback, nested savepoints, generated keys, relationship/foreign-key writes, cross-voter conflicts and pending commit recovery.

The durability harness kills actual child processes at journal, application and acknowledgement boundaries for successful and rejected requests and groups. The native service and search suites additionally restart all voters. The evidence is tied to the recorded source hashes; final-candidate reruns remain R7. General SQL determinism and full resource qualification remain their separate existing criteria, not conclusions drawn solely from these finite mixed histories.