

Two databases for programs and proofs: a design-delta survey of geolog-alpha and Coln

deltas in both directions, at pinned revisions, with the reasoning behind each
divergence steelmanned before it is judged

davidad's AI ensemble*

MAITRIA — Mathematical Assurance and Interaction Toolkit for Reliable Intelligent Agents

2 August 2026 — draft for discussion

1 Scope, sources, and how claims are graded

This report compares the designs of two databases that grew from one brief: **Coln** (github.com/coln-project/coln, the Safeguarded AI programme's categorical-core implementation, developed by Martin Kleppmann, Owen Lynch, and collaborators), and **geolog-alpha** (the maitria coalition's chase-engine database, “a local-first database whose tables are logical consequences”). Both descend from the geolog design lineage — Coln's own foundational notes still use the name — and both aim at the same niche: a local-first database whose data model is a logical theory, built to hold programs and proofs rather than JSON. They have diverged substantially, and the divergences are the subject.

The comparison is pinned:

- Coln at commit `654a184` (2026-07-28, the public repository head at survey time), read together with its full manual (69 numbered trees, served at manual.geolog.sgai.uk) and its design-note corpus (`notes/`: *Why* (Kleppmann), *Prospectus* (Lynch), *Vision* (Deikun), the geometric-type-theory notes, the merge notes). Implementation packages were read at source, with particular attention to the survey window: the commits that entered the default branch's history after an earlier survey pin at `293c915` (2026-07-17) — 80 commits by ancestry count, merges included and some authored earlier on branches, the last landing 2026-07-28; +14,061 lines by pin-to-pin diffstat. Both numbers reproduce as `git rev-list -count` and `git diff -stat` between the two pins.
- geolog-alpha at revision `b4c7fa2` (2026-08-02): the full crate workspace (28 crates), `README.md`, `ENGINEERING.md`, the committed theories, and the companion book's Geolog part.

Three claim grades are used throughout. **[observed]** means read at the pinned source, with the path stated; a tag governs the specific factual claim of the sentence it closes, not every clause of its paragraph. **[inferred]** means inferred from observed artifacts; the inference is stated so it can be attacked. **[judgment]** means an engineering judgment by the authors; these are the claims a reader should discount first, and they are labeled so that she can. Steelman and judgment paragraphs are judgment as a class; grades inside them mark only their factual citations. One scope rule, stated once: claims about the companion repositories of either project (geolog's proof-coupled companion, Coln's upstream Automerge-lineage components) are reported at the level of what the surveyed repositories and documents themselves assert, as their surfaces stood on the survey date, and are not independently re-audited here. In particular, no finished end-to-end verified checker is claimed anywhere in this report, for either project.

Bias disclosure. The authors work within the geolog lineage, and a comparison written by one side of a divergence has a predictable failure mode: the home team's choices get argued, the other team's get summarized. Three disciplines are applied against it. First, every claimed geolog advantage is preceded by a steelman of Coln's choice — why a competent team chose it and what it buys them — written before the

*Drafted, source-surveyed, and revised primarily on Claude Fable 5; an adversarial review round on GPT-5.6 Sol was commissioned with explicit instructions to hunt home-team bias, and its findings are folded in. The authors work within the geolog lineage; §1 states the resulting discipline.

judgment. Second, every “geolog should replicate this” recommendation is priced honestly against geolog’s architecture, including what it would break. Third, the deltas run in both directions, and the direction favorable to Coln (§6) is placed before the direction favorable to geolog (§7). Residual bias should be assumed anyway.

This report describes designs, recommends candidate transfers, and names current deltas. Organizational conclusions of any kind are outside its scope; the recommendations are design observations for whoever finds them useful, not a plan for either team.

2 One brief, two theses

Both projects answer the same question — *what should a database be, if its contents are programs, proofs, and models rather than records?* — and both take from Steve Vickers the founding conviction that geometric logic is the logic of observable data. From there they part, and the cleanest way to state the whole report is to state the two theses.

Coln’s thesis: a language, riding proven storage. Coln calls itself “a data-oriented proof assistant” [observed] (README.md) and, in its manual’s opening tree, “a database with an expressive language for schemas, queries, and migrations” [observed] (manual/trees/0001.tree). Its foundational tree gives six self-descriptions, among them: a dependent type theory cut down to what fits in a database; a module system for egglog/eqlog-style reasoning; a proof assistant with quotient inductive-inductive signatures as a first-class concept; “automerge for acsets” [observed] (002H.tree). The strategy note is explicit about the division of labor: Automerge is named the closest existing system, and the plan is to “reuse the infrastructure we have already built for Automerge (physical data layout, versioning mechanisms, compression, replication protocols, authentication and access control, sync servers and clients, user interface tooling, etc.), and replace only the JSON data model layer” [observed] (notes/why.tex). The center of gravity is therefore the *language* [inferred] (from where the implementation effort and the manual’s depth actually sit): an elaborator from a typed surface syntax to a relational IR, with storage, sync, and access control drawn from the Ink & Switch / Automerge lineage — partly today (a working Subduction sync round-trip is in-repo, storing commit chunks as Sedimentree loose commits [observed] (test_subduction.rs)), mostly by stated plan (Keyhive for authentication and access control is planned, not integrated) — and trust vested in the core calculus now, with succinct-proof compression named as future work.

geolog’s thesis: a log, held to logical discipline. geolog-alpha’s self-description is “a local-first database whose tables are logical consequences” [observed] (README.md): a theory in a geometric fragment of first-order logic, plus a log of signed, content-addressed commits; the engine computes a chase fixpoint — in its README’s gloss, the least set of derived facts the axioms force; more precisely, for the existential fragment, a universal model canonical up to the homomorphic-equivalence contract its chase documentation states. Tables are not authored; they are entailed. Merge is set union on commits followed by re-derivation. The repository positions itself equally as infrastructure (“a principled replacement for event-sourcing spines”) and as the untrusted production tier of a larger assurance architecture: certificates for claims about dynamical systems are ordinary rows in the same database, and independent checkers — among them a certified binary whose soundness theorem is being assembled in HOL4 down to machine code — re-derive those claims without trusting the engine [observed] (README “Trust and auditability”; the companion book). The center of gravity is therefore the *log and its consequences* [inferred] (symmetrically, from where the implementation effort sits): signed identity, re-derivability, and refusal disciplines first; surface language and interactive tooling second.

The compressed delta. Coln spends its novelty budget on the expressiveness frontier (how much of dependent type theory fits in a database) and aims to obtain distribution and trust by integrating an existing stack. geolog spends its novelty budget on the assurance frontier (how little must be trusted for derived state and certificates to be believed) and keeps its logic deliberately small, with typed, engine-owned escape hatches where the small logic cannot say what is needed. Most of the individual deltas below are corollaries of this one sentence. [judgment]

3 Where the designs agree

Six load-bearing choices are shared — worth listing first, because they mark out the region both projects treat as settled, and because that region is where ideas (and potentially code) move cheapest between

them. Given the shared lineage, commonality here may be inherited rather than independently re-derived; establishing genuine item-by-item independence would need dated design records this survey did not attempt, so the list claims sharedness only. **[inferred]**

1. **The commit graph is the ground truth; tables are replayed views.** Coln’s store documentation: the commit graph keeps the durable sequence of changes while the tables keep the current replayed state **[observed]** (`coln-store`). geolog: the rebuild invariant — all derived state re-derivable from the commit bag in any order, incremental-vs-rebuilt diffed in CI, divergence release-blocking **[observed]** (`ENGINEERING.md #4`).
2. **Content-addressed identity at the commit layer.** Coln: a store’s document identity is its root-commit hash **[observed]** (`test_subduction.rs`; `why.tex`’s versioning story); geolog: commit identity is the hash of canonical envelope bytes, and element identity is the declaring commit’s content address **[observed]** (`geolog-sync README`). A related but distinct move — structural row deduplication by hash-consing with union-find canonical representatives — landed in Coln’s store in July **[observed]** (`roweq/`); whether its canonical row identity is stable across replicas is a contract the code does not yet state, so the two projects’ row-level identity stories are not claimed equivalent here.
3. **The monotone (geometric) fragment is what makes incremental *checking* honest.** Kleppmann’s note argues constraints can be checked incrementally “because the proof checker is guaranteed to be *logically monotonic*” **[observed]** (`why.tex`) — new facts never invalidate old consequences, the CALM-shaped property. geolog’s chase and merge story rest on the same observation. Neither project claims retraction is monotone; both keep it outside the fragment (§4.3).
4. **Congruence closure belongs near the store.** Coln landed hash-consed rows with a union-find of canonical representatives in the storage layer **[observed]** (`roweq/rowing.rs`); geolog’s chase crate carries congruence closure beside sequent classification **[observed]** (`workspace table, README`).
5. **Functions lower to relations; three table shapes.** Coln’s $\Sigma\Pi$ normal form derives the arg-keyed / keyless / all-key table shapes and generates migration views together with the normalization **[observed]** (`manual 001T–001X`); geolog’s meta-schema stores function and relation rows in the same three shapes. Same transform, discovered twice.
6. **Documentation is authority, not commentary.** Coln: “if the code deviates from the manual then the code is considered to have a bug” **[observed]** (`manual practice`); geolog: design claims carry citations to in-repo design records, deviations are marked with reasons **[observed]** (`ENGINEERING.md #7`).

4 The deltas, dimension by dimension

4.1 Data model and semantic foundation

Coln builds a two-level dependent type theory sized to fit a database: matter-levels (`query < theory < top`) and homotopy-levels (`unit < prop < set`), with function domains restricted to the query level; theories parameterized by *models* of other theories as typed citizens; an `@ind modality` — a multimodal adjoint type theory with a lock/unlock adjunction, whose modal Π admits locked variables in domains but never codomains — giving a principled account of “you may eliminate over a completed carrier but only introduce into one still being defined”; an `init` former denoting initial models, with inductivity tracked as a theory-level attribute so that `init` on an inductive theory is rejected at elaboration; and quotient inductive-inductive signatures as the first-class notion of inductive definition **[observed]** (`manual 001G–001S, 002A–002I`; `notes/initial-models.tex, geott.tex`). Since mid-July the elaborator for this language has been hardened by a large golden-test battery — 270 new test files including proof irrelevance, records and theories parameterized by models, and nested theories **[observed]** (repository log, 2026-07-17 through 2026-07-28).

geolog keeps the axiom language to the geometric fragment and puts its foundational effort into *self-hosting*: the schema language is itself a theory (the meta-schema), so theories, extensions, and migrations are rows in the same substrate they govern **[observed]** (`README`; `theories/GeologMeta.geolog`). Each axiom carries a mode — **chase** fires and derives; **report** judges and never fires; the default is **report**, so derivation is always an explicit choice **[observed]** (`README`). What the geometric fragment cannot say enters as declared, typed machinery rather than as logic: aggregation nodes (whole-fibre folds over ordered semirings, staged so that a fold’s result is defined, never asserted — derived rows cannot

collide with the commit log) **[observed]** (geolog-agg README); ground vocabulary decided by external procedures behind a binder with refusal-over-guessing semantics (§4.5); and a tensor surface — einsum-style contraction atoms and fold operators in the theory grammar — exercised by a committed simplicial-complex theory whose $d^2 = 0$ law is stated as contraction rows **[observed]** (geolog-normalize surface battery; theories/SComplex.geolog).

Steelman of Coln’s choice. The type-theoretic route gives one uniform account of parameterization, initiality, quotients, and modularity — phenomena geolog handles by separate, engine-owned machinery — and it is worked out in Coln’s notes to a depth that has no public geolog counterpart. If the product is a language in which mathematicians and program analyses state inductive structures, a foundation in which “transitive closure” is one `init` applied to one parameterized theory is a genuinely beautiful answer, and elaborator-first sequencing is the coherent build order for it.

Judgment. These are two answers to the same question — what to do about what geometric logic cannot say — and they are not symmetric in risk. Coln climbs the type-theory ladder, which buys expressiveness whose operational cost arrives later (`init` denotes a least fixpoint no shipped Coln component yet computes; §4.3). geolog bolts on typed machinery, which buys computability now at the cost of a less unified story — three escape hatches (folds, oracles, tensors) instead of one calculus. The `@ind/init` discipline itself, however, is detachable from the rest of Coln’s ladder, and geolog’s parameterized-theory scoping rules would be strengthened by adopting its reading (§6, R5). **[judgment]** Meanwhile the self-hosting asymmetry runs the other way: a Coln store is born from a root commit carrying its schema — one store, one schema, with migrations existing today as language-level paired views and no store-level schema-evolution story **[observed]** (coln-store docs; manual 001X) — while geolog *represents* schemas, extensions, and migrations in-band, as rows under the same log discipline as the data, with a revision seam (declaration index, revision-body builder) in the workspace **[observed]** (README; geolog-revision). Neither tree yet demonstrates a long-lived instance evolving its active schema end-to-end without resettlement — rebuilding the store from scratch under the new schema — so the honest statement is positional: in-band representation appears to offer the more direct path to live schema evolution; neither repository yet establishes the storage changes its own path would require. **[judgment]**

4.2 Identity, authorship, and byte-level trust

Coln’s commits carry a 32-byte author field that is, at the pinned revision, “a placeholder of all zeros” **[observed]** (coln-store/src/commit/mod.rs); commits are content-addressed but unsigned; the in-repo sync test runs an open admission policy; and access control is planned as Keyhive, the Automerge lineage’s capability system, per the strategy note **[observed]** (why.tex).

geolog’s commits are self-authenticating ed25519-signed envelopes: the signature covers the canonical skeleton bytes, decoding rejects non-canonical forms, and nothing about validity depends on who delivered the bytes — “transport moves bytes; admission establishes truth locally” **[observed]** (geolog-sync README). One chain per honest author is an invariant whose violation a replica *detects and reports as equivocation* once it has seen both signed forks; the protocol neither excludes the forks nor provides global agreement or remediation — detection is a local, incremental judgment **[observed]** (same README, which states this limitation itself). Two scope limitations that same document states, carried here because omitting them would slant the comparison: realm attribution (which partition an entry belongs to) is *not* in the signed skeleton — a hostile relay could relabel a valid history, and signed realm binding is planned, not held; and the current transport profiles assume a trusted deployment boundary **[observed]** (the README’s own honesty ledger). A capability-algebra crate (grades, judgments, replay) ships in the workspace **[observed]** (geolog-caps).

Steelman. Coln’s strategy note explicitly sequences authentication and access control behind planned Keyhive reuse while current implementation effort concentrates on the language and query engines **[observed]** (why.tex). Keyhive is the Automerge lineage’s capability system; reusing it avoids designing authentication twice, and a pre-alpha that pretended to a trust story it does not yet have would be worse than placeholder zeros that say what they are. Kleppmann’s published work on Byzantine-fault-tolerant replication is part of what the reuse strategy aims to inherit.

Judgment. The asymmetry is nonetheless real and load-bearing. geolog treats authorship as semantics: merge behavior, provenance weights, and equivocation judgments *consume* the author identity, so it must exist before those layers are designed — and it does. Coln will eventually thread Keyhive’s identity notion under a data model whose merge tiebreaks descend from Automerge’s (counter, actor) operation identifiers; that retrofit is real work, and until it happens the store’s operations carry no cryptographic authorship at all. Front-loading signed identity buys freedom from a later semantic retrofit at the cost of building identity machinery early; deferring it buys elaborator progress at the cost of that retrofit. This report weighs the

first tradeoff higher for the shared ambition — databases whose contents include proofs, written by mutually untrusting parties including machine agents — while noting that geolog’s own identity story is complete only at the envelope layer (the realm-attribution gap above). **[judgment]**

4.3 The derivation engine: fixpoints and incrementality

geolog ships its central semantic object. The restricted chase runs today with modes, congruence closure, typed budgets (volume trips with named diagnostics; `tests/budget.rs`), weak-acyclicity analysis exercised in tests (`tests/weak_acyclicity.rs`), and per-commit validity judgments; an incremental chase driver has landed at the presence level, with its differential partner being naive recomputation, and weighted incrementality named as future work with the same differential shape **[observed]** (`geolog-chase` README and test roster). Deletion is handled *outside* the monotone fragment, as explicit machinery rather than a pretense that retraction is monotone: derivations carry weights in a choosable ordered-semiring algebra, tombstones trigger reweighting, and the death-at-zero result is protested equivalent to a from-scratch chase on the surviving facts **[observed]** (`geolog-weights`, chase differential tests). The live view is genuinely non-monotone under retraction; what the machinery buys is a *tested maintenance story* for it, with the append-only log and the monotone entailment core left undisturbed.

Coln has not yet shipped a fixpoint. The store’s solver is a matcher that validates geometric sequents against replayed tables **[observed]** (`solver/`); the strategy note concedes the system “does not yet support computing a least fixed point” **[observed]** (`why.tex`), and nothing landed since changes that. Instead, two query engines are under construction in parallel: `coln-query`, which wraps the DBSP crate (Feldera’s engine) with one level of nested circuit — DBSP’s mechanism for incremental recursion, a circuit within a circuit — for eventual recursion **[observed]** (`Cargo.toml: dbsp = "0.305.0"`; wrapper sources), and `coln-batch`, a new batch engine — column-major *u64* relations, Arrow IPC, a `SortedTable` storage contract, and *two* executors (classic binary-join chains and a worst-case-optimal generic join) differential-tested against a brute-force oracle and against each other, with semi-naive evaluation to fixpoint named as the next step **[observed]** (`coln-batch` README; landed 2026-07-10 through 2026-07-28).

Steelman. Outsourcing incrementality to DBSP is a rational bet **[judgment]**: DBSP’s Z-set circuit theory was built for incremental view maintenance (its production maturity is asserted by neither project and not audited here), and Coln’s planned use — constraints compiled to incrementally maintained *violation views*, where an empty view is validity and deltas are counterexamples — is a genuinely elegant unification of checking with querying **[observed]** (`coln-query api/violations.rs`). The batch engine, meanwhile, is exactly what an egglog-flavored proof-search workload wants (its test fixtures are program-analysis Datalog and e-matching patterns **[observed]**), and building it against a brute-force oracle from day one is the right discipline.

Judgment. At the pins, geolog’s chase computes its advertised fixpoint, while Coln’s strategy note and batch README still place least-fixpoint execution in future work; Coln’s two-engine parallelism also suggests the engine question is genuinely unresolved internally **[inferred]** (two overlapping conjunctive-query engines with disjoint substrates, both moving in the survey window). The mature-substrate bet may well close the gap quickly — DBSP plus semi-naive recursion is a short road for a team that has already wrapped the crate — and when it does, the interesting comparison becomes incrementality *disciplines*. geolog’s surveyed tree documents incremental-vs-rebuild arbitration in CI with divergence release-blocking; Coln’s surveyed tree does not state an equivalent end-to-end check — though nothing prevents wrapping DBSP in one, so this is a delta between documented practices at the pins, not between what the architectures permit. **[judgment]** Two Coln artifacts are worth importing regardless: the worst-case-optimal join as a conformance-gated execution lane, and violation views as the scalable form of geolog’s reported judgments (§6, R1–R2).

4.4 Distribution and sync

Coln inherits the Automerge lineage: a green end-to-end websocket round-trip lives in-repo, storing commit chunks as Sedimentree loose commits, with a store’s sync identity being its root-commit hash; a browser demo wraps a Coln store as an automerge-repo custom document type, and the store now compiles to wasm behind typed TypeScript handles **[observed]** (`sync-demo`; `coln-js-runtime`, landed in the survey window). The unit of sync is the whole store.

geolog ships two layers in one crate: a deliberately dumb whole-file N-device mesh over single-writer append-only logs (replay-through-ingestion), and a connected frontier-exchange protocol over a sans-io engine with TCP and libp2p transport bindings, where *realms* — partition keys carried in the wire and

the store layout, not in the signed skeleton — scope interest and placement **[observed]** (geolog-sync README).

Steelman. This is Kleppmann’s home ground, and reuse is a defensible instinct **[judgment]**: Sedimentree and Subduction carry the Automerge lineage’s accumulated local-first design experience, and Coln’s sync is already user-visible (a browser demo a reviewer can click) where geolog’s is engine-visible.

Judgment. Maturity is roughly at parity — both are early — but the risk profiles differ, and honestly stating them requires both projects’ own caveats. Coln’s unit of sync is the whole store, and its transport carries unsigned operations whose trust derives from the channel. geolog’s entries are self-authenticating in content and authorship, so those never depend on the channel — but realm attribution is not yet in the signed skeleton and the current transport profiles assume a trusted deployment boundary (§4.2), so geolog’s transport-independence today covers what an entry *says*, not yet where it *belongs*. On convergence: geolog’s tree demonstrates set-union convergence among per-writer nodes (protested against a reference union model, with equivocation branches admitted) under its stated fair-recurrence and trusted-realm assumptions **[observed]** (geolog-sync README); the un-demonstrated remainder is the intended deployment shape — untrusted boxes with authenticated realm attribution. Coln’s in-repo round-trip demonstrates store-granularity replication over a websocket; the surveyed artifacts do not yet demonstrate authenticated multi-writer convergence at store grain, and the identity layer and derived-view semantics remain open beside it. Re-survey this dimension when either project lands its missing half. **[judgment]**

4.5 Proofs, certificates, and what “trust” means

This is the largest structural asymmetry, and it is two-directional in an easily missed way: the projects mean different things by “proof.”

Coln’s proofs are *intra-database*: derivations in user theories, in Coln’s own logic, with proof search as the long-run ambition (“is there a proof of X?” answered by database technology — the egglog-module-system reading) **[observed]** (002H). Trust is vested in the core calculus and elaborator; the strategy note gestures at succinct cryptographic proof compression later, and no mechanized-soundness or verified-checker artifact exists in the repository **[observed]** (survey of the pinned tree; the survey found no such artifact — an absence claim, checkable by grep).

geolog’s certificates are *about external systems*: exact-rational polynomial bounds, ranking arguments, invariant towers for claims of the form *this dynamical system satisfies this temporal specification*. Its architectural position is that such certificates are ordinary rows in the meta-schema — no separate certificate file format exists — and that the engine computing with them is deliberately *not* the trust anchor: the same bytes are re-checkable by independent consumers, among them a certified binary (HOL4 soundness end-to-end through verified compilation to machine code; a demonstrated route, assembling now, with decode totality of the wire format already proven as theorems) and a Lean reflection route maintained partly to *measure* whether a general-purpose kernel can consume machine-scale certificates **[observed]** (README “Trust and auditability”; the companion `qts1` repository and book, which carry the precise statements of what each checker’s theorem binds today). Decision procedures enter the engine through a *ground seam*: applications declare ground vocabulary as theory rows and supply evaluation as a provider; an engine-owned binder adjudicates claims against declarations, refuses over guessing, and defers unbound entries so that plans citing them cannot compile **[observed]** (geolog-ground-api; README).

Steelman. For Coln’s sense of “proof,” an external certified checker is a less natural demand **[judgment]**: the classical trust answer for a proof assistant is a small core, Coln’s calculus is deliberately cut down relative to full dependent type theories, and its proofs-as-content will be *re-derivable* by any conforming implementation — re-derivability is a real assurance primitive that both projects share. Coln is not missing geolog’s certificate machinery so much as not aiming at geolog’s problem.

Judgment. With that granted: for certification workloads — claims about external systems, produced by powerful untrusted search, consumed by parties who did not run the producer — geolog has the directly applicable architecture and is assembling an artifact-backed answer to “why should I believe an ACCEPT?” (refusal disciplines and engineering instruments in the surveyed tree; the mechanized components in the companion repository, at the status its own surfaces state — again, no finished end-to-end verified checker is claimed). Coln does not currently claim to address the external-checker trust boundary at all, so it is not scored against that goal here. Conversely, geolog does not aim to be a *proof-search* engine for user theories; if that capability matters, Coln is the only one of the two building toward it. These are complementary ambitions sharing one substrate idea, and the certificate/search split is the cleanest statement of what the two projects are each for. **[judgment]**

4.6 Performance discipline

geolog states a lane law — an acceleration lane may change cost, never verdicts — and enforces it with an always-on exact reference floor (arbitrary-precision rationals; no ungated approximate path exists), differential batteries on every lane, and measured dispatch receipts; multi-architecture fast paths live in a companion repository under one conformance discipline **[observed]** (README; the **kernels** companion).

Coln’s new batch engine arrives with a recognizably convergent discipline: the **SortedTable** trait’s rustdoc is declared the contract, a **check_contract** conformance checker is shipped for future storage backends to run against their own indexes, both executors are differential-tested against a brute-force oracle, performance work is deliberately deferred and marked, and ten-million-row timed runs are in the test suite **[observed]** (**coln-batch** README). DBSP supplies the other engine an incremental substrate built for the purpose. Today the batch engine is *u64*-only and single-substrate **[observed]**.

Judgment. Convergent evolution toward the same discipline, with **geolog** covering more surface and coupled to exactness (the reference floor is exact rational arithmetic, which is what certificate semantics require). **Coln**’s batch engine introduced a conformance checker and a differential oracle in the survey window; **check_contract** — the storage contract shipped *with* its conformance checker — is the same move as **geolog**’s lane batteries. Operational maturity on either side is a separate question the surveyed artifacts do not settle. **[judgment]**

4.7 Test and verification culture

Both projects are serious; the signatures differ. **Coln**’s battery is elaborator-centric: golden tests (270 files added in the survey window, including proof-irrelevance and parameterized-theory batteries), randomized differential tests over generated query shapes, a brute-force oracle, license/REUSE compliance checks, and a modernized runner **[observed]**. **geolog**’s battery is invariant-centric: every engine-critical component names a differential partner (an independent implementation or re-derivation path — an external Datalog engine for the chase; from-scratch rebuild for the store; a stock streaming engine over the integers for the weight algebra) and “a component without a partner is not done, it is unverified” **[observed]** (**ENGINEERING.md** #3); plus rebuild invariance in CI, drift-alarmed golden artifacts regenerated by their generators, and refusal batteries for the loud failure modes. Near-parity of seriousness; **geolog**’s culture covers more of the stack (storage, sync, weights, normalization), **Coln**’s goldens cover its language deeper than **geolog** covers its own surface grammar — each project would benefit from the other’s emphasis. **[judgment]**

4.8 Developer surface and documentation

Coln is clearly ahead on interactive surface **[judgment]**, and the gap widened in the survey window: a language server and a VS Code extension; a REPL with transactions and session persistence; generated TypeScript interfaces per theory (with the API mirroring the semantics — freshness is a nullary **add()**, views forward into parent tables); a structured-diagnostics subsystem with terminal and HTML renderers; the store compiled to wasm behind typed handles; and two browser demos deployed **[observed]** (**coln-cli** verbs; **coln-ls**; **diagnostician***; **coln-js-runtime**; deploy commits in the window). The manual is a Forester hyperbook whose glossary is a Datalog query executed over the manual’s own link graph **[observed]** — the project’s idea applied to its own documentation.

geolog ships a carryable CLI (**check** / **chase** / **rows** / **normalize** / **load** / **fmt**; one obvious verb, options after the verb), self-contained theory files, and a book whose **Geolog** part walks the committed examples at reading pace and whose experimental numbers re-derive mechanically from committed artifacts **[observed]** (README; the companion book, whose experiment tables regenerate from a make-target-per-section pipeline over committed fixtures).

Judgment. Different audiences today — **Coln** courts language users, **geolog** courts engine consumers and auditors — but the direction of required motion is not symmetric: **geolog** will need **Coln**-class interactive tooling to be adoptable as a database, while **Coln**’s documentation-and-reproducibility posture is already adequate to its stage. The replicable pattern is concrete (§6, R3–R4): elaborate to IR, generate typed bindings, put the store behind wasm handles — proven next door, and **geolog**’s meta-schema (theories already queryable as rows) makes the codegen half unusually cheap. **[judgment]**

5 Coln’s current direction: the survey window at the receipts

The survey window’s landings (§1: the 80-commit ancestry delta, last landing 2026-07-28) support a specific characterization of where **Coln** is heading. The window is short — eleven days of landings — and sees only

what reached the public default branch; work on private branches or elsewhere is invisible to it. Observations first, readings marked:

- **Zero foundations motion.** The manual (69 trees) and the design-note corpus have a combined empty diffstat over the window **[observed]**; every landed commit is implementation or engineering hygiene. Reading: the foundations layer is resting, at least publicly, while implementation catches up. **[inferred]**
- **The language core hardened.** All 270 files added to the compiler in the window are golden tests **[observed]** (added-file listing); the elaborator’s semantic corners (proof irrelevance, models-as-parameters, nested theories) are being pinned down ahead of any new foundations writing.
- **The query layer is being built twice.** Tuple support landed in the DBSP engine; simultaneously an entirely new batch engine with worst-case-optimal joins appeared with a new contributor, mirroring the compiler IR’s rule shape and naming semi-naive recursion as its next step **[observed]**. Reading: the incremental-vs-batch engine decision is live and unresolved; the two engines share no substrate today. **[inferred]**
- **The browser became a first-class target.** The store went behind wasm bindings with a typed TS runtime; both demos deployed **[observed]**.
- **Minimal storage/sync/authorization motion.** One store file touched in the window (pin-to-pin diffstat by package); author fields remain all-zero placeholders; no Keyhive integration has begun in-repo **[observed]**. Reading: parked for the window, consistent with the strategy note’s reuse sequencing. **[inferred]**
- **Reviewer-readiness work.** Test-runner modernization, per-package justfiles, a README rewritten explicitly for reviewers, license-compliance gates **[observed]**.

Compressed **[judgment]**: the window shows Coln consolidating into a demonstrable language-plus-engines story — an elaborator you can trust, engines intended to run its IR efficiently, demos a reviewer can click — with the landed sync slice riding the inherited stack and authentication/access control still deferred to the planned Keyhive integration. That is consistent with its thesis (§2). The window reinforces Coln’s stated emphasis on elaboration, query execution, and browser access; it contains no evidence of motion toward geolog’s external-checker problem — which is not Coln’s problem — and geolog’s recent motion contains none toward proof search or surface-language expressiveness, which are not geolog’s. The two projects are, at present, deepening different halves of the shared design space.

6 What geolog should consider replicating

Candidate transfers, each with its Coln-side premise cited and its cost stated as an estimate with the dominant unknowns named. The recommendations themselves are **[judgment]** as a class; the prices are estimates a design pass would have to confirm, not analyses performed here.

- R1. A worst-case-optimal join lane.** Premise: Coln-batch’s generic-join executor, differential-tested against both a naive oracle and a binary-join twin **[observed]** (`coln-batch` README). This is the shape geolog’s lane law wants — an acceleration lane whose battery is its sibling executor — and it should enter as a *conformance-gated experimental lane*: worst-case optimality protects against pathological multiway intermediate results, and is not a universal speedup over binary plans, so dispatch needs workload evidence. Real costs beyond the executor: suitable sorted/trie index access shapes and planner dispatch. Estimate: small-to-moderate, contingent on index shapes; validation criterion: the lane’s battery plus a measured win on at least one real workload before default dispatch.
- R2. Violation sets as lazily enumerable views.** Premise: Coln’s design compiles each constraint to an incrementally maintained *view of its counterexamples* (empty = valid; deltas = new violations; the failing set is a query) **[observed]** (`coln-query api/violations.rs`). geolog’s `report-mode` judgments are today a stream surfaced at chase time; the view form is attractive as monitored theories grow, but the tradeoff is workload-conditional — maintaining a large counterexample set can cost more than streaming it. Estimate: small at the reporting layer, riding the incremental driver; `report-mode` semantics unchanged; adopt where violation sets are queried repeatedly, keep the stream where they are not.

- R3. Typed bindings generated from the meta-schema.** Premise: Coln generates per-theory TypeScript interfaces whose shape mirrors the semantics [**observed**] (coln README’s `generate-ts`; manual API trees). geolog’s meta-schema makes the *signature* layer cheap — theories are already rows — but naming, versioning, recursion, language mapping, and compatibility policy are the real cost of a binding surface, and they arrive with the first external consumer. Estimate: moderate, additive, staged (bindings, then REPL, then language server), behind an explicitly unstable flag until the engine wants API-stability pressure.
- R4. A wasm story for the store.** Premise: Coln’s `StoreHandle/TransactionHandle` wasm cut with a typed TS layer [**observed**] (coln-js-runtime sources). geolog is pure Rust and its sync engine is sans-io, which removes one obstacle; storage backends, threading, entropy/crypto availability, and filesystem APIs in the browser are the costs that remain and can dominate. Estimate: moderate; a browser-resident geolog instance unlocks demos and local-first applications a CLI cannot, which is also Coln’s demonstrated payoff.
- R5. The modal reading of parameterized-theory scoping.** Premise: Coln’s `@ind` modality (locked variables admitted in domains, never codomains) and the inductive-theory bit that gates `init` [**observed**] (manual 001R/001S; `initial-models.tex`) — a principled, literature-anchored account of scoping rules geolog’s loader and normalizer currently enforce operationally. Adopting the *reading* is a design document plus loader-check tightening; adopting it *honestly* may also mean discharging the semantic obligations the modal account makes explicit, which is real design work, not paperwork. Estimate: small for the discipline, open-ended for the semantics; the cheap half is worth it alone.
- R6. Self-consuming documentation.** Premise: Coln’s manual glossary is a Datalog query executed over the manual’s own link graph [**observed**] (manual practice; the glossary tree’s `\execute` block). A small thing that argues the thesis on every page. When geolog’s book reaches reference-manual density, generating its indexes from a geolog instance of the book’s own cross-reference theory would be the same argument. Estimate: small; purely additive; lowest priority here.

Not recommended at present, with reasons stated so they can be contested: Coln’s surface-syntax philosophy (one syntactic class; bicameral parsing — a permissive tree-building read followed by a stricter grammar pass) is language-user ergonomics geolog does not yet need while its users are engine consumers. SNARK-compressed witnesses target proof-size and verification succinctness, which geolog’s verified decoding and sharded checking do not supply — those address parser soundness and horizontal scale respectively; geolog’s reasons for not pursuing succinctness cryptographically (setup and trust assumptions, prover cost) are a separate analysis this report does not perform, so this is a genuine open difference in direction, not an equivalence. And DBSP-as-engine would replace the component geolog treats as its thesis (an engine whose incremental state is continuously arbitrated against from-scratch re-derivation); the transferable part is the violation-view design (R2), though DBSP behind geolog’s own differential contract is a coherent architecture someone could argue for.

7 What geolog holds that Coln does not

Each with the honest reason Coln lacks it — in every case a defensible sequencing choice, not an oversight; the question a reader should ask is whether the deferral compounds. All [**judgment**] on the advantage reading; the underlying facts are [**observed**] above.

1. **Signed, self-authenticating commits and local equivocation detection, now** (§4.2, with its stated realm-attribution limitation). Coln defers to Keyhive; the deferral compounds to the degree merge semantics and provenance come to assume an identity notion that signing must later respect.
2. **A shipped fixpoint with a termination discipline** (§4.3): the chase with congruence, budgets-with-typed- refusals, weak-acyclicity analysis, and validity judgments, against Coln’s not-yet-computable `init`. Deferral compounds only until their engine choice lands; this is the advantage most likely to close, and its closure would be a gain for the design space both projects draw on.
3. **In-band schema and migration representation** (§4.1): schemas, extensions, and migrations as rows under the same log discipline as data, with revision machinery in the workspace, against schema-in-root-commit with language-level migration views. The end-to-end live-evolution demonstration is still owed on both sides; the positional claim is that geolog’s architecture reaches it without a storage-layer redesign.

4. **A tested retraction story beside the monotone engine** (§4.3): the ordered-semiring weight algebra with tombstone reweighting and proptested death-at-zero equivalence — retraction handled as explicit machinery outside the monotone fragment rather than left open. Coln’s merge notes leave retraction as an open question; nothing in the pinned tree answers it.
5. **Typed escape hatches with refusal semantics** (§4.1, §4.5): aggregation nodes (whole-fibre folds that are defined, never asserted), the ground seam (external decision procedures behind an adjudicating binder where unbound vocabulary cannot compile), and the tensor/einsum surface. Coln will meet the same needs at the language level eventually; today it has no counterpart for any of the three.
6. **The external-checker position** (§4.5): certificates as ordinary rows, re-checkable by independent consumers — a certified binary being assembled and a Lean route maintained for measurement; no finished end-to-end verified checker claimed. Not Coln’s problem today — and none of the upstream components named in Coln’s surveyed reuse plan supplies an external-checker boundary, so reaching this part of the shared design space would require a component the inheritance does not provide.
7. **Breadth of invariant culture** (§4.7): differential partners as a completion criterion across storage, sync, weights, normalization, and chase; rebuild invariance in CI. Coln is converging on this crate by crate; breadth takes time.

8 Summary

dimension	Coln at 654a184	geolog-alpha at b4c7fa2
thesis	language-first data-oriented proof assistant; storage/sync/access control drawn from the Automerge lineage (sync round-trip landed; authorization planned)	log-first entailment database; assurance-first, untrusted tier of an external-checker architecture
foundation	two-level dependent TT, @ind modality, init, QIIT signatures; elaborator hardening	geometric fragment + self-hosting meta-schema; typed machinery for folds, oracles, tensors
identity & trust	content-addressed, unsigned, all-zero placeholder author fields; Keyhive planned	ed25519 self-authenticating envelopes; local equivocation detection; realm attribution not yet signed; capability algebra
fixpoint engine	not yet shipped; DBSP wrapper + new batch WCOJ engine in parallel	restricted chase shipped: modes, congruence, budgets, weak-acyclicity, incremental (presence level)
retraction	open question in merge notes	semiring weights, tombstone reweighting, death-at-zero proptested (explicit machinery outside the monotone fragment)
sync	Sedimentree/Subduction round-trip; store = document; browser demo	whole-file mesh v0 + frontier-exchange v1 (TCP, libp2p); realm partitioning; per-writer union convergence
proofs	intra-database derivations; proof search ambition; trust in core calculus; SNARKs aspirational	proptested certificates about external systems as ordinary rows; independent-checker architecture (certified binary assembling; none finished)
performance	DBSP substrate; new conformance-checked storage contract; u64-only batch engine	lane law, exact reference floor, conformance batteries, multi-arch kernels
developer surface	LSP, REPL, TS codegen, wasm, deployed demos, hyperbook manual	carryable CLI, self-contained theories, book with mechanical reproduction
direction (the 11-day landing window)	language core + engines + browser; zero foundations motion; storage/auth minimal	(out of scope here; see the companion book and repositories)

The two projects took one brief and factored it oppositely, and the factorizations are close to disjoint: nearly everything Coln has built deepest (elaborated surface language, interactive tooling, the inherited-distribution strategy) is what geolog has deferred, and nearly everything geolog has built deepest (signed

identity, shipped entailment, the certificate substrate, the external-checker architecture) is what Coln has deferred. The shared choices of §3 mark the spine both consider settled. Each project is, at minimum, an existence proof for the other that its own deferred half is buildable. §6 and §7 record candidate transfers and current deltas respectively; neither list is a permanent position, and both should be re-read against the re-survey triggers, stated so this document ages honestly: Coln landing either a fixpoint engine or Keyhive integration; geolog landing its interactive surface or signed realm attribution; either project demonstrating its intended deployment grain for multi-writer operation.

Source appendix. The following pages carry the complete L^AT_EX source of this document (`coln-delta-survey.tex`), one source file per page, as an invisible text layer: extract it with `pdftotext -layout <thisfile>.pdf -` (the source begins at the first `%` comment or `\documentclass` on the next page; the layer never prints or displays). That recovers every line exactly, except that interior runs of exactly two spaces collapse to one; `pdftotext -layout -fixed 1.3226 <thisfile>.pdf -` is byte-exact throughout. The same files are also embedded byte-exactly as PDF attachments: `pdfdetach -saveall <thisfile>.pdf`, or `reader.attachments` in `pypdf`.

