

Native SQL service

Written by Vikrant Rathore with assistance from Ronak Rathore. Updated 2026-09-25.

sqlodin serve is a standalone Odin process: it owns its disk database, durable Paxos voter, nonblocking TLS connections, SQL admission, read barriers, and results. Peers exchange consensus messages directly over mTLS. Python is a client, not the server or replication coordinator. The server accepts one to five fixed voters. End-to-end release qualification used three voters; targeted tests and models also cover five. See the release record for qualification evidence, measured performance and operational limits.

Build and run

```
git submodule update --init --recursive
# macOS and Linux: builds all pinned native static dependencies automatically.
python3 tools/build_cli.py
```

There is no system-library fallback. Build prerequisites are a C compiler, ar, Make and Perl in addition to Odin/Python. See the self-contained build guide. Embedded users importing src do not depend on OpenSSL. The installed Odin release has no core TLS package; this service uses a small OpenSSL 3 binding.

Declare the *same complete, sorted voter plan* on every server before creating any database. Example node 1 configuration (paths are explicit, not relative to the JSON):

```
{
  "cluster": "orders",
  "node": 1,
  "listen": "10.175.52.19:7600",
  "data": "/home/insan/projects/sqlodin/orders/node1",
  "certificate": "/home/insan/projects/sqlodin/orders/node1.pem",
  "key": "/home/insan/projects/sqlodin/orders/node1.key",
  "ca": "/home/insan/projects/sqlodin/orders/ca.pem",
  "members": [
    {"id": 1, "address": "10.175.52.19:7600", "identity": "node1.sqlodin.test"},
    {"id": 2, "address": "10.175.52.20:7600", "identity": "node2.sqlodin.test"},
    {"id": 3, "address": "10.175.52.21:7600", "identity": "node3.sqlodin.test"}
  ],
  "clients": ["app.orders.sqlodin.test"]
}
```

Provision CA-signed certificates before launch. Voter certificates need both serverAuth and clientAuth extended key usage and the exact configured DNS SAN; client certificates need clientAuth and an explicitly listed client SAN. Private keys stay on the relevant machines, with restrictive filesystem permissions. Keep the CA private key separate from the running service. The test tools create short-lived fixture certificates; they are not an operator enrollment or rotation system.

```
mkdir -m 700 /home/insan/projects/sqlodin/orders/node1
bin/sqlodin serve node1.json --create # first creation only
bin/sqlodin serve node1.json         # subsequent restart
```

Use each member's ID, listening address, database directory and key pair in its own configuration. --create refuses an existing database; ordinary startup refuses a missing database. Recovery verifies durable identity and complete history. SIGINT and SIGTERM leave the event loop and close the owned resources. In-flight requests may have unknown outcomes and require identity-preserving retries. TLS close-notify draining is not implemented.

SQL clients

The Python package provides the normal application interface: reusable connect, execute, query, named rows, atomic buffered transaction contexts, and explicit recovery of unknown write outcomes. It has no base runtime dependency and builds with uv. Version 0.2 adds typed Vector bindings, atomic FTS/content updates, exact distance queries and one-snapshot hybrid search. The optional SQLAlchemy dialect now defaults to optimistic SERIALIZABLE ORM/Core transactions, including generated keys, rollback and savepoints. See the ORM transaction contract for commit/retry behavior.

The native CLI also accepts a protocol request file:

```
{ "cluster": "orders", "address": "10.175.52.19:7600", "identity": "node1.sqlodin.test",  
  "certificate": "app.pem", "key": "app.key", "ca": "ca.pem" }  
  
{ "op": "query", "sql": "SELECT id, customer FROM orders WHERE id=?1",  
  "parameters": [ { "kind": "integer", "integer": 42 } ] }  
  
bin/sqlodin request client.json query.json
```

The CLI prints the JSON response, exits 0 for success, 1 for a server rejection or unknown-outcome response, and 2 when it cannot verify a response. An execute request must supply a nonzero 32-character lowercase hexadecimal session and a positive sequence starting at 1. Persist and retry the same file after an uncertain result; do not generate a new identity. Python handles these mechanics within a connection.

Protocol and correctness

Version 1 uses a little-endian unsigned 32-bit length followed by strict JSON inside TLS 1.3. Every request includes protocol: 1 and the configured cluster. Maximum request size is 64 KiB and maximum response size is 1 MiB. There is one outstanding application request per connection. Pipeline buffering is not an application API.

The authenticated exact SAN determines the connection role. A voter must complete a hello matching its configured node ID and compatibility fingerprint before any Paxos packets are accepted. The fingerprint includes network/encoding/policy versions, upstream pin, local SQLite build fingerprint and sorted membership. The packet's source must equal the authenticated voter; its destination must equal this node. SQL clients cannot send voter traffic, and voter connections cannot execute SQL.

execute converts a bounded transaction body and typed parameters into the existing durable mutation. Success requires the expected value to be chosen and locally applied, including a durable result. A displaced proposal is retried under the same request identity. A timeout never cancels an already proposed write. Client session identities are namespaced by authenticated client SAN, so cross-node retries retain their identity and a different client identity cannot reuse that session. Certificate rotation preserving the SAN preserves the namespace; a changed SAN does not.

query defaults to consistency: "linearizable". After invocation the service observes the highest seen slot of a read quorum (itself and at least one peer in a three-voter cluster), waits until its applied prefix reaches that frontier, then executes the read-only query in the same serialized event-loop turn. The barrier writes nothing and needs no sync (SOD 0005). Results include column names and typed values (Null/Integer/Real/Text/Blob; BLOB bytes are base64 on the wire). Queries return the whole bounded result or an error, never a partial success. consistency: "local" bypasses the quorum barrier and may be stale. status is local process state and an applied watermark, not proof of quorum health.

Ownership, limits, and remaining work

The server owns a fixed 32-entry connection array, per-connection request state, a 64-entry output ring and at most 2 MiB queued output per connection. Configuration has its own arena; transient decoding/results use the turn's temporary allocator. Borrowed TLS identities and decoded packets are consumed before their storage is released; durable mutation state is copied into the connection/host. A serialized loop owns all host and SQLite access. Poll-based nonblocking I/O, TCP_NODELAY, 5-second handshake timeouts, 120-second idle timeouts and request deadlines bound connection work. An unavailable peer is packet loss; upstream retransmission and catch-up handle reconnection. Storage failures stop the server.

Transaction limits remain 4096 SQL bytes, eight statements, sixteen parameters and 256 UTF-8 bytes per text parameter. Query results have a 4096-row and 256 KiB internal budget plus the read VM instruction limit. These bound individual work; they do not prove total RSS/CPU isolation. The service reserves peer admission, groups durable transitions, retires sessions through replicated epochs, and performs certified snapshots, generation publication and catch-up. Retained consensus history is capped at 8 GiB with admission reserves. Application data and staging space are separate. Unknown replicated execution or storage failures fail closed; arbitrary SQL and blocked kernel I/O have no fixed wall-clock guarantee. Fixed membership accepts writes through every voter but does not shard SQLite execution. The current contracts are resource limits, SQL policy, and backup, replacement and renewal.

Incus-style expiring single-use enrollment and named cluster administration remain a separate implementation step. A certificate only authenticates a machine. It must not implicitly change the voter set; dynamic add/replace requires a verified consensus membership transition and safe state transfer first. Do not change the members list in a running cluster to simulate expansion.

Interactive SQL client

`sqlodin connect CLIENT.json` provides the interactive/scriptable Odin client. Use `.help` for SQL, transaction, formatting, catalog and cluster commands. The CLI preserves uncertain writes in a private durable client state file; `.retry` resolves the exact identity. `sqlodin local` embeds the pinned SQLite shell for standalone files. See the complete CLI guide for commands and boundaries.

Verification

```
python3 tools/check_network_service.py --output new-service-report.json
python3 tools/check_network_hosts.py --hosts user@host1 user@host2 user@host3 \
  --binary build/linux-sqlodin --output new-three-host-report.json
cd languages/python && uv sync --extra test && uv run pytest && uv build
```

The first test uses three native processes and persistent-disk directories (Linux rejects tmpfs/ramfs). It covers native CLI requests, writes through every voter, typed rows, rollback, result/SQL limits, role authorization, malformed frames, six concurrent mixed-workload sessions, cross-master retries, minority refusal, and all-process SIGKILL/reopen convergence. The second runs those network scenarios on three explicit Linux hosts, with direct mTLS traffic and five-minute child watchdogs. These are correctness checks, not comparative performance benchmarks or production certification.

Search compatibility

Current durable identity and peer fingerprints use **format 5 / policy 9**. Built-in FTS5 creation is admitted; defensive mode protects its shadow tables, and duplicate FTS row IDs become durable constraint rejections. Other virtual modules, including vec0, remain restricted. Vectors bind as `{"kind": "vector", "vector": [1, 0, 0]}` and use the existing fixed float32 mutation storage (384 total components per request). Older-format/policy stores and peers are rejected. There is no

automatic or rolling migration. Use the explicit offline format-4 migration procedure and retain the source; other incompatible engine identities require a separately validated migration.

The search client maintains content and FTS rows atomically; direct SQL can bypass that relationship. Search functions run on the read connection. Large exact scans may exceed the existing read budget; result limits do not imply ANN indexing. FTS custom tokenizers and maintenance commands are outside the tested API.

Optimistic transaction protocol

`begin` takes a fresh quorum barrier and returns `read_version`. `preview` carries that version, the staged `sql/parameters`, and optional `read_sql/read_parameters`. A fresh barrier precedes private evaluation; its SQLite transaction always rolls back. The response includes outer-statement changes, `lastrowid`, and optional result rows. Neither operation durably publishes application changes. Preview work is VM-budgeted.

`execute` with a nonzero `read_version` performs a conditional durable commit of the complete body. The ordered applier returns a persisted `Conflict` when the revision changed. Zero retains the unconditional native batch API. The revision is part of both the explicit mutation codec and request digest. Read-only commit and savepoint bookkeeping need no extra protocol operation: no server transaction is held open.