Provenance as First-Class Execution Semantics: A Profile-Relative Contract for Versioned Derivations

Matthew Long
The YonedaAI Collaboration
YonedaAI Research Collective
Chicago, IL
matthew@yonedaai.com · https://yonedaai.com

23 July 2026

Provisional status: OBSTRUCTED

Abstract

A row can carry the name of its source and still conceal almost everything needed to explain it. The name does not say which snapshot was read, which records contributed, which mapping transformed them, why conflicting identities were reconciled, which plan actually ran, or whether retention has already removed part of the evidence. Calling all of these facts "provenance" does not make them interchangeable.

This paper proposes a profile-relative execution contract for provenance in CatDB, a research architecture for versioned semantic integration. The contract keeps five layers distinct: source identity, relational derivation, versioned semantic artifacts, retrospective execution, and authority or reconciliation decisions. For a finite positive bag-relational fragment, it reuses established semiring provenance and records derivations as shared circuits. It then connects those circuits to immutable schema, mapping, migration, query, plan, materialization, snapshot, and decision references. Every explanation reports its model, granularity, retained evidence, and one of five completeness classes. A W3C PROV projection supplies interchange without pretending that an entity-activity graph preserves polynomial structure or value origin.

The proposal includes normalization and identity rules, a query surface, counterexamples, privacy and retention behavior, falsifiers, systems baselines, and separate implementation, experiment, and formalization gates. The possible contribution is the typed composition boundary across database, workflow, version, and decision evidence. Why-, where-, lineage-, semiring-, workflow-, storage-, and security-oriented provenance are prior art.

This is not a completed systems result. The repository has no frozen provenance profile, provenance fixtures, `catdb-provenance` runtime, independent provenance oracle, provenance theorem, exporter, or benchmark run. The prerequisite query and mapping-data semantics are not frozen. Accordingly, all CatDB execution, completeness, performance, security, and formal claims in this manuscript are OPEN or OBSTRUCTED.

Contents

1 Introduction

1.1 One result, several different questions

Suppose an analyst asks for active customers and their current support tiers. The customer records came from two PostgreSQL snapshots, a CSV correction, and a service response. A mapping renamed fields and converted units. A reconciliation rule linked two identifiers, while a reviewer rejected a third candidate. A materialized result was refreshed incrementally. The planner pushed one selection to PostgreSQL and evaluated a residual predicate locally.

The analyst may ask:

1. Which source records contributed to this row?
2. Which location supplied the displayed tier value?
3. Which alternative derivations made the row appear?
4. Which schema, mapping, query, and plan revisions governed the run?
5. Which identity decision joined these records?
6. Would a changed mapping invalidate the result?
7. Is the answer complete, or did an opaque call, retention rule, source failure, or access policy hide part of it?

These are not variants of one graph traversal. Why- and where-provenance are different models [9]. Backward lineage for a warehouse view has requirements that depend on the view definition and retained auxiliary data [14]. Positive relational how-provenance has an established semiring semantics [17]. Prospective and retrospective workflow provenance distinguish the planned recipe from the observed run [15]. W3C PROV supplies interoperable entities, activities, agents, and relations [24, 26]. A source label, polynomial, workflow trace, and PROV graph therefore answer overlapping but nonidentical questions.

The design error addressed here is not a lack of provenance terminology. It is the loss of these distinctions at execution boundaries. A connector emits a row identifier without its snapshot. A mapping records an operation name without the immutable mapping artifact. A planner records a physical stage without the semantic query it was meant to implement. A reconciliation score is copied into a result as though it established truth. A retained lineage set is described as a complete explanation even after field origin was discarded.

1.2 Research question and bounded answer

The research question is:

> How should provenance compose across mappings, migrations, reconciliation, materialization, and queries while preserving explicit evidence and completeness boundaries?

The proposed answer is deliberately finite:

> For a versioned, explicitly delimited transform fragment, record typed source identity, derivation, execution, artifact, and decision evidence. Compose only the admitted derivations. Every answer states the provenance model, granularity, retained evidence, and completeness class.

Evidence status. ENGINEERING HYPOTHESIS. The contract is specified here but is not implemented..

1.3 Contributions and prior-art boundary

This paper contributes a provisional contract, not a new general provenance algebra:

1. a question-to-representation table that prevents a lineage set, value origin, derivation circuit, activity graph, or trust decision from impersonating another model;
2. a positive bag-relational composition profile that reuses $\mathbb{N}[X]$ annotations and makes unsupported operators explicit;
3. a typed envelope linking derivations to immutable artifact versions, snapshots, plans, materializations, and decisions;
4. normalization, completeness, retention, disclosure, and W3C PROV projection rules;
5. counterexamples and falsifiers for overclaiming source tags, completeness, rewrite preservation, incremental equivalence, and trust; and
6. executable and formal roadmaps with separate acceptance gates.

Source tagging for heterogeneous integration is decades old [30]. Uncertainty and lineage were integrated in ULDBs and Trio [1, 6]. Database-style and workflow provenance have been composed [2]. Provenance-aware storage, database middleware, PostgreSQL extensions, Spark dataflow tracing, and operator-integrated lineage are established systems directions [5, 21, 27–29]. The novelty question is whether a typed, versioned bridge across these evidence layers produces useful and affordable explanations in the stated CatDB fragment. That remains open.

2 Evidence and dependency status

2.1 What exists

The repository contains a research manifest, claim register, prior-art map, implementation and formalization roadmaps, benchmark artifact contract, structural Rust instance envelopes, and smaller Rust, Haskell, and Lean work on schemas, paths, and mappings. These artifacts constrain the proposal. They do not execute provenance.

P13 depends on two upstream papers. P1 must define immutable schema and mapping identities. P4 must define the instance-level mapping and data-transformation semantics whose provenance is to compose. Logging an underspecified operation name cannot repair either dependency.

2.2 What is absent

No current repository artifact supplies:

- a frozen query or provenance profile;
- source-record and snapshot identity rules accepted by review;
- provenance fixtures or a fixture manifest;
- a `catdb-provenance` crate or provenance query surface;
- an independent Haskell oracle;
- a Lean provenance representation or theorem;
- mapping, migration, reconciliation, connector, or materialization capture hooks;
- a W3C PROV exporter with a projection-loss audit; or
- correctness, overhead, security, or user-study runs.

The implementation-roadmap obligation `SYS-PROV-01` is OPEN: every admitted result must carry the exact version, snapshot, and record lineage required by profile v1, composing through selection, projection, and join. The `SYS-MAT-01` full-versus-incremental data and provenance equivalence obligation is also OPEN. The query/provenance semantic gate is OBSTRUCTED until its profiles and fixtures are reviewed.

3 Models before records

3.1 Source tokens

Definition 3.1 (Source token)**.** A source token is a typed reference

$$x = (n, s, r, g)$$

Table 1: Current evidence status. "Literature" supports only the cited external result within its assumptions.

Layer	Status	Current reason
why/where/how distinctions	SUPPORTED BY PRIOR LITERATURE	Primary sources define the models
typed CatDB envelope	OPEN	Manuscript proposal only
query and mapping semantics	OBSTRUCTED	Upstream profiles not frozen
Rust runtime	OPEN	Owning crate absent
independent oracle	OPEN	Haskell module absent
formal laws	OPEN	Lean representation absent
materialization equivalence	OPEN	Runtime and fixtures absent
PROV export	OPEN	Exporter and loss report absent
security and disclosure	OPEN	Policy semantics absent
experiments	OPEN	No P13 run bundle

where n is a source namespace, s is a source snapshot or cursor identity, r is a stable record identity within that snapshot, and g is a declared granularity such as record, field, byte range, or text span.

The token says what the capture boundary claims to have observed. It does not authenticate the source, prove that the record is true, or guarantee that the same external identifier denotes the same entity in a later snapshot.

Definition 3.2 (Granularity subsumption)**.** For granularities g_f, g_c, write $g_f \preceq g_c$ only when the profile publishes a total deterministic parent projection

$$\uparrow_{f \to c} : (n, s, r, g_f) \longrightarrow (n, s, r', g_c)$$

whose fine region is wholly contained in exactly one coarse region. A field-level or byte-range token can answer a record-level request exactly only when this projection is available, every retained fine token maps unambiguously, and the capture profile is complete for the requested question. The response returns the projection rule and both granularities. Missing coverage yields Partial; missing or ambiguous parentage yields Unknown. Coarse evidence is never silently refined into finer evidence.

Requirement 3.3 (Token collision resistance)**.** The profile must reject two observations that normalize to the same source token while denoting different bytes, records, fields, or spans under the declared source-identity contract.

Evidence status. OPEN. No source-token profile or collision fixture exists..

3.2 Lineage, why, where, and how

A lineage set lists contributing tokens. It is useful for backward and forward dependency questions, but it loses alternative derivations and multiplicity. Why-provenance identifies witnesses or contributing subdatabases. Where-provenance identifies source locations from which output values were copied. How-provenance distinguishes alternative and joint derivations algebraically [9, 17].

For example, if output tuple t can be derived either from source record x or from a join of y and z, the lineage set $\{x, y, z\}$ does not preserve the distinction

$$p_t = x + yz.$$

If two copies of x contribute under bag semantics, a set also loses the coefficient. Conversely, the polynomial does not tell which input field was copied into the displayed output field unless value origin is modeled.

Counterexample 3.4 (A lineage set is not an explanation circuit). Consider two databases whose output row has the same contributing token set $\{x, y, z\}$. In the first, the row follows $x + yz$; in the second, it follows $xy + xz$. Their lineage sets agree, but removing x has different effects. Any interface that returns the set and labels it an exact how-explanation is unsound.

3.3 Prospective and retrospective provenance

Prospective provenance records the intended recipe: immutable schema, mapping, migration, query, logical plan, physical plan, materialization definition, and policy artifacts. Retrospective provenance records what happened: source observations, operator activities, retries, partial results, decisions, outputs, failures, and publication events. Workflow research already makes this distinction [2, 15].

The two must be connected but not collapsed. A plan artifact can exist without a successful execution. An execution can depart from a plan through fallback or retry. Recording the plan alone does not establish that a connector executed its claimed snapshot. Recording a trace alone does not establish that its physical operators implemented the logical query.

3.4 Question-to-representation contract

Table 2: Each question binds to a model and an explicit insufficiency.

Question	Required representation	Insufficient substitute
Which records contributed?	record tokens plus lineage edges or circuit support	source name, workflow activity, or value origin alone
Why did this row appear?	declared why-model or derivation circuit	an unordered token set presented without model
Where did this value come from?	field/range token and value-origin rule	row lineage alone

Question	Required representation	Insufficient substitute
How was the row derived?	normalized additive/multiplicative circuit	PROV entity-activity graph alone
Which versions governed it?	immutable artifact references and source frontiers	a timestamp or mutable object name
Which plan ran?	logical and physical plan identities plus retrospective stage events	physical trace without logical binding
Why were identities joined?	decision artifact, evidence refs, authority, outcome, and revision	similarity score alone
What depends on this source?	forward dependency index with completeness metadata	a backward-only trace
Does it survive a schema change?	changed artifact plus revalidation of affected semantics	old provenance graph alone
Is this explanation complete?	profile, evidence retention, disclosure, source availability, and completeness class	success status alone

4 Positive relational derivations

4.1 Profile boundary

The first algebraic profile admits finite bags, scans, pure total selection, projection, union, and inner equijoin. Required attributes are non-null. Scalar equality and collation are fixed by the query profile. Results are unordered canonical multisets. Outer joins, null and three-valued logic, difference, general negation, aggregation, recursion, ordering, volatile functions, floating-point equality, and opaque external calls are unsupported until separately versioned.

This is intentionally smaller than SQL. Aggregation needs additional value provenance structure [3]. Difference has known limits for uniform semiring extensions [4]. Provenance games offer a different approach to negation and why-not questions [23]. Datalog circuits address recursion with their own representation and complexity choices [16]. None may be imported by implication.

4.2 Annotations

Let X be the finite set of source tokens observed for a run. Each input tuple is annotated by a generator in $\mathbb{N}[X]$. For an annotated relation R, write $R(t)$ for the annotation of tuple t, with absent tuples mapped to 0. Addition records alternative derivations and multiplication

records joint use. Coefficients retain bag multiplicity.

Definition 4.1 (Profile-v1 denotation)**.** For the admitted fragment, the annotation semantics are:

$$[\![\mathrm{scan}(R)]\!](t) = R(t), \tag{1}$$

$$[\![\sigma_\phi Q]\!](t) = \begin{cases} [\![Q]\!](t) & \text{if } \phi(t), \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

$$[\![\pi_A Q]\!](u) = \sum_{\pi_A(t)=u} [\![Q]\!](t), \tag{3}$$

$$[\![Q_1 \cup Q_2]\!](t) = [\![Q_1]\!](t) + [\![Q_2]\!](t), \tag{4}$$

$$[\![Q_1 \bowtie_\theta Q_2]\!](u) = \sum_{\substack{t_1,t_2 \\ t_1 \bowtie t_2 = u \\ \theta(t_1,t_2)}} [\![Q_1]\!](t_1)[\![Q_2]\!](t_2). \tag{5}$$

Evidence status. SUPPORTED BY PRIOR LITERATURE as an algebraic basis [17]; OPEN as a CatDB profile and runtime..

4.3 Circuits rather than forced expansion

Canonical expanded polynomials can grow sharply. A proposed runtime stores a directed acyclic circuit with token, zero, one, addition, and multiplication nodes. Commutative children are sorted by child digest. Associative nodes are flattened where the profile permits. Neutral elements are removed and annihilation by zero is applied. Equal subcircuits share a content-addressed node. Expansion remains a query or test operation, not the storage default.

Circuits for provenance are established prior art [16]. The open CatDB issue is whether this normalization is deterministic across Rust and an independent oracle, retains the profile's multiplicities, and stays operationally bounded.

Requirement 4.2 (Denotational normalization)**.** For every well-formed circuit c,

$$\mathsf{Eval}(\mathsf{Norm}(c)) = \mathsf{Eval}(c)$$

in every commutative semiring admitted by the profile.

Evidence status. OPEN. It needs definitions, cross-language fixtures, and selected machine-checked laws..

4.4 Typed substitution across mappings

Suppose a mapping F produces intermediate tokens y from source tokens X, and a query Q derives outputs from y. Composition substitutes each intermediate generator with its mapping derivation:

$$\mathsf{Prov}(Q \circ F)(t) = \mathsf{Prov}(Q)(t)\,[y \mapsto \mathsf{Prov}(F)(y)]\,.$$

Substitution is allowed only when source, schema, snapshot, mapping, and granularity types agree. A string replacement over identifiers is not composition.

Proposition 4.3 (Sequential/composed target)**.** *For admitted positive mappings F, G and input instance I, the intended law is*

$$\mathsf{Norm}(\mathsf{Prov}(G, F(I)) \circ \mathsf{Prov}(F, I)) = \mathsf{Norm}(\mathsf{Prov}(G \circ F, I)) .$$

Evidence status. OBSTRUCTED. The proposition is a target, not a theorem, because P4's instance-level mapping language and composition semantics are not frozen..

4.5 Rewrites

Equal visible rows do not guarantee equal provenance. A rewrite is enabled only if it preserves the declared result and provenance profile. For each enabled rewrite $Q \rightsquigarrow Q'$, the gate compares both normalized bags and normalized derivation circuits:

$$\mathsf{Norm}([\![Q]\!]_I) = \mathsf{Norm}([\![Q']\!]_I).$$

Counterexample 4.4 (Distinct erases evidence). Let a bag contain two equal visible rows with source annotations x and y. Projection followed by duplicate elimination can return one visible row. A row-only test sees agreement whether its provenance is $x + y$, $\{x, y\}$, x, or an opaque marker. The rewrite is not provenance preserving until duplicate semantics and the required model are fixed.

5 Composition across the execution stack

5.1 Source ingestion

A connector observation binds:

- connector and capability-manifest digests;
- source namespace and authenticated endpoint class;
- transaction, snapshot, cursor, or best-effort frontier;
- source record and optional field or range identity;
- pushed and residual operations;
- observed bytes or a digest when retention permits;
- timing, retry, partial-availability, and failure facts; and
- an exactness class justified by the connector contract.

A source may expose only page offsets, unstable row numbers, or an opaque service response. Such observations cannot be upgraded to stable exact record identity. They remain Opaque, Partial, or Unknown.

Counterexample 5.1 (Timestamp is not a snapshot). Two requests begin at the same wall-clock time but read different committed states. If the envelope identifies both by timestamp alone, later explanation cannot distinguish them. The timestamp may remain an observation, but it cannot serve as the snapshot identity.

5.2 Schema mapping and migration

A mapping activity references immutable source and target schemas, the mapping artifact, discharged and deferred obligations, the admitted instance-level operator, and the input and output derivations. A migration adds direction, unit, and materialization identity. If the transformation synthesizes a target object, the profile must say whether the token denotes the new object, its generating derivation, or both.

Category-theoretic structure may type mappings and compose transformations. It does not eliminate capture, storage, identity, or query costs. A mapping between schemas also does not uniquely determine how a particular runtime transformed instances. Prospective artifact provenance must be paired with retrospective execution evidence.

Requirement 5.2 (No semantic event by name alone)**.** An event claiming a mapping or migration must carry the immutable artifact identity and a profile-supported operator or an explicit opaque boundary. A free-form label such as `"customer-map-v2"` is not sufficient.

5.3 Reconciliation

Reconciliation provenance records evidence and authority without turning either into truth. A candidate decision contains:

$$\begin{aligned} d = (&\text{decision revision}, \text{candidate identities}, \text{evidence refs},\\ &\text{rule/model revision}, \text{score with semantics}, \text{authority},\\ &\text{outcome}, \text{approvals}, \text{reversal link}, \text{retention class}). \end{aligned}$$

Possible outcomes include accept, reject, defer, require approval, split, and revoke. Similarity is evidence for a decision, not semantic identity. Changing or reversing a decision invalidates downstream results according to the dependency graph.

Counterexample 5.3 (Confidence is not identity). Two records have vector similarity 0.99. A rule accepts them today and is reversed tomorrow after an authoritative identifier conflicts. If the provenance stores only the score, it cannot explain who or what authorized the old merge, identify affected results, or distinguish evidence from the decision. The result envelope was incomplete for the stated question.

Evidence status. OBSTRUCTED. P7 decision and authority semantics are not available..

5.4 Planning and execution

The logical query identity, normalized logical plan, physical plan, connector capabilities, pushed operators, residual operators, and actual stages are separate records. The planner

emits a decision certificate. Connectors emit source observations. The executor emits activities, retries, fallback, partial results, and outputs. A later provenance query can then distinguish:

- what the request meant;
- what the planner intended;
- what each connector claimed to execute;
- what the local engine observed; and
- what result artifact was published.

An execution trace does not prove semantic plan correctness. That relation requires query-profile equivalence fixtures and backend differential tests.

5.5 Materialization and incremental refresh

A materialization provenance record binds its definition, dependency frontier, refresh mode, prior checkpoint, input deltas, output changes, derivation updates, cursor publication, and durable result identity. Incremental view maintenance is established prior art [8, 18]. The CatDB obligation is stronger than equal visible rows:

$$\mathsf{Norm}(\mathrm{Data}_{\mathrm{inc}}) = \mathsf{Norm}(\mathrm{Data}_{\mathrm{full}}) \quad \wedge \quad \mathsf{Norm}(\mathsf{Prov}_{\mathrm{inc}}) = \mathsf{Norm}(\mathsf{Prov}_{\mathrm{full}}).$$

Cursor advancement and the corresponding durable data and provenance effects must be atomic under the supported failure model.

Provenance sketches can support data skipping and can themselves be incrementally maintained [25]. A sketch is an overapproximation for its purpose, not exact derivation evidence. A result using a sketch must declare approximate granularity and must not receive ExactForProfile.

Evidence status. OPEN. No materialization implementation or equivalence run exists..

5.6 Workspace and distribution boundaries

Workspace operations contribute immutable base, overlay, commit, merge, approval, and publication identities. Distribution contributes delivery, ordering, retry, causal, replay, checkpoint, and stale-state events. Neither layer may invent a second provenance model. They emit typed events into the one versioned envelope owned by the provenance abstraction.

A finite execution graph can be acyclic even when the complete version history contains revisions, reversals, and references across time. Cycle rules are therefore profile-specific: derivation circuits may require a DAG; version and influence graphs need separate constraints.

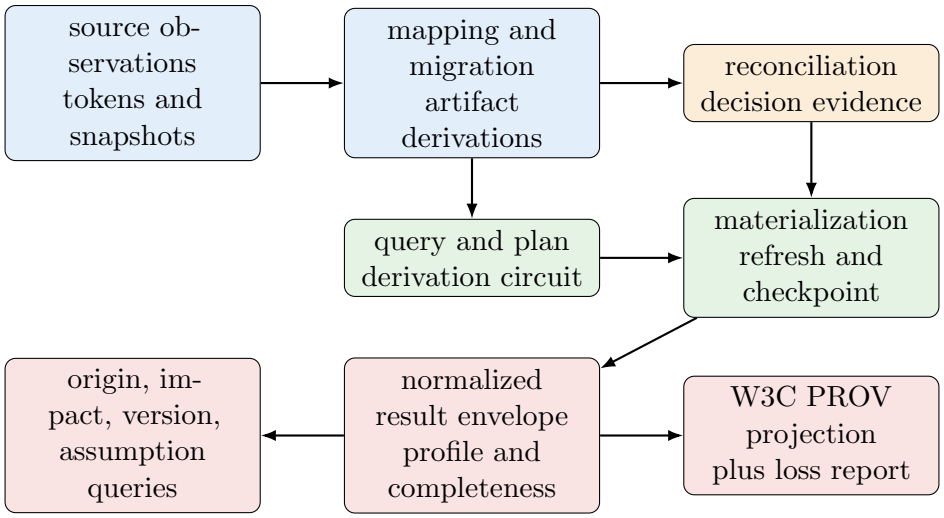

Figure 1: Proposed composition boundary. Arrows denote typed references and substitution obligations, not implemented dataflow.

6 Candidate typed profile

6.1 Stable references

Every durable reference carries a namespace, kind, version, digest algorithm, digest, and canonicalization profile. Mutable aliases may point to immutable objects, but provenance stores the resolved immutable identity used by the operation.

```
ArtifactRef {
  namespace,
  kind,
  version,
  canonicalization_profile,
  digest_algorithm,
  digest
}
```

Digest equality establishes equality of canonical bytes under the named algorithm and profile. It does not prove authority, authorship, semantic equivalence, or external authenticity.

6.2 Derivation and value origin

```
DerivationCircuit {
  profile_ref,
  source_token_namespace,
  root_node,
  nodes[],
  canonicalization_profile,
  completeness
}
```

```
ValueOrigin {
  output_record,
  output_field,
  rule: Copy | Constant | PureExpression | Opaque,
  input_ranges[],
  expression_ref,
  completeness
}
```

Profile v1 admits exact value origin only for declared copy, constant, and pure total expression rules. A unit conversion may list input fields and the versioned expression, but it is not where-provenance in the narrow copied-cell sense unless the selected model defines it that way.

6.3 Activity and decision records

```
ActivityEvidence {
  activity_id,
  prospective_artifact_refs[],
  logical_plan_ref,
  physical_plan_ref,
  used_entities[],
  generated_entities[],
  source_frontiers[],
  connector_observations[],
  started_at,
  ended_at,
  outcome,
  retry_parent,
  partiality_reasons[]
}

DecisionEvidence {
  decision_ref,
  candidates[],
  evidence_refs[],
  rule_or_model_ref,
  authority_ref,
  outcome,
  approval_refs[],
  reverses,
  uncertainty_semantics,
  completeness
}
```

Times help sequence observations. They do not replace version, transaction, snapshot, or causal identity.

6.4 Result envelope

```
ProvenanceEnvelope {
  envelope_profile_ref,
  result_artifact_ref,
  semantic_commit_ref,
  schema_refs[],
  mapping_refs[],
  migration_refs[],
  query_ref,
  logical_plan_ref,
  physical_plan_ref,
  source_frontiers[],
  derivation_circuit_ref,
  value_origin_refs[],
  activity_refs[],
  decision_refs[],
  materialization_ref,
  policy_ref,
  retention_ref,
  completeness,
  missing_evidence[],
  redaction_summary,
  created_at
}
```

This listing is a proposed field inventory. Exact types, optionality, cardinality, serialization, and compatibility rules belong in a reviewed profile and fixture schema.

7 Completeness is part of the answer

7.1 Five classes

Definition 7.1 (Completeness class)**.** Every provenance response has one of:

ExactForProfile Exact for the named profile, question, granularity, retained interval, source frontier, and disclosure principal.

Partial Some known evidence is absent because capture, source availability, or retention was incomplete.

Opaque An activity is known but its internal derivation is not represented.

Redacted Evidence exists or may exist but policy withholds it.

Unknown The system cannot establish which of the stronger classes applies.

ExactForProfile is never an unqualified global claim. A record-lineage answer may be exact for profile v1 and still provide no value origin. An authorized administrator and a tenant may receive different graph projections. A retention horizon may make an old answer partial. A failed source can make a federated result's provenance incomplete even if the returned rows are syntactically valid.

Requirement 7.2 (Monotone honesty)**.** Normalization, export, redaction, and query processing may preserve or weaken the completeness class. They may not strengthen it without new verified evidence.

Evidence status. PAPER REASONING contract; runtime enforcement is OPEN..

7.2 Evidence matrix

The answer carries machine-readable reasons:

Table 3: Examples of evidence that constrain completeness.

Condition	Maximum honest class	Required response behavior
stable source token, retained circuit, full admitted run	ExactForProfile for the named question/profile	list profile, frontier, granularity, and evidence digests
source records unavailable during partial federation	Partial	name missing source/frontier and affected result region
opaque API or model call	Opaque	list declared inputs, outputs, activity, and unsupported internal explanation
policy hides a contributing record	Redacted	avoid leaking protected identity through graph shape, counts, or timing
legacy result without capture manifest	Unknown	refuse exactness and state which capture facts cannot be established
retention deleted field ranges but kept row tokens	Partial for value origin; possibly exact for row lineage	classify per question, not once per result
sketch overapproximates contributing partitions	approximate Partial	report false-positive possibility and sketch revision

8 Normalization and identity

8.1 Canonical form

A shared artifact requires cross-implementation agreement on:

1. Unicode, scalar, identifier, and byte canonicalization;
2. object-field and map ordering;

3. set, bag, and sequence distinctions;
4. number and timestamp encodings;
5. source-token namespace construction;
6. associative and commutative circuit normalization;
7. multiplicity and coefficient representation;
8. content-addressed sharing;
9. unknown, opaque, partial, and redacted markers; and
10. version negotiation and unsupported-version behavior.

Two equivalent polynomials need not have the same syntax. A practical profile must decide whether canonical identity means syntactic normal form, circuit isomorphism, expanded polynomial equality, or denotational equality under a bounded evaluator. The first release should use a deterministic syntactic normal form and separately test denotation on finite models.

Counterexample 8.1 (Commutative but nondeterministic). One runtime serializes $x + y$; another serializes $y + x$. They evaluate equally but receive different digests. If artifact identity depends on raw construction order, cache keys, differential tests, and forward indexes diverge. Sorting by a stable child digest repairs this example but does not by itself solve all algebraic equivalence.

8.2 Identity is typed

The text `"42"` may be a source row key, target entity key, plan node, execution activity, or content digest fragment. Identity comparison is defined only within compatible namespaces and kinds. Cross-system translation is an explicit mapping with its own evidence.

Requirement 8.2 (No inferred authenticity)**.** Receiving a stable source identifier establishes only that the connector reported that identifier under its contract. Authenticity requires separate transport, credential, signature, attestation, or trusted-source evidence.

9 Provenance query surface

9.1 Operations

The proposed API exposes typed questions rather than a single unrestricted graph endpoint:

```
origin(result_ref, field?, model, depth, principal)
why(result_ref, record, model, depth, principal)
derivation(result_ref, record, representation, budget, principal)
upstream(result_ref, depth, kinds, principal)
downstream(artifact_or_token_ref, depth, kinds, principal)
versions(result_ref, kinds)
assumptions(result_ref, include_deferred)
```

```
decisions(result_ref, status, principal)
execution(result_ref, stages, attempts, principal)
schema_survival(result_ref, proposed_change_ref)
export_prov(result_ref, profile, principal)
  -> {prov_document, loss_report, completeness, ...}
```

Every response includes:

- question and normalized parameters;
- model and granularity;
- provenance-profile and result-envelope revisions;
- completeness class and reasons;
- source frontiers and retained interval;
- policy principal and redaction summary;
- returned nodes, edges, circuits, or origins;
- truncation, depth, count, byte, and time budgets;
- evidence digests; and
- unsupported or unknown fields.

For `export_prov`, `loss_report` is required rather than optional. It names each represented, summarized, extended, omitted, redacted, or opaque internal structure. An export without that field is invalid even if its PROV document satisfies the external syntax and constraints.

9.2 Bounded expansion

Provenance queries can be larger or more sensitive than the result. The API must bound depth, nodes, edges, circuit expansion, response bytes, CPU, and wall time. Truncation changes completeness for the requested expansion and is returned as data, not buried in a transport warning. Continuation tokens bind the result, profile, principal, and query parameters to prevent cross-policy reuse.

9.3 Schema survival

Provenance helps identify affected artifacts, but it cannot by itself prove that a result survives a schema change. The procedure is:

1. resolve the old result envelope and proposed change;
2. find dependencies through typed artifact and field/range references;
3. classify obviously unaffected references where the profile supports such a rule;
4. revalidate mappings, queries, constraints, decisions, and materializations under the changed semantics;

5. replay or recompute where static reasoning is insufficient; and
6. report survived, invalidated, unknown, or requires-replay with evidence.

Counterexample 9.1 (Reachability is not survival). A column is renamed with an apparently preserved identifier, but its unit changes from dollars to cents. A graph reachability check finds the same dependency. The old result does not survive without validating the mapping and scalar semantics. Provenance locates the risk; it does not discharge it.

10 W3C PROV interoperability

10.1 Projection

W3C PROV defines a conceptual model and representations centered on entities, activities, agents, generation, use, derivation, attribution, association, and related qualified influences [24, 26]. CatDB can project:

- immutable source, semantic, plan, materialization, and result artifacts to `prov:Entity`;
- mapping, migration, query, reconciliation, refresh, and publication runs to `prov:Activity`;
- users, services, connectors, organizations, and policy authorities to `prov:Agent`;
- inputs to `prov:used`;
- outputs to `prov:wasGeneratedBy`;
- revisions and derivations to qualified PROV relations; and
- planned artifacts to `prov:Plan` where appropriate.

The export must satisfy the applicable PROV constraints [11]. Access and query behavior should also heed the PROV-AQ separation between locating provenance and deciding whether it is authoritative, correct, trustworthy, or disclosable [22].

10.2 Loss report

The projection does not automatically preserve:

- additive versus multiplicative derivation structure;
- coefficients and bag multiplicity;
- canonical circuit identity;
- precise field or range origin;
- profile-relative exactness;
- unsupported-operator semantics;

- uncertainty and reconciliation outcome distinctions;
- policy-redaction behavior; or
- source-token namespace constraints.

Every export therefore includes a machine-readable loss report naming which internal structures were represented, summarized, extended, omitted, redacted, or made opaque.

Requirement 10.1 (No lossy round-trip claim)**.** An imported PROV graph is external evidence. It is not a verified internal derivation envelope unless it carries a recognized profile, validates, has trusted identity bindings, and contains all required structures for the question.

Evidence status. SUPPORTED BY PRIOR LITERATURE for the W3C model; OPEN for the CatDB projection, validation, and loss audit..

11 Trust, integrity, privacy, and retention

11.1 Provenance is not truth

Provenance can support an assessment of quality, reliability, or trustworthiness. It does not produce that assessment by itself. A precise trace of a false source remains precise. A signed false statement remains false. A trusted agent may make a mistaken decision. An exact circuit can correctly explain a semantically invalid query.

The envelope separates:

- identity evidence;
- integrity evidence;
- authority and policy decisions;
- uncertainty or confidence;
- derivation semantics;
- execution observations; and
- completeness.

11.2 Integrity

Secure provenance work treats integrity and confidentiality as distinct requirements [7, 13, 19]. A hash chain may expose modification or reordering under a named serialization and threat model. It does not establish that every event was captured, that a compromised producer told the truth, or that the first trusted anchor was correct.

The proposed integrity manifest records event digests, parent links, canonicalization profile, signer or trusted anchor when used, append semantics, gap detection, clock assumptions, and verification outcome. Optional cryptography must be described by the exact property it supplies.

Counterexample 11.1 (A complete-looking hash chain). A malicious producer omits one source read, then hashes every remaining event correctly. Chain verification passes. The chain proves consistency of the retained sequence under the hash assumption, not capture completeness.

11.3 Confidentiality and disclosure

Provenance can reveal protected data even when the result is public. A hidden edge may expose participation. Node counts can reveal alternatives. Timing can reveal whether a protected branch exists. A forward-impact query can reveal tenants or models that depend on a source.

Policy therefore applies to provenance operations, not only to result rows. Each response is evaluated under a principal, purpose, tenant, profile, and retention context. Redaction must consider nodes, edges, attributes, counts, continuation tokens, error classes, and timing. Returning an empty exact graph when evidence was withheld is forbidden; the response is Redacted or a policy denial.

11.4 Retention and deletion

Retention can remove raw source values while preserving digests, record tokens, circuit structure, or aggregate audit facts. Deletion policy can conflict with reproducibility and accountability. The profile does not hide that conflict.

A retention revision specifies:

- evidence classes and retention horizons;
- legal or policy basis;
- compaction and deletion operations;
- tombstone or proof-of-deletion behavior;
- effects on backward, forward, value-origin, and export queries;
- downgrade rules for completeness; and
- revalidation of downstream materializations and caches.

Exact provenance is available only relative to retained evidence. No paper claim should imply indefinite perfect history.

12 Systems precedents and evaluation baselines

Source-tagging systems already address heterogeneous origin [30]. ULDBs and Trio connect lineage with uncertainty [1, 6]. Panda integrates provenance and data [20]. GProM uses middleware and query rewriting across several provenance tasks [5]. ProvSQL brings several provenance interpretations to PostgreSQL [29].

Physical capture choices matter. Titian integrates record-level lineage with Spark [21]. Smoke places capture logic in physical operators and optimizes expected lineage queries [28].

Provenance-aware storage captures system-level history automatically, while also exposing the semantic limits of an opaque storage boundary [27]. These are direct baselines, not background decoration.

Formal explanation work also demands exact guarantees. Trace slicing for database queries can state disclosure and recomputation properties [12]. P13 must likewise say what an explanation preserves and under which calculus, rather than claiming that a graph "explains" an answer.

The CQL project reports symbolic lineage and JDBC import/export behavior in its official materials [10]. Its documentation also warns about convenience-generated identifiers. This is external precedent, not evidence that CatDB implements categorical provenance or inherits CQL performance.

Table 4: Baseline families and the mismatch that must be retained.

Baseline	Comparison	Required caveat
PostgreSQL without provenance	execution and storage overhead	no fine-grained explanation baseline
ProvSQL	relational annotations and where-provenance	not the full version/workflow/decision envelope
GProM	query-rewrite provenance tasks	different capture and backend architecture
Smoke	eager, lazy, and deferred lineage tradeoffs	operator-integrated lineage, not semantic artifact provenance
Titian	distributed dataflow lineage	Spark-specific execution model
raw event log	prospective/retrospective activity	no exact record derivation semantics
W3C PROV graph	interoperability and graph queries	does not preserve the internal circuit by default
full recomputation	data and provenance result	expensive but necessary correctness oracle for the supported fragment

13 Evaluation protocol

13.1 Research questions

RQ1 Do normalized derivations match an independent oracle for every admitted fixture and generated finite model?

RQ2 Does composed execution preserve sequential mapping and query provenance?

RQ3 Does incremental maintenance match full recomputation for both data and provenance?

RQ4 What capture, storage, and explanation-query overhead does each provenance mode impose?

RQ5 Do W3C PROV exports validate and report all declared projection loss?

RQ6 Do schema-survival answers match seeded semantic changes and replay outcomes?

RQ7 Do disclosure policies prevent unauthorized inference without silently returning complete-looking answers?

RQ8 Does the cross-layer envelope improve explanation accuracy over row-lineage and raw-log baselines?

13.2 Modes and ablations

At minimum, compare provenance off; artifact identity only; record lineage set; exact positive derivation circuit; circuit plus value origin; prospective plus retrospective activities; the full admitted decision envelope; and an explicitly approximate sketch mode. Ablate artifact versions, source frontiers, plan identity, decision evidence, value origin, completeness metadata, and retention/disclosure fields one at a time.

The off mode measures overhead but cannot serve as a correctness baseline for provenance. A weaker profile cannot be used to claim the cost of a stronger one.

13.3 Fixtures and generated models

Hand-worked families must cover:

- token namespace collision, snapshot drift, and unstable row IDs;
- selection preservation and rejection;
- projection alternatives and bag coefficients;
- union alternatives, self-join multiplication, and join fan-out;
- sequential versus composed typed mappings;
- unsupported difference, aggregation, null, volatile, and opaque operators;
- accepted, rejected, deferred, and reversed reconciliation decisions;
- materialization refresh, duplicate delivery, out-of-order change, crash, and cursor atomicity;
- valid, invalid, lossy, redacted, and partial PROV exports;
- retention-induced downgrade; and
- graph-shape, count, timing, and continuation-token disclosure.

The independent finite evaluator may not call Rust, SQL, a backend, FFI, or the fixture's expected field. Exhaustive small models and independently seeded property tests compare result bags and circuits. A found counterexample is minimized and retained. Passing finite tests do not prove the unbounded runtime.

13.4 Metrics

Correctness metrics include token false positives and negatives, exact circuit agreement, value-origin agreement, composed/sequential agreement, full/incremental data and provenance agreement, PROV validation and loss agreement, schema-survival precision and recall on seeded changes, and unauthorized disclosure failures.

Performance metrics include capture CPU and wall time, execution slowdown, bytes per source and output row, circuit nodes and edges, sharing ratio, write amplification, forward and backward query latency, export time and size, policy overhead, refresh overhead, and retention-compaction cost.

Operational metrics include missing observations, opaque activities, retries, duplicate events, cursor failures, stale materialization use, retention-induced unknown answers, disclosure-denied expansions, and workloads exceeding resource budgets.

13.5 Run contract

Each immutable run records:

- hypothesis and claim IDs;
- profile, fixture, schema, mapping, query, and plan revisions and digests;
- repository revision and dirty state;
- runtime, container, operating system, CPU, memory, storage, backend versions, and configuration;
- dataset generator, seed, source snapshots, and workload split;
- provenance mode, retention, disclosure, and completeness policy;
- warmup, repetitions, exclusions, failures, and timeouts;
- raw outputs and checksums;
- statistical method, intervals, and correction policy;
- preregistered acceptance and falsification thresholds; and
- every failed or rejected run.

No numerical threshold is supplied in this manuscript because no pilot run supports one. A future preregistration must choose workload-specific correctness requirements and practical overhead bounds before examining held out outcomes.

14 Implementation roadmap

14.1 I0: freeze the profile

Specify query fragment, tokens, snapshots, artifact references, circuit syntax, value origins, activities, decisions, result envelope, normalization, completeness, retention, disclosure, W3C projection, and version negotiation. Independent database-provenance and security review must find no unresolved critical or important ambiguity.

14.2 I1: hand-worked fixtures

Create an immutable schema-v2 extension and provenance-v1 cases. Each case states the question, model, exact input, hand-derived output, completeness, negative alternative, and rationale. A manifest hashes every artifact.

14.3 I2: independent Haskell oracle

Implement a total finite denotation for the admitted operators and record lineage. It consumes decoded inputs and emits normalized bags and circuits. It must not invoke Rust, SQLite, PostgreSQL, SQL parsing, FFI, or expected fixture outputs.

14.4 I3: owning Rust abstraction

Create `catdb-provenance` as the only event and result-envelope schema. It owns token, circuit, normalization, query, completeness, retention, and projection types. Other crates emit typed obligations and observations. A connector-specific lineage encoding is translated at the boundary and never becomes a second authoritative model.

The first circuit profile uses a versioned deterministic normal-form serialization: domain-separated node tags; canonical scalar and coefficient bytes; ordered children for noncommutative constructors; lexicographically sorted child digests with retained multiplicities for commutative constructors; and a domain-separated root digest. Decoders reject duplicate encodings of one normal form. Shared golden fixtures require Rust and Haskell to emit identical bytes and digests; finite denotation tests remain a separate check because byte identity alone does not prove semantic completeness.

14.5 I4: capture hooks

Add capture in dependency order:

1. schema, mapping, migration, and query compilation;
2. connectors and source frontiers;
3. planner and executor;
4. reconciliation decisions;
5. materialization and checkpoint publication;

6. workspaces and distribution; and
7. policy, export, retention, and deletion.

No hook may claim a stronger identity or completeness class than its owning boundary can support.

14.6 I5: queries and interoperability

Implement bounded origin, why, derivation, upstream, downstream, versions, assumptions, decisions, execution, survival, and PROV export operations. Test policy on graph structure and metadata, not only on returned values.

14.7 I6: equivalence and backend conformance

Compare the independent oracle, Rust, SQLite, and each supported PostgreSQL version on the portable fragment. Then test full versus incremental recomputation, connector translation, crash recovery, and cursor atomicity. Only passing these gates can promote `SYS-PROV-01` or `SYS-MAT-01`.

15 Formalization roadmap

Formalization should target the finite core, not simulate a database engine.

15.1 F0: representation and denotation

Define finite token namespaces, well-formed typed references, positive derivation circuits, normalization, and evaluation into a commutative semiring. Candidate theorems are:

$$\mathrm{WellFormed}(c) \Longrightarrow \mathrm{WellFormed}(\mathsf{Norm}(c)), \tag{6}$$

$$\mathsf{Eval}(\mathsf{Norm}(c)) = \mathsf{Eval}(c), \tag{7}$$

$$\mathrm{share}(c) \simeq c. \tag{8}$$

The last relation means equal denotation under content-addressed sharing, not equal runtime bytes without a serialization theorem.

15.2 F1: operator and substitution laws

Prove the admitted scan, selection, projection, union, and join laws; semiring identities; associative typed substitution; and selected rewrite equivalences. State bag assumptions and exclude unsupported operators.

15.3 F2: finite incremental equality

For selected positive operators, define batch and delta semantics and prove normalized data and provenance equality. Crash recovery, connector behavior, filesystem durability, and database transaction behavior remain systems tests.

15.4 F3: finite PROV projection properties

A tractable target is that required entity, activity, and agent references appear injectively in the projection and that the loss report marks every collapsed circuit or value-origin component. Formalizing the whole W3C PROV family is not an initial objective.

15.5 F4: later security theorem

A later profile may state noninterference for two runs that differ only in high-confidentiality provenance observed by a low principal. Access-control unit tests do not prove this theorem. *Evidence status.* OPEN. No P13 Lean declaration exists. Even future kernel acceptance would not establish runtime capture completeness..

16 Falsification conditions

The central engineering hypothesis is rejected or narrowed if a retained case shows any of the following:

1. two different source observations normalize to one token;
2. the same observation normalizes differently across implementations;
3. an admitted query disagrees with the independent oracle on rows, multiplicity, tokens, or circuits;
4. composed and sequential mappings disagree;
5. an enabled rewrite preserves rows but changes required provenance;
6. a source, connector, or opaque activity is omitted while an answer is marked exact;
7. row lineage is returned as exact value origin;
8. a reversed reconciliation decision cannot identify affected results;
9. incremental and full recomputation disagree on data or provenance;
10. a crash advances a cursor without corresponding durable effects;
11. W3C PROV export is invalid or omits declared loss;
12. PROV import is treated as trusted internal evidence without required bindings;
13. a retention or redaction event leaves an answer marked unqualified exact;

14. protected participation is inferable from graph shape, counts, errors, continuation tokens, or timing;
15. a hash chain is presented as proof of capture completeness;
16. provenance overhead exceeds preregistered bounds for the target workload;
17. the cross-layer envelope does not improve answer correctness over row-lineage and event-log baselines; or
18. the proposed delta is fully explained by an existing system under the same semantic, version, decision, and disclosure profile.

Failures are research outputs. A passing rerun cannot replace a failed run without retaining both and documenting the cause.

17 Limitations and exact nonclaims

The profile is intentionally narrow. It does not explain arbitrary SQL, general recursion, machine-learning internals, human judgment, or remote APIs. The positive algebra does not silently expand to negation, aggregation, nulls, ordering, or volatile functions. Stable identifiers remain claims within a namespace, not proof of authenticity. Retention and disclosure can make once-available explanations incomplete.

For avoidance of doubt:

1. CatDB does not invent why-, where-, lineage-, or semiring provenance.
2. It does not provide complete provenance for arbitrary code, APIs, SQL, recursion, machine-learning models, or human decisions.
3. Category theory does not eliminate provenance-capture cost.
4. A schema mapping does not by itself determine instance or query provenance.
5. A lineage set is not a complete explanation.
6. W3C PROV is an interoperability model, not the complete internal derivation semantics.
7. Provenance does not prove truth, correctness, authority, identity, trust, or reproducibility.
8. Source identity does not establish source authenticity.
9. Vector similarity does not establish semantic identity.
10. A hash chain does not establish completeness or immutability without a threat model and trusted anchors.
11. Complete provenance is not free in runtime, storage, query, privacy, or operational complexity.
12. Exact provenance is profile-relative and may be unavailable after retention or redaction.

13. Partial source availability cannot yield an unqualified complete answer.
14. Record lineage does not automatically provide field-level origin.
15. Positive semiring provenance does not automatically cover difference, negation, aggregation, recursion, nulls, ordering, or volatile functions.
16. Incremental data equality does not establish provenance equality.
17. A physical execution trace does not prove semantic plan correctness.
18. Provenance alone cannot show that a result survives schema change; the changed semantics must be revalidated.
19. A finite execution DAG does not imply that the complete version history is acyclic.
20. CQL's external capabilities do not establish CatDB implementation or performance.

These are design constraints, not rhetorical disclaimers. A future implementation or paper result that violates one must change the profile or narrow the claim.

18 Open questions

The most immediate questions are practical:

- Is the positive fragment useful enough to justify exact circuits?
- Should v1 store only circuits, or circuits plus a bounded expanded form?
- Which source observations are strong enough for ExactForProfile?
- How are synthesized target objects tokenized?
- Which mapping operator supplies P4's instance-level derivation?
- Which reconciliation outcomes may enter results without approval?
- What normalization supports independent agreement without excessive expansion?
- Which graph redactions avoid participation leakage?
- What retention horizon is needed for audit, rollback, and practical reproduction?
- Which provenance modes are affordable in embedded SQLite and remote PostgreSQL?
- How are remote provenance statements authenticated?
- What is the first useful why-not profile?
- Which PROV extensions preserve decisions and circuit references without misleading consumers?

The first valid implementation artifact is the reviewed profile and fixture suite. Drafting a stronger abstract or adding an untyped event log would not move the central claim.

19 Conclusion

Provenance becomes first-class when execution cannot return an explanation without saying what kind of explanation it is. A source token, lineage set, value origin, derivation circuit, version reference, physical trace, decision record, and PROV graph each carry useful information. They also each omit something. The proposed contract keeps those omissions visible.

For the first positive fragment, the relational algebra can reuse provenance semirings. The remaining work is the bridge: typed substitution across mappings, immutable version and snapshot references, retrospective activity, reconciliation evidence, materialization equality, bounded provenance queries, W3C PROV projection with loss, and completeness that weakens honestly under opacity, retention, or disclosure.

No such CatDB runtime exists in the current repository. The manuscript therefore ends at a falsifiable contract. Progress requires reviewed profiles, hand-derived fixtures, an independent oracle, an owning implementation, backend and crash conformance, retained benchmarks, and selected finite theorems. Until those gates pass, "first-class execution semantics" names a research hypothesis, not a systems result.

A Claim ledger

Table 5: Manuscript claim status.

ID	Claim	Status	Falsifier or gate
P13-C01	why, where, lineage, how, workflow, and PROV are distinct models	SUPPORTED BY PRIOR LITERATURE	primary sources contradict the distinction
P13-C02	positive annotations compose by semiring addition and multiplication	SUPPORTED BY PRIOR LITERATURE	cited result does not support stated fragment
P13-C03	typed versioned envelope can answer the fixed cross-layer questions	ENGINEERING HYPOTHESIS	implementation fails correctness or utility gates
P13-C04	profile-v1 normalization is deterministic and denotation preserving	OPEN	cross-language disagreement or theorem failure
P13-C05	sequential and composed mapping provenance agree	OBSTRUCTED	P4 semantics and fixtures required
P13-C06	source tokens identify exact observed records at frontiers	OPEN	collision, drift, or unsupported connector identity
P13-C07	reconciliation evidence remains distinct from truth	OBSTRUCTED	P7 authority and decision profile required

ID	Claim	Status	Falsifier or gate
P13-C08	incremental and full data plus provenance agree	OPEN	retained mismatch or crash/cursor violation
P13-C09	PROV projection is valid and its loss report complete	OPEN	validator or loss-oracle mismatch
P13-C10	disclosure avoids protected inference	OPEN	red-team graph, count, error, token, or timing leak
P13-C11	the envelope has acceptable overhead	OPEN	preregistered workload bound exceeded
P13-C12	selected finite circuit laws are machine checked	OPEN	named declarations and axiom audit absent

B Planned artifact layout

```
runs/<run-id>/
  preregistration.json
  manifest.json
  environment.json
  profiles/
  fixtures/
  source-frontiers/
  artifacts/
  circuits/
  value-origins/
  activities/
  decisions/
  materializations/
  explanation-queries.jsonl
  prov-export/
  prov-loss-report.json
  schema-survival.jsonl
  incremental-equivalence.jsonl
  disclosure-audit.jsonl
  failures/
  raw/
  analysis/
  report/
  checksums.txt
```

The current repository contains no P13 run with this layout.

C Review checklist

A review candidate should be rejected or returned for correction if:

1. a provenance model is named imprecisely;
2. "exact" lacks profile, question, granularity, frontier, retention, or disclosure scope;
3. a lineage set is treated as a derivation circuit or value origin;
4. algebraic scope silently includes an unsupported operator;
5. a plan trace is treated as proof of semantic correctness;
6. source identity is treated as authenticity;
7. reconciliation evidence is treated as truth;
8. PROV projection loss is omitted;
9. retained security evidence tests only data-row access;
10. incremental evidence compares data without provenance;
11. benchmark modes use unequal provenance profiles;
12. a formal result is generalized to runtime capture; or
13. a CatDB claim is promoted without a digest-bound artifact and executable witness.

References

[1] Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Chris Hayworth, Shubha U. Nabar, Tomoe Sugihara, and Jennifer Widom. Trio: A system for data, uncertainty, and lineage. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 1151–1154. VLDB Endowment, 2006. URL https://www.vldb.org/conf/2006/p1151-agrawal.pdf.

[2] Yael Amsterdamer, Susan B. Davidson, Daniel Deutch, Tova Milo, Julia Stoyanovich, and Val Tannen. Putting lipstick on pig: Enabling database-style workflow provenance. *Proceedings of the VLDB Endowment*, 5(4):346–357, 2011. doi: 10.14778/2095686.2095693.

[3] Yael Amsterdamer, Daniel Deutch, and Val Tannen. Provenance for aggregate queries. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 153–164. ACM, 2011. doi: 10.1145/1989284.1989302.

[4] Yael Amsterdamer, Daniel Deutch, and Val Tannen. On the limitations of provenance for queries with difference. In *3rd USENIX Workshop on the Theory and Practice of Provenance*. USENIX Association, 2011. URL https://www.usenix.org/conference/tapp11/limitations-provenance-queries-difference.

[5] Bahareh Sadat Arab, Su Feng, Boris Glavic, Seokki Lee, Xing Niu, and Qitian Zeng. GProM: A swiss army knife for your provenance needs. *IEEE Data Engineering Bulletin*, 41(1):51–62, 2018. URL https://par.nsf.gov/biblio/10082097-gprom-swiss-army-knife-your-provenance-needs.

[6] Omar Benjelloun, Anish Das Sarma, Alon Y. Halevy, and Jennifer Widom. ULDBs: Databases with uncertainty and lineage. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 953–964. VLDB Endowment, 2006. URL https://www.vldb.org/conf/2006/p953-benjelloun.pdf.

[7] Uri Braun, Avraham Shinnar, and Margo Seltzer. Securing provenance. In *1st USENIX Workshop on Hot Topics in Security*. USENIX Association, 2008. URL https://www.usenix.org/event/hotsec08/tech/full_papers/braun/braun.pdf.

[8] Mihai Budiu, Tej Chajed, Frank McSherry, Leonid Ryzhyk, and Val Tannen. DBSP: Automatic incremental view maintenance for rich query languages. *Proceedings of the VLDB Endowment*, 16(7):1601–1614, 2023. doi: 10.14778/3587136.3587137.

[9] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and where: A characterization of data provenance. In *Database Theory — ICDT 2001*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330. Springer, 2001. doi: 10.1007/3-540-44503-X_20.

[10] CategoricalData. CQL: Categorical query language official site and frequently asked questions. Official project documentation, 2026. URL https://categoricaldata.net/. FAQ: https://github-wiki-see.page/m/CategoricalData/CQL/wiki/frequently-asked-questions; accessed 23 July 2026.

[11] James Cheney, Paolo Missier, and Luc Moreau. Constraints of the PROV data model. W3c recommendation, World Wide Web Consortium, 2013. URL https://www.w3.org/TR/prov-constraints/.

[12] James Cheney, Amal Ahmed, and Umut A. Acar. Database queries that explain their work. *arXiv preprint arXiv:1408.1675*, 2014. URL https://arxiv.org/abs/1408.1675.

[13] Stephen Chong. Towards semantics for provenance security. In *1st USENIX Workshop on the Theory and Practice of Provenance*. USENIX Association, 2009. URL https://static.usenix.org/events/tapp09/tech/full_papers/chong/chong_html/.

[14] Yingwei Cui, Jennifer Widom, and Janet L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM Transactions on Database Systems*, 25(2):179–227, 2000. doi: 10.1145/357775.357777.

[15] Susan B. Davidson and Juliana Freire. Provenance and scientific workflows: Challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1345–1350. ACM, 2008. doi: 10.1145/1376616.1376772.

[16] Daniel Deutch, Tova Milo, Sudeepa Roy, and Val Tannen. Circuits for datalog provenance. In *17th International Conference on Database Theory*, pages 201–212. OpenProceedings.org, 2014. doi: 10.5441/002/ICDT.2014.22.

[17] Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 31–40. ACM, 2007. doi: 10.1145/1265530.1265535.

[18] Ashish Gupta, Inderpal Singh Mumick, and V. S. Subrahmanian. Maintaining views incrementally. *ACM SIGMOD Record*, 22(2):157–166, 1993. doi: 10.1145/170036.170066.

[19] Ragib Hasan, Radu Sion, and Marianne Winslett. Secure provenance: Protecting the genealogy of bits. *USENIX ;login:*, 34(3):42–49, 2009. URL https://www.usenix.org/legacy/publications/login/2009-06/openpdfs/hasan.pdf.

[20] Robert Ikeda and Jennifer Widom. Panda: A system for provenance and data. In *2nd USENIX Workshop on the Theory and Practice of Provenance*. USENIX Association, 2010. URL https://www.usenix.org/conference/tapp-10/panda-system-provenance-and-data.

[21] Matteo Interlandi, Kshitij Shah, Sai Deep Tetali, Muhammad Ali Gulzar, Seunghyun Yoo, Miryung Kim, Todd Millstein, and Tyson Condie. Titian: Data provenance support in spark. *Proceedings of the VLDB Endowment*, 9(3):216–227, 2015. doi: 10.14778/2850583.2850595.

[22] Graham Klyne and Paul Groth. Provenance access and query. W3c working group note, World Wide Web Consortium, 2013. URL https://www.w3.org/TR/prov-aq/.

[23] Sven Köhler, Bertram Ludäscher, and Daniel Zinn. First-order provenance games. In *In Search of Elegance in the Theory and Practice of Computation*, volume 8000 of *Lecture Notes in Computer Science*, pages 382–399. Springer, 2013. doi: 10.1007/978-3-642-41660-6_20.

[24] Timothy Lebo, Satya Sahoo, Deborah McGuinness, et al. PROV-O: The PROV ontology. W3c recommendation, World Wide Web Consortium, 2013. URL https://www.w3.org/TR/prov-o/.

[25] Pengyuan Li, Boris Glavic, Dieter Gawlick, Vasudha Krishnaswamy, Zhen Hua Liu, Danica Porobic, and Xing Niu. In-memory incremental maintenance of provenance sketches. In *Proceedings of the 29th International Conference on Extending Database Technology*. OpenProceedings.org, 2026. doi: 10.48786/EDBT.2026.05.

[26] Luc Moreau and Paolo Missier. PROV-DM: The PROV data model. W3c recommendation, World Wide Web Consortium, 2013. URL https://www.w3.org/TR/prov-dm/.

[27] Kiran-Kumar Muniswamy-Reddy, David A. Holland, Uri Braun, and Margo Seltzer. Provenance-aware storage systems. In *2006 USENIX Annual Technical Conference*. USENIX Association, 2006. URL https://www.usenix.org/conference/2006-usenix-annual-technical-conference/provenance-aware-storage-systems.

[28] Fotis Psallidas and Eugene Wu. Smoke: Fine-grained lineage at interactive speed. *Proceedings of the VLDB Endowment*, 11(6):719–732, 2018. doi: 10.14778/3184470.3184475.

[29] Pierre Senellart, Louis Jachiet, Silviu Maniu, and Yann Ramusat. ProvSQL: Provenance and probability management in PostgreSQL. *Proceedings of the VLDB Endowment*, 11 (12):2034–2037, 2018. doi: 10.14778/3229863.3236253.

[30] Y. Richard Wang and Stuart E. Madnick. A polygen model for heterogeneous database systems: The source tagging perspective. In *Proceedings of the 16th International Conference on Very Large Data Bases*, pages 519–538. Morgan Kaufmann, 1990. URL https://www.vldb.org/conf/1990/P519.PDF.