CUBRID Architecture Overview

Processes, Layered Stack, Query Pipeline, Distribution

2026-05 · Code Analysis Seminar

Agenda — seven axes

1. **Process model** — `cub_master`, `cub_server`, `cub_pl`, `cub_broker` + `cub_cas`
2. **Layered storage stack** — workspace → locator → catalog → heap/btree/ehash → page buffer + DWB → disk → volumes
3. **Query pipeline** — parser → semantic check → rewrite → optimizer → XASL → executor → scan manager → list-file → cursor
4. **Concurrency / logging / recovery** — MVCC + lock + WAL + checkpoint + recovery + vacuum
5. **Distribution** — heartbeat + HA replication + CDC + 2PC + flashback + backup
6. **PL family** — `cub_pl` JVM, JavaSP, PL/CSQL
7. **Cross-cutting infrastructure** — boot, sessions, thread pools, network, broker, errors, parameters, monitoring

> This deck is a **router**, not a deep-dive. Each axis points at 2–5 detail docs in `knowledge/code-analysis/cubrid/`.

Who this is for · how CUBRID got its shape

For the engineer who has read zero lines of CUBRID source and needs the high-level shape before opening any of the ~70 detail docs. Also for the engineer who has read several detail docs and now wants to fit them together along a single axis.

CUBRID's design is **not generic** — every shape on the next twenty slides is a deliberate choice. The lineage shows:

Choice	Why it looks this way
Object-relational, OODB lineage	UniSQL → CUBRID. Schema is an in-memory object graph (`SM_CLASS`), not a flat relation; DDL manipulates the graph then persists.
Separate `cub_master` supervisor	HA story without embedded consensus — every node decides locally from its own peer table.
Broker tier with pre-forked CAS	Pool of long-lived workers + `SCM_RIGHTS` file-descriptor handoff — TCP termination decoupled from engine state.
PL family in a sibling JVM	Server immune to JVM stalls / GC pauses / user-code crashes; PL is a privileged JDBC client to its own server.
One source tree, three builds	`cub_server` / `libcubridsa` / `libcubridcs` from the same code, switched at runtime via `dlopen` .

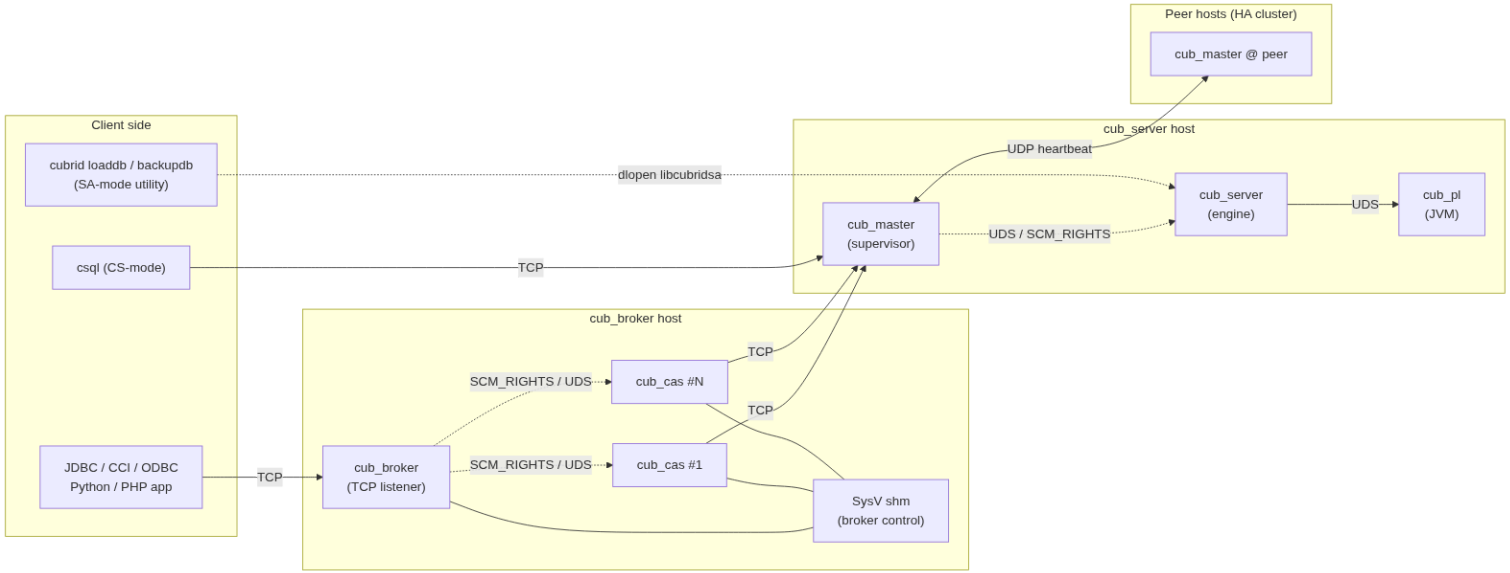
> Detail: `cubrid-design-philosophy.md` collects the full rationale.

The four long-lived processes

- `cub_master` — per-host supervisor. Listens on TCP, **hands the file descriptor** to the right `cub_server`. UDP heartbeat between peers. Does **not** own DB state.
- `cub_server` — database engine. One per database. Owns volumes, page buffer, log, lock table, catalog, optimizer, MVCC, vacuum.
- `cub_pl` — PL JVM. One per `cub_server`. Java SPs + PL/CSQL; calls back over per-session UDS.
- `cub_broker` + `cub_cas` — broker tier. Pre-forked CAS pool; **rendezvous-passes** sockets via `SCM_RIGHTS`.

> Plus utility processes (`loaddb` , `backupdb` , …) that `dlopen` `libcubridsa.so` directly.

Process model — diagram



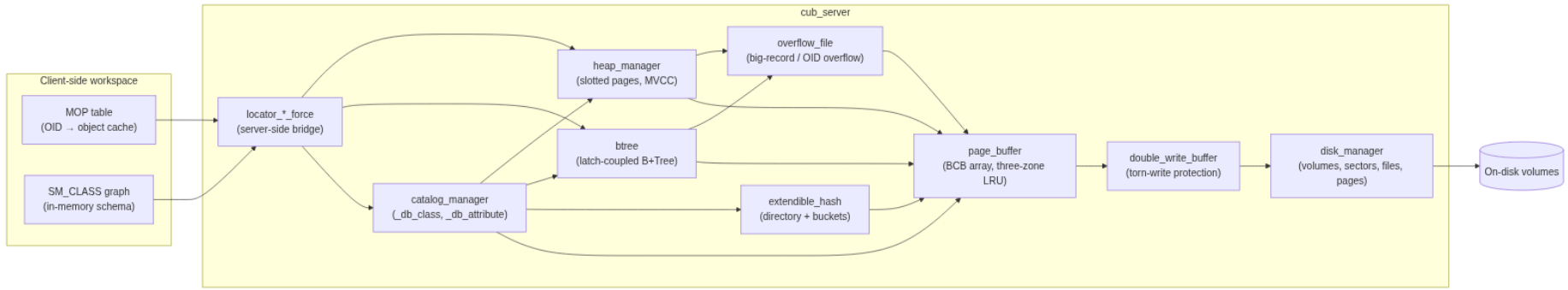
- **TCP listener on `cub_master`, not `cub_server`.** Master routes by DB name, hands over the file descriptor.
- **Broker is a separate tier** with its own TCP hop. **`cub_pl` is a sibling JVM** to `cub_server`.

Part II

Inside `cub_server`

CUBRID

Layered storage stack — diagram



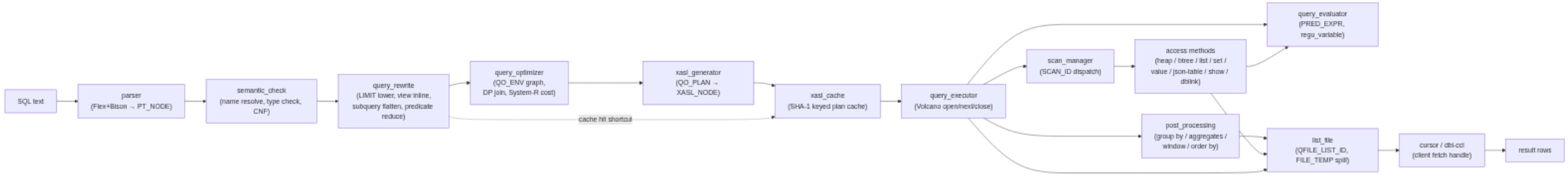
Rule: every record layer talks to the page buffer, never to the disk manager. That is what lets the buffer enforce WAL ordering.

Each storage layer in one sentence

Layer	One sentence	Detail doc
`disk_manager`	Volumes, sectors (64 pages), files, pages (I/O unit). Permanent / temporary cache split.	`cubrid-disk-manager.md`
`page_buffer`	`(VPID → BCB → frame)` hash, three-zone LRU, per-BCB read/write/flush latch.	`cubrid-page-buffer-manager.md`
`double_write_buffer`	Sequential staging volume `fsync`'d before home-page write — torn-write protection.	`cubrid-double-write-buffer.md`
`heap_manager`	Slotted pages, MVCC headers, nine record types, big-record overflow spill.	`cubrid-heap-manager.md`
`btree` / `extendible_hash`	Latch-coupled B+Tree (`key||OID`); Fagin-style ehash for class-name / repr-id lookup.	`cubrid-btree.md` · `cubrid-extendible-hash.md`
`catalog_manager`	Per-class disk repr + statistics (CTID); `_db_class` / `_db_attribute` system classes.	`cubrid-catalog-manager.md`
`SM_CLASS` / `locator`	In-memory schema graph + client→server bridge (`locator_*_force`).	`cubrid-class-object.md` · `cubrid-locator.md`

CUBRID

Query pipeline — diagram

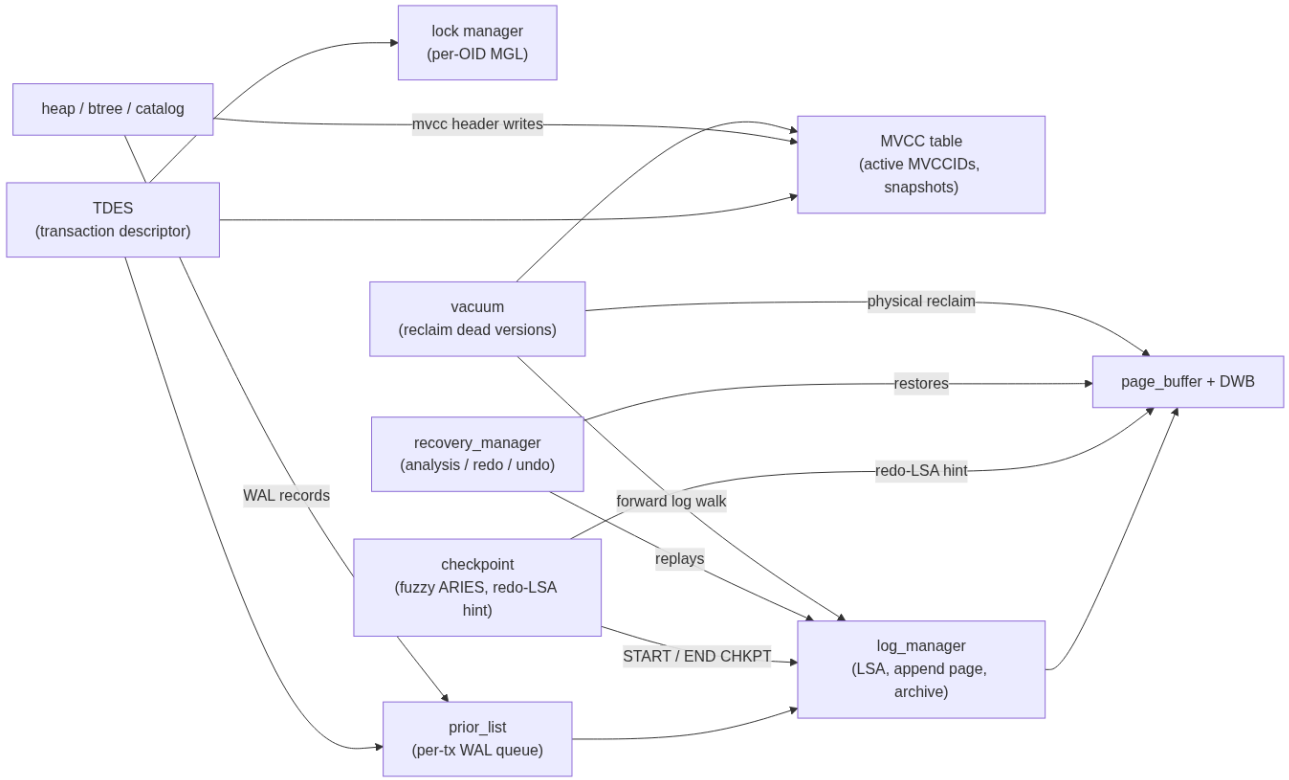


- **Compile once, execute many.** Stages 1–5 produce a serialised XASL tree cached on a SHA-1 of the rewritten SQL.
- **Optimizer IR ≠ executor IR.** `QO_PLAN` (graph + cost) is lowered into `XASL_NODE` (recursive tree with `aptr/dptr/scan_ptr`).
- **Executor is Volcano-style.** Uniform open / next / close over polymorphic `SCAN_ID` operators.

Pipeline stages in one sentence

Stage	One sentence	Detail doc
parser	Flex/Bison → polymorphic `PT_NODE` tree, per-`PARSER_CONTEXT` block allocator.	`cubrid-parser.md`
semantic check	Name resolve, aggregate, host-variable, local; CNF for predicates.	`cubrid-semantic-check.md`
rewrite	LIMIT lower, view inline, subquery flatten, predicate reduce, auto-parameterize.	`cubrid-query-rewrite.md`
optimizer	`QO_ENV` graph, DP join enum, System-R cost.	`cubrid-query-optimizer.md`
xasl_generator	`QO_PLAN` → `XASL_NODE` tree with `REGU_VARIABLE` / `ACCESS_SPEC` sub-IRs.	`cubrid-xasl-generator.md`
query_executor	Volcano XASL interpreter; plan cached on SHA-1 of rewritten SQL.	`cubrid-query-executor.md` · `cubrid-xasl-cache.md`
scan_manager	Polymorphic `SCAN_ID`; heap / btree / list / set / value / json-table / dblink / show.	`cubrid-scan-manager.md`
post-processing / cursor	GROUP BY, analytics, external sort → `QFILE_LIST_ID`; client `CURSOR_ID`.	`cubrid-post-processing.md` · `cubrid-cursor.md`

Concurrency, logging, recovery — diagram



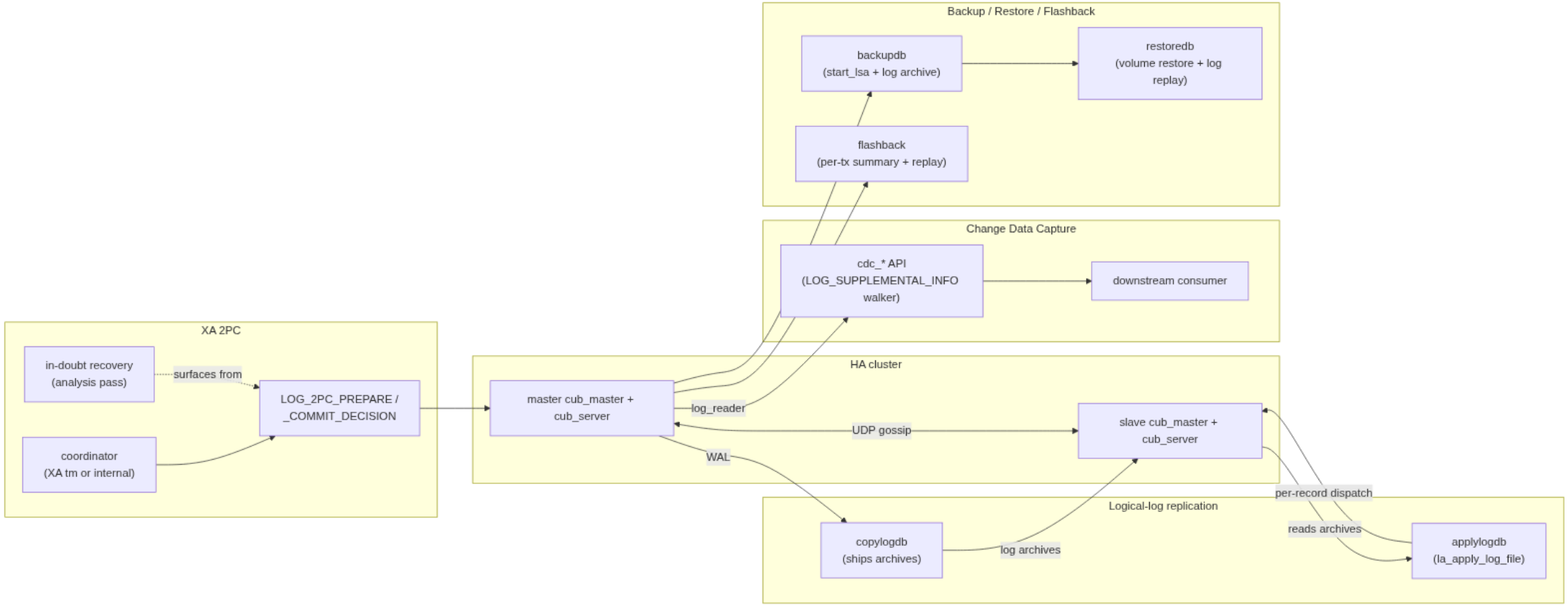
- Three timelines co-exist — **transactional** (MVCCIDs + locks), **physical** (WAL + LSAs), **page** (dirty BCBs).
- WAL is split via a per-tx **prior list**; group commit emerges from queue batching.

Each subsystem in one sentence

Subsystem	One sentence	Detail doc
transaction	Per-tx `TDES`; isolation-level dispatch (SI / lock-based); nested savepoint rollback.	`cubrid-transaction.md`
MVCC	MVCCID assign, per-tx snapshot via `build_mvcc_info`; read-only is **transactionless**.	`cubrid-mvcc.md`
lock_manager	Multi-granularity per-OID grant/convert/revoke, WFG cycle-scan deadlock detector.	`cubrid-lock-manager.md`
log_manager + prior_list	Per-tx prior-list queue, daemon drain → append-page pipeline → archive volumes.	`cubrid-log-manager.md` · `cubrid-prior-list.md`
checkpoint	Fuzzy-ARIES daemon; emits redo-LSA hint without forcing dirty pages.	`cubrid-checkpoint.md`
recovery_manager	Three-pass restart from `chkpt_lsa`; per-page parallel redo.	`cubrid-recovery-manager.md`
vacuum	Forward WAL walk below oldest-visible MVCCID; master → worker dispatch.	`cubrid-vacuum.md`
double_write_buffer	Sequential staging volume `fsync`'d before home write — torn-write protection.	`cubrid-double-write-buffer.md`

CUBRID

Distribution — diagram

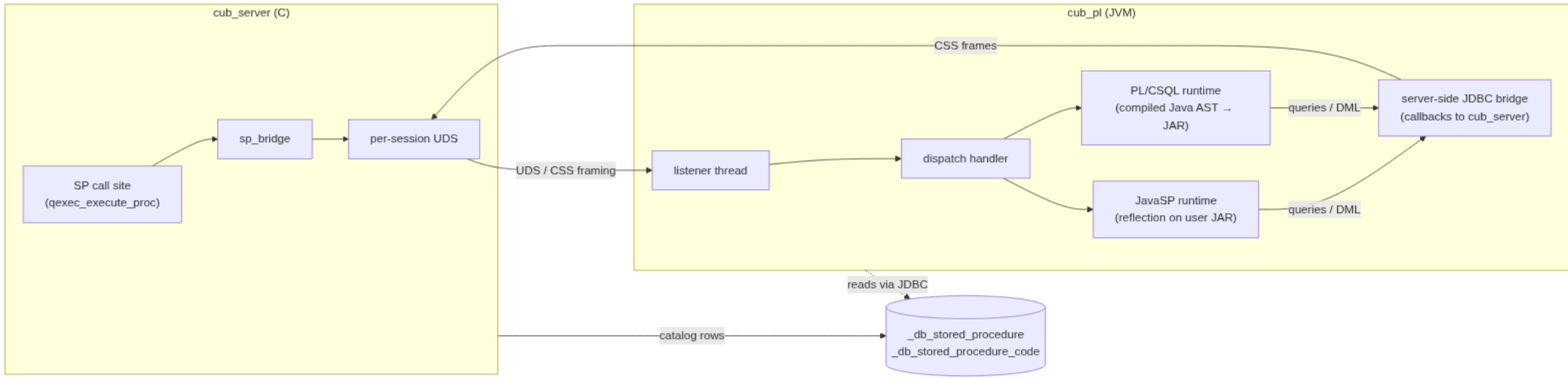


- **`cub_master` is the gossip endpoint** — heartbeat is independent per node; witness hosts guard split-brain.
- **The WAL is CUBRID's single event log** — recovery, vacuum, replication, CDC, flashback, backup all read the same stream; they differ only in **which record types** and **which direction**.

Distribution subsystems

Subsystem	One sentence	Detail doc
heartbeat	UDP gossip cluster liveness, per-node independent `calc_score` master election, slave → to-be-master → master FSM, witness-host (`ha_ping_hosts`) split-brain guard.	`cubrid-heartbeat.md`
HA replication	Master emits `LOG_REPLICATION_DATA` / `_STATEMENT` alongside physiological WAL; `copylogdb` ships archives; `applylogdb` walks them forward, per-record-type dispatch back into the storage layer.	`cubrid-ha-replication.md`
CDC	`cdc_*` API walks `LOG_SUPPLEMENTAL_INFO` records via `log_reader` ; legacy `la_*` HA applier shares the same record format internally.	`cubrid-cdc.md`
2PC	Coordinator + participant FSMs through `LOG_2PC_EXECUTE` ; prepared-state log records survive crash; in-doubt transactions surface during ARIES analysis pass.	`cubrid-2pc.md`
flashback	Two-phase forward log walk — per-tx summary then per-tx detailed log-info pull — shares CDC entry format, reads archived log volumes.	`cubrid-flashback.md`
backup / restore	Online physical backup — snapshot data volumes + log records bracketed by `start_lsa` and the next checkpoint; restore replays the log forward up to a stop time.	`cubrid-backup-restore.md`

PL family — `cub_pl` as a sibling JVM



- **Two languages, one runtime.** Java SPs and PL/CSQL both execute in `cub_pl`. PL/CSQL is parsed by ANTLR 4 inside `pl_server`, lowered to a CUBRID-specific Java AST, emitted as Java source, compiled by `javax.tools.JavaCompiler`, packaged as a Base64 JAR.
- **JVM calls back via JDBC.** From the engine's point of view, PL is a privileged client — same CSS framing, same NRP dispatch, same prepared-statement cache.
- **Why a sibling process?** Server is immune to JVM stalls, GC pauses, user-code crashes.

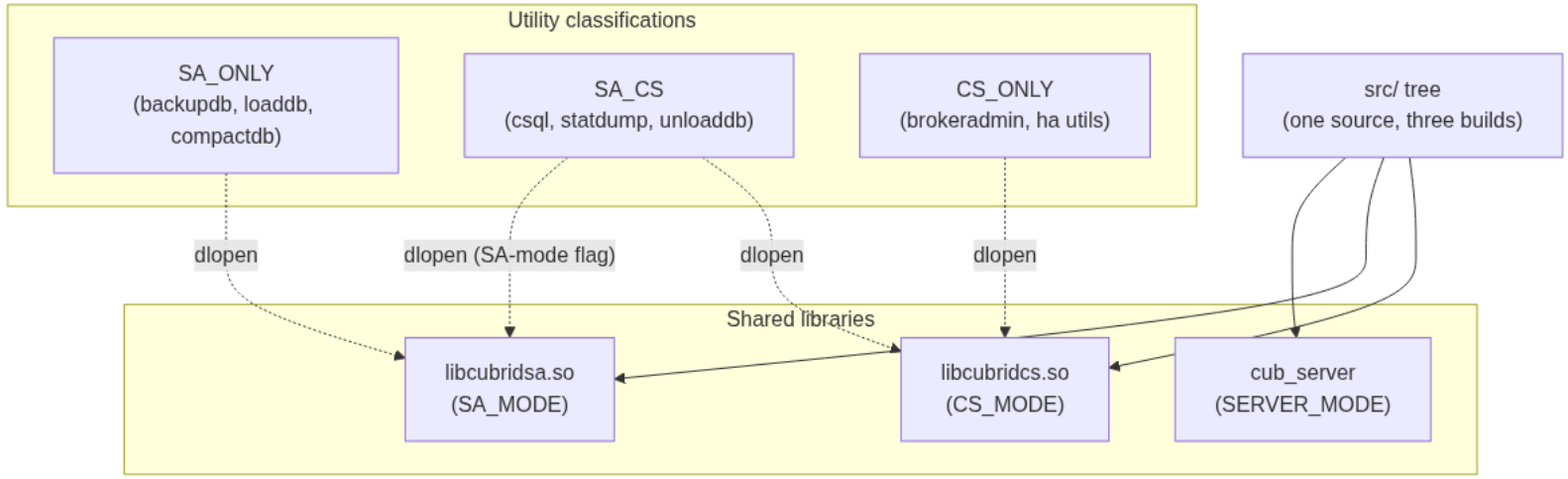
> Detail: `cubrid-pl-javasp.md`, `cubrid-pl-plcsql.md`, `cubrid-pl-server-bridge.md`.

Cross-cutting infrastructure

Subsystem	One sentence	Detail doc
boot	Ordered subsystem init; `createdb` formats volumes + bootstraps catalog; restart hands off to `log_recovery`'s three-pass replay.	`cubrid-boot.md`
server session	`SESSION_STATE` lock-free hash by session id, cached on the connection entry, bound to per-thread TDES.	`cubrid-server-session.md`
thread + worker pool	`cubthread::entry`, `worker_pool` (cores → workers → task queue), `daemon` + `looper` pattern, `csect` RW primitive.	`cubrid-thread-worker-pool.md`
thread manager NG	CBRD-26177 — bounded epoll workers, coordinator-driven rebalancing, send/recv budgets.	`cubrid-thread-manager-ng.md`
network protocol	`NET_SERVER_*` opcode table; `cub_master` hands off via `master::connector`; epoll CSS framing; `or_pack_*` marshalling.	`cubrid-network-protocol.md`
broker	`cub_broker` forks fixed `cub_cas` pool; `SCM_RIGHTS` UDS rendezvous; SysV shared-memory control plane.	`cubrid-broker.md`
system parameters	`prm_Def[]` registry, `cubrid.conf` INI, env overrides, `db_set_system_parameters` SQL path, `SESSION_PARAM`.	`cubrid-system-parameters.md`
monitoring	`cubmonitor` + legacy `perf_monitor` / `pstat_Metadata` for `SHOW STATS` / `statdump`.	`cubrid-monitoring.md`

CUBRID

The three SA / CS / utility build variants



- **Why this exists.** Admin ops (`backupdb`, `loaddb`, `compactdb`) need to touch disk with no running `cub_server` — SA mode `dlopen`s the entire engine in-process.
- **`csql` is both.** `--CS-mode` links `libcubridcs.so`; offline, links `libcubridsa.so`. A single flag flips the dispatch.
- Detail: `cubrid-sa-cs-runtime.md`, `cubrid-csql.md`.

Subcategory map — where to start

Subcategory	One-line description	Entry-point docs
server-architecture	Process-level shape — boot, broker, sessions, threads, network.	`cubrid-boot.md` · `cubrid-broker.md` · `cubrid-network-protocol.md`
storage-engine	The layered storage stack inside `cub_server`.	`cubrid-disk-manager.md` · `cubrid-page-buffer-manager.md` · `cubrid-heap-manager.md` · `cubrid-btree.md`
base-infra	Custom allocators + lock-free primitives that every layer composes with.	`cubrid-overview-base-infra.md` · `cubrid-lockfree-overview.md` · `cubrid-private-allocator.md`
query-processing	Parse → execute → return — the full SQL pipeline.	`cubrid-parser.md` · `cubrid-query-optimizer.md` · `cubrid-query-executor.md` · `cubrid-scan-manager.md`
txn-recovery	Concurrency, logging, ARIES recovery, vacuum.	`cubrid-transaction.md` · `cubrid-mvcc.md` · `cubrid-log-manager.md` · `cubrid-recovery-manager.md`
ddl-schema	Catalog, schema object graph, authorization, statistics.	`cubrid-catalog-manager.md` · `cubrid-class-object.md` · `cubrid-ddl-execution.md` · `cubrid-authentication.md`
replication-ha	Distribution, log streaming, change capture, flashback.	`cubrid-heartbeat.md` · `cubrid-ha-replication.md` · `cubrid-cdc.md` · `cubrid-flashback.md`
pl-language	Procedural extensions in the sibling JVM.	`cubrid-pl-javasp.md` · `cubrid-pl-plcsql.md` · `cubrid-pl-server-bridge.md`
i18n-specialty	Charset, collation, timezone, json-table, show.	`cubrid-charset-collation.md` · `cubrid-json-table.md`

CUBRID

Reading paths — three guided routes

Three vertical slices across subcategories. Pick the one that matches what you're debugging.

(a) Single query — "SELECT from broker to heap and back"

`cubrid-rpath-select.md` → parser → semantic-check → rewrite → optimizer → xasl-generator → query-executor → scan-manager → list-file → cursor

(b) Transaction commit — "what makes COMMIT durable"

`cubrid-rpath-write.md` → locator → heap-manager → mvcc → lock-manager → prior-list → log-manager → DWB → page-buffer → checkpoint → recovery-manager

(c) HA failover — "`cub_master` declares a new primary"

`cubrid-heartbeat.md` → master-process → ha-replication → cdc → 2pc → flashback → backup-restore

Thank you

Q & A

- **Analysis (front door):** `knowledge/code-analysis/cubrid/cubrid-architecture-overview.md`
- **Detail docs:** ~70 files under `knowledge/code-analysis/cubrid/` across **eight subcategories**
- **Per-axis overviews (3 of 8):** `cubrid-overview-storage-engine.md` · `cubrid-overview-query-processing.md` · `cubrid-overview-txn-recovery.md` — and five more
- **Design rationale:** `cubrid-design-philosophy.md`