Composable Schema Mappings for Data Integration: From Point-to-Point ETL to Semantic Compilation

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

23 July 2026

Abstract

Data-integration programs often repeat the same source and consumer semantics in pair-specific pipelines. A shared domain reduces that duplication only when the required transformations really do factor through it, the mapping language remains closed over the required compositions, and the domain retains every distinction its consumers need. We define a provisional CatDB contract for testing those conditions. Sources map into one versioned domain, which then maps into consumers. Pair-specific exceptions, identity reconciliation, information loss, ambiguity, governance, and physical operations still count as costs.

The familiar $N \times M$ versus $N+M$ comparison is useful only as conditional accounting. It is neither a universal theorem nor a novelty claim. For an actual connection relation $A \subseteq I \times J$, which may be sparse, the direct baseline has $|A|$ mappings. The mediated condition has source and consumer mappings only if every required pair factors, and it also incurs domain maintenance, reconciliation, exception, and operational work. We formalize this accounting with a three-source, three-consumer example. The example includes local source and consumer changes, a domain-wide change, a pair-specific privacy and freshness exception, an unresolved identity candidate, a sparse topology, and a poor domain that erases a distinction one consumer needs.

Schema arrows alone do not determine data-flow direction. For a schema mapping $F : S \to T$, established functorial migration has both covariant operators $\Sigma_F, \Pi_F : \mathbf{Inst}(S) \to \mathbf{Inst}(T)$ and the contravariant operator $\Delta_F : \mathbf{Inst}(T) \to \mathbf{Inst}(S)$. Consequently, $\Delta_{G \circ F}$ is not forward ETL from S to T. A forward integration claim requires a separately selected and versioned data operator, data-exchange construction, query rewrite, or materialization semantics.

Current CatDB evidence is narrower. Rust computes a finite total-mapping and composition subset, while Lean machine-checks selected mapping identity, closure, unit, and associativity laws under audited assumptions. The current loss envelopes are structural only. There is no accepted forward integration operator, factorization checker, connector, identity resolver, compiled plan, Rust/Haskell result comparison, or

repair-surface experiment. The claim that domain mediation reduces total correct-repair work therefore remains an ENGINEERING HYPOTHESIS. This paper does not claim to eliminate ETL, ELT, data movement, cleaning, scheduling, loading, retries, governance, or distributed-systems costs.

Contents

1 Introduction

1.1 Plain-language integration problem

Consider a commerce organization that has three source systems. A store database uses integer customer identifiers, foreign keys, and one-letter status codes. A marketplace export nests buyer data inside purchases and treats an email address as a local key. A partner API uses URIs, links, and numeric lifecycle codes. Three consumers need the data in different

forms: a support application wants a current operational view, a warehouse wants analytical order facts, and an external API wants nested customer documents.

A direct implementation can build a transformation for each required source-consumer pair. If all nine pairs are needed, it has nine definitions. A mediated implementation instead maps each source into one domain and the domain into each consumer, leaving six base mappings in the diagram. The smaller count does not settle the hard questions. Each direct transformation must still factor through the domain. The domain must not collapse a status that the warehouse needs. A marketplace-to-support pair may have a privacy rule that no other pair shares, and someone must have authority to decide whether two source records denote the same customer. The design must also account for data movement, freshness, and the effect of later schema changes.

CatDB proposes that these decisions become typed, composable, versioned artifacts. Federation, ELT, ETL, and materialization remain physical strategies. The proposed semantic layer does not remove extraction, transport, staging, cleaning, deduplication, scheduling, backfill, quality monitoring, retries, loading, rate limits, or incident response. ARKTOS and subsequent ETL work already treat those activities as real workflow concerns [25]; official ELT documentation likewise places transformation after loading rather than making transport disappear [13].

1.2 Research question

The central question is:

> *Under what assumptions can source-to-domain and domain-to-consumer mappings replace pairwise integration logic, and when do they reduce total correct-repair work rather than merely reduce the number of arrows in a diagram?*

The conditional hypothesis is:

> When required source-to-consumer transformations genuinely factor through one adequate shared domain, the mapping language is closed under the needed composition, source and consumer obligations remain separable, and domain, exception, identity, loss, ambiguity, governance, and physical work are counted, a domain-mediated design can reduce independently authored mapping definitions from one per required pair toward one per source plus one per consumer.

Status. ENGINEERING HYPOTHESIS.

The hypothesis concerns a bounded integration fragment and a controlled population of changes. It is rejected or narrowed when mappings fail to factor, exceptions or identity work grow pairwise, domain evolution dominates savings, sparse demand favors direct mappings, or fewer artifacts do not produce lower correct-repair work.

1.3 Contributions

This provisional paper contributes:

1. a formal distinction between schema-level mapping composition and the separately selected direction of instance data flow;
2. a connection-graph and cost model that compares $|A|$ direct mappings with mediated base mappings, pair exceptions, domain governance, reconciliation, and physical work;
3. a factorization contract with exact, conditional, lossy, partial, ambiguous, unsupported, and unknown verdicts;
4. a mapping taxonomy covering fields, paths, structures, enums, synthesis, identity, information loss, partiality, and ambiguity;
5. one three-source, three-consumer example containing both a favorable factorization case and counterexamples;
6. an implementation and fixture plan that separates bounded Rust and Lean evidence from missing P4 execution evidence; and
7. a preregisterable repair-surface experiment against pairwise, GAV, LAV, and Information Manifold baselines.

The elementary mapping-count propositions in this paper are PROVED under their assumptions. The selected Lean mapping laws are MACHINE-CHECKED in the audited P1 dependency. Rust fixture results are COMPUTATIONAL. None of those statuses upgrades the P4 maintenance hypothesis to experimentally supported.

1.4 Nonclaims

This paper does not show that every source-consumer transformation factors through one domain, that all mapping languages are closed under composition, or that a canonical schema suits every organization. It does not equate schema matching with an approved executable mapping, object identification with record identity, or vector similarity with identity. It does not treat "no checked loss" as proof of invertibility. It does not claim virtual and materialized results are automatically equivalent or that generated plans match hand-tuned physical performance.

Most importantly, the paper does not claim that ETL, ELT, or physical data movement disappears. It studies whether some semantic transformation logic can move from pair-specific code into reusable mappings while the operational work remains visible and measured.

2 Evidence vocabulary and current status

2.1 Status vocabulary

The manuscript uses evidence labels per claim and per layer.

- SUPPORTED BY PRIOR LITERATURE: a cited primary paper, standard, or official document establishes the result or capability under its own assumptions.

- PROVED: a complete paper proof establishes the exact statement under listed assumptions; this does not mean machine-checked.
- MACHINE-CHECKED: a named Lean declaration was accepted by the pinned kernel with the reported axiom inventory.
- EXPERIMENTALLY SUPPORTED: a reproducible experiment supports an engineering claim within its tested population and conditions. No P4 claim currently receives this label.
- COMPUTATIONAL: a named implementation computed a named finite result under a recorded command.
- STRUCTURAL-ONLY: a representation decoded, normalized, or round-tripped, but the advertised semantic operation was not computed.
- ENGINEERING HYPOTHESIS: a proposed implementation or measurable systems effect lacks the required experiment.
- OPEN CONJECTURE: a mathematical statement remains unproved or unmechanized. This label does not substitute for a missing systems experiment.
- OPEN: definitions, proof, implementation, or experiment remain incomplete.
- OBSTRUCTED: a required language or semantic boundary has not been fixed well enough to implement or test honestly.
- SPECULATIVE: the idea lacks enough definition or evidence for a stronger label. No central P4 claim uses this class.

No P4 claim is labeled experimentally supported in this revision.

2.2 Evidence cut

The evidence cut is 22 July 2026. The accepted prerequisite evidence consists of a finite total schema-mapping subset, bounded Rust composition fixtures, and selected Lean mapping laws. The P4-specific evidence needed for forward integration and repair measurement is absent.

2.3 What prerequisite evidence does not establish

Accepted P1 mapping evidence is necessary for P4 because an integration composition cannot be more reliable than its schema mappings. It is not sufficient. Mapping fixtures do not produce source records, domain instances, consumer records, queries, materializations, provenance, or repair measurements. Lean category laws do not verify Rust, Haskell, JSON fixtures, connectors, business semantics, identity decisions, or physical execution.

The current evidence therefore supports one bounded statement: selected schema mappings can be validated and composed in the current finite profile, and selected general mapping laws have been machine-checked in Lean. It does not support end-to-end integration or a maintenance improvement.

Artifact or claim	Current status	What the status permits
Rust total mapping validation	computational	finite typed assignments and diagnostics in Slice 1
Rust mapping composition	computational	six named composition fixtures in a restricted fragment
Rust identity mapping	computational	two named schema-level identity fixtures
Direct equation preservation	computational, incomplete	only the selected direct target-equation fragment
Lean mapping laws	machine-checked	named identity, closure, unit, and associativity declarations under audited assumptions
Loss envelopes	structural-only	representation exists; semantic loss analysis is not computed
Forward data operator	obstructed	no source-to-domain-to-consumer execution claim
Factorization checker	open	no direct-versus-mediated verdict is computed
Identity reconciliation	open	no record-level equivalence decision engine
Virtual/materialized execution	open	no connector, plan, result, or provenance comparison
Repair-surface experiment	open	no $N + M$-style maintenance result

Table 1: Current P4 evidence boundary.

3 Prior art and novelty boundary

3.1 Categorical integration

Functorial data migration establishes schemas as categories, instances as functors, and the Σ, Δ, and Π migration operators [23]. Relational foundations for functorial migration give compositional query and relational implementation results [24]. Algebraic data integration and algebraic databases develop executable integration and richer typeside semantics [21, 19]. FQL/CQL practice and case studies predate CatDB [28, 2, 4].

CatDB cannot claim that categories first made mappings composable or data integration executable. Its candidate delta is an operational package: versioned mappings, explicit operator direction, conservative loss and ambiguity records, governed identity references, physical strategy separation, and a measured repair-surface study. That package is not yet implemented or evaluated.

3.2 Model management and schema matching

Model management already treats schemas and mappings as first-class objects and studies matching, composition, merging, difference, inversion, translation, and execution [1, 20]. Schema matching already combines names, structure, constraints, and instances [18, 17].

A proposed correspondence is not a lawful mapping, a value conversion, an identity assertion, or an authority decision. CatDB may retain matcher output as versioned candidate

evidence, but the reuse claim begins only after the mapping is typed, validated, assigned a direction and data semantics, and reviewed.

3.3 Mediated schemas, GAV, and LAV

Federation and mediator architectures address heterogeneous, autonomous sources [22, 27]. The Information Manifold describes sources against a mediated domain and selects sources for high-level queries [15]. Data-integration theory formalizes global schemas, source descriptions, GAV, LAV, and query answering [16].

These systems already move integration work from pairwise programs into a mediated representation. GAV defines the global schema in terms of sources and concentrates changes in global views. LAV describes sources against the global schema and moves work into query rewriting. CatDB must compare total correct work against both rather than treating one hub diagram as new.

3.4 Data exchange and composition

Data exchange supplies solutions, universal solutions, chase procedures, fresh values, and certain-answer semantics [10]. Mapping composition has established expressiveness boundaries: ordinary source-to-target tgds are not generally closed under composition, motivating richer second-order dependencies [11].

Therefore, the phrase "mappings compose" is incomplete without a mapping language, equality, and closure result. A CatDB compiler must return an explicit unsupported or residual result when a required composition leaves its language.

3.5 Clio, ETL, and mapping standards

Clio turns user-confirmed correspondences into declarative mappings and transformation queries for relational and XML settings [9]. Its later architecture compiles mappings through an abstract query graph into several physical languages [14]. ARKTOS models and executes ETL workflows, including quality, scheduling, and contingency concerns [25]. R2RML standardizes relational-to-RDF mappings, while RML extends that style to heterogeneous sources [26, 8].

CatDB cannot claim novelty for generating transformations, separating a mapping from target syntax, or handling heterogeneous sources. A defensible comparison must test whether its typed preservation, versioning, composition, loss, ambiguity, identity, and provenance contracts add measurable value.

3.6 Ontology access, semantic layers, and data mesh

Ontop provides mapping-aware virtual access from SPARQL to relational sources [3]. LookML documents version-controlled semantic models with relationships and generated SQL, while the dbt Semantic Layer documents centralized metrics and automatic join handling [12, 5]. These official product documents establish capabilities, not independent evidence for a maintenance curve.

Data-mesh architecture emphasizes domain ownership, data as a product, self-service infrastructure, and federated governance [6, 7]. It also warns against one static enterprise model. P4 therefore treats D as a bounded, versioned domain model, not an automatically universal canonical schema.

3.7 Candidate novelty

The plausible systems contribution is a single restricted integration artifact model that connects:

- typed versioned schema mappings and composition;
- an explicit data-flow operator and result-equivalence contract;
- conservative loss, partiality, synthesis, and ambiguity evidence;
- governed identity decisions outside schema object assignment;
- equivalent virtual and materialized realizations under one oracle; and
- a controlled comparison of total correct-repair effort.

Status. ENGINEERING HYPOTHESIS.

Until those pieces exist together, P4 is a specification and experimental plan rather than a demonstrated CatDB integration contribution.

4 Source-domain-consumer mapping model

4.1 Connection graph

Let

$$I = \{1, \ldots, N\}, \qquad J = \{1, \ldots, M\}$$

index source and consumer schemas. The actual connection relation is

$$A \subseteq I \times J.$$

The bipartite graph (I, J, A), not the product $I \times J$, describes demand. Define the participating endpoints:

$$I_A = \{i \in I \mid \exists j \ (i, j) \in A\}, \qquad J_A = \{j \in J \mid \exists i \ (i, j) \in A\}.$$

Definition 4.1 (Direct integration family)**.** A direct family assigns every required edge $(i, j) \in A$ one independently authored mapping artifact

$$P_{ij} : S_i \to T_j,$$

together with its selected data operator, policies, physical strategy, tests, and provenance requirements.

Definition 4.2 (Domain-mediated family)**.** A domain-mediated family consists of a versioned bounded-domain schema D, one source mapping

$$F_i : S_i \to D \qquad (i \in I_A),$$

one consumer mapping

$$G_j : D \to T_j \qquad (j \in J_A),$$

and pair-specific exception artifacts E_{ij} for the required edges whose semantics cannot be assigned solely to a source, the domain, a consumer, a shared identity policy, or a shared physical capability.

The schema-level composite for a required edge is:

$$H_{ij} = G_j \circ F_i : S_i \to T_j.$$

It exists as a lawful mapping only when schema versions match, object assignments are total, arrow images are well-typed paths, attribute expressions are in the admitted language, required equations are preserved, and composition remains representable.

4.2 Immutable factorization artifact

Definition 4.3 (Factorization artifact)**.** A factorization artifact for $(i, j) \in A$ records:

1. the exact revisions of S_i, D, T_j, F_i, G_j, and P_{ij} when a direct oracle exists;
2. the mapping-language and equality-profile versions;
3. the normalized schema composite H_{ij};
4. the selected data operator, direction, and snapshot semantics;
5. the direct-versus-mediated extensional comparison target;
6. closure, residual, or unsupported evidence;
7. typed path, attribute, equation, and selected constraint obligations;
8. loss, partiality, synthesis, ambiguity, and identity references;
9. pair-specific policy exceptions; and
10. a separate physical federation, ELT, ETL, or materialization plan.

Artifact identity is content-addressed after canonical serialization. A mutable human name may point to a new revision but does not change an existing artifact. A factorization record with a missing operator direction cannot receive an executable verdict.

4.3 Factorization verdicts

Definition 4.4 (Factorization verdict)**.** For one required edge, the checker returns one of:

$$\{\mathsf{exact}, \mathsf{conditional}, \mathsf{lossy}, \mathsf{partial}, \mathsf{ambiguous}, \mathsf{unsupported}, \mathsf{unknown}\}.$$

The meanings are:

exact direct and mediated denotations agree in the admitted fragment under the declared equality.

conditional agreement requires named constraints, coverage predicates, identity decisions, or scalar assumptions.

lossy a concrete checked witness shows that consumer-required information is discarded or identified.

partial the mapping applies only to an explicit subset and reports its excluded cases.

ambiguous non-equivalent alternatives remain without an authority decision.

unsupported the transformation is outside the mapping or execution profile.

unknown the selected sound checks cannot decide the obligation.

The current repository does not compute these P4 verdicts.

5 Data-flow direction and instance semantics

5.1 Schema direction is not data direction

Status. SUPPORTED BY PRIOR LITERATURE.

For a schema mapping $F : S \to T$, established functorial data migration provides:

$$\Sigma_F, \Pi_F : \mathbf{Inst}(S) \to \mathbf{Inst}(T), \qquad \Delta_F : \mathbf{Inst}(T) \to \mathbf{Inst}(S)$$

[23, 24]. The operators have different semantics and directions. In particular:

$$\Delta_{G_j \circ F_i} : \mathbf{Inst}(T_j) \to \mathbf{Inst}(S_i).$$

It consumes a consumer-side instance and returns a source-side instance. Calling this forward ETL from S_i to T_j would reverse the operator's direction.

5.2 Selected forward profile

Definition 5.1 (Forward data profile)**.** A forward data profile φ assigns each admitted schema mapping $F : S \to T$ a partial transformation

$$\mathcal{E}_F^{\varphi} : \mathbf{Inst}_{\varphi}(S) \longrightarrow \mathsf{Outcome}_{\varphi}(T)$$

with versioned semantics for scalar values, nulls, missing values, multiplicity, generated identifiers, constraints, errors, snapshot, provenance, loss, and unsupported cases.

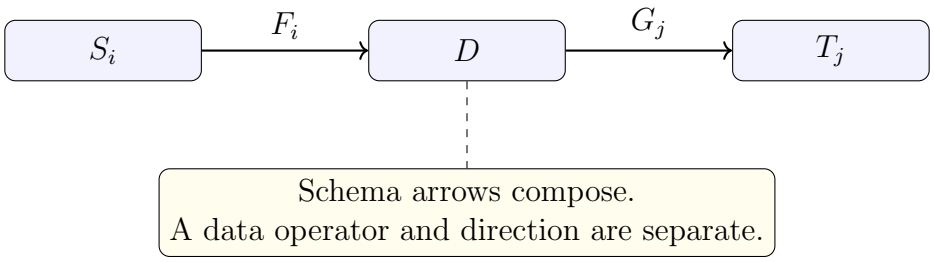

Figure 1: Schema mapping direction does not select Σ, Δ, Π, chase-based exchange, query rewriting, or materialization.

The profile could select a bounded Σ-like construction, a dependency-and-chase data exchange, a query-result semantics, or another explicit forward transformation. It cannot use the generic word "mapping" to conceal which construction runs.

Definition 5.2 (Compositional forward profile)**.** A forward profile is compositional for $F : S \to D$ and $G : D \to T$ when:

$$\mathcal{E}^{\varphi}_{G\circ F}(X) \approx_{\varphi} \mathcal{E}^{\varphi}_{G}\big(\mathcal{E}^{\varphi}_{F}(X)\big)$$

for every admitted $X \in \mathbf{Inst}_{\varphi}(S)$, after accounting for the outcome type and under one declared equality, identifier policy, snapshot, and constraint profile.

This law is an obligation, not a consequence of schema composition alone. If the transformation creates fresh identifiers or quotients values, the profile must specify canonical representatives or an equality insensitive to their spelling. For virtual query integration, the equality may compare result bags or certain answers rather than materialized instances.

5.3 Direct oracle obligation

If a direct baseline P_{ij} exists, a factorization claim requires:

$$[\![P_{ij}]\!]_{\varphi}(X) \approx_{\varphi} [\![G_j \circ F_i]\!]_{\varphi}(X).$$

Syntactic similarity between mapping artifacts is insufficient. The test must run the same source instance, scalar profile, identity decisions, exceptions, and snapshot against both sides.

5.4 Current operator status

Status. OBSTRUCTED.

No accepted CatDB profile currently defines $\mathcal{E}^{\varphi}$ for P4. Rust Slice 1 computes neither Σ, Δ, Π, a chase, a virtual integrated query, nor a materialized plan. Its represented Σ and Π cases are unsupported fixtures. Thus P4 has no computational forward-data factorization result.

6 Conditional $N \times M$ versus $N + M$ accounting

6.1 Artifact count

The direct authored mapping count is:

$$K_{\text{pair}} = |A|.$$

The mediated base count, before exceptions and shared artifacts, is:

$$K_{\text{base}} = |I_A| + |J_A|.$$

It equals $N + M$ only when every source and consumer participates. It does not include D, governance, identity, tests, exceptions, generated plans, or operations.

Proposition 6.1 (Conditional base-mapping count)**.** *Status.* PROVED.

Assume one source-to-domain mapping is authored for every $i \in I_A$, one domain-to-consumer mapping for every $j \in J_A$, and every required edge $(i, j) \in A$ factors through those two mappings without a pair-specific mapping definition. Then the mediated family has

$$|I_A| + |J_A| \leq N + M$$

base mapping definitions and supplies one schema composite for every required edge.

Proof. By assumption, the mediated family contains exactly one F_i for each element of I_A, giving $|I_A|$ source mappings, and exactly one G_j for each element of J_A, giving $|J_A|$ consumer mappings. The two sets are disjoint by role, so their union has size $|I_A| + |J_A|$. Because $I_A \subseteq I$ and $J_A \subseteq J$, the count is at most $N + M$. For each $(i, j) \in A$, both endpoints participate, so F_i and G_j exist and their assumed factorization supplies $G_j \circ F_i$. □

Remark 6.2. The proposition counts base definitions, not total work. Its factorization premise is the central empirical and semantic difficulty. It says nothing about execution, exceptions, identity, governance, domain maintenance, or repair cost.

6.2 Sparse-topology counterexample

Proposition 6.3 (Sparse demand can favor direct definitions)**.** *Status.* PROVED.

There are connection relations with full endpoint participation for which $|A| < N + M$. For those relations, a one-definition-per-edge direct family has fewer authored mapping definitions than the mediated base family before domain and exception costs are added.

Proof. Take $N = M = 3$ and

$$A_{\text{sparse}} = \{(1, 1), (1, 2), (2, 2), (3, 3)\}.$$

Every source and consumer participates, hence $|I_A| + |J_A| = 3 + 3 = 6$. But $|A_{\text{sparse}}| = 4 < 6$. The direct family has four definitions, whereas the mediated family has six base mappings before counting D, governance, identity, or exceptions. □

The example does not prove that direct integration is cheaper overall. It shows why NM is an invalid baseline for a sparse requirement graph.

6.3 Full cost accounting

Let $C(Z)$ measure correct authoring and repair work for artifact or activity Z. Let E_{ij} be pair-specific exception logic, C_D domain maintenance and governance, C_R identity and reconciliation work, and C_X nonsemantic physical and operational work. The comparison is:

$$C_{\text{pair}} = \sum_{(i,j)\in A} C(P_{ij}) + C_X^{\text{pair}},$$

$$C_{\text{mediated}} = \sum_{i\in I_A} C(F_i) + \sum_{j\in J_A} C(G_j) + \sum_{(i,j)\in A} C(E_{ij}) + C_D + C_R + C_X^{\text{mediated}}.$$

The headline count approaches $N + M$ only under dense demand, an adequate domain, factorability, closure, side-local rules, bounded exceptions, a stable identity boundary, manageable domain evolution, comparable correctness, and full accounting. If any assumption fails, the arithmetic does not support a maintenance claim.

6.4 Repair surfaces

For a source-local change c_i to S_i, the ideal pairwise repair touches $\deg_A(i)$ direct definitions. The mediated repair surface is:

$$R^{S_i}_{\text{mediated}} = R(F_i) + R(D_i) + R(R_i) + \sum_{j:(i,j)\in A} R(E_{ij}) + R(\text{generated artifacts}).$$

A consumer-local change has the dual form around G_j. A domain change can touch every participant:

$$R^{D}_{\text{mediated}} = R(D) + \sum_i R(F_i) + \sum_j R(G_j) + R(R) + R(E).$$

Proposition 6.4 (Conditional source locality)**.** *Status.* PROVED.

Suppose a source change c_i preserves the domain contract, affects no shared identity or domain policy, requires no pair-specific exception change, and generated artifacts depend on S_i only through F_i. Then the authored semantic repair set in the mediated dependency graph is contained in $\{F_i\}$.

Proof. By premise, c_i changes neither D, any G_j, any shared identity or domain policy, nor any E_{ij}. Every downstream generated artifact reads the source contract through F_i, so repairing F_i restores the same domain-facing contract. No other authored semantic artifact has a dependency on the changed source representation. Thus the authored semantic repair set is contained in $\{F_i\}$. □

Remark 6.5. This is a dependency-graph result under strong separability premises, not an empirical claim that real changes satisfy them. The repair-surface experiment is designed to measure how often they do.

7 Three-source, three-consumer example

7.1 Sources

The example fixes three versioned sources.

$S_1 =$ `StoreSQL@7` has two relations:

```
customer(customer_id, name, email, status)
orders(order_id, customer_id, total_cents)
```

Customer identifiers are local integers, the relationship is a foreign key, and status is $\{\texttt{A}, \texttt{I}, \texttt{V}\}$.

$S_2 =$ `MarketJSON@12` stores a `purchase` document with `purchaseRef`, nested `buyerRef`, `contactEmail`, `state`, and `amountUsd`. The buyer reference is marketplace-local and status is $\{\texttt{enabled}, \texttt{disabled}, \texttt{premium}\}$.

$S_3 =$ `PartnerAPI@4` exposes `Account` resources identified by URIs with a `contactEmail` field, and `Order` resources linked by a buyer URI. Lifecycle is numeric $\{1, 0, 2, 9\}$, where 9 means a partner risk hold.

All identifiers remain source-local unless a governed reconciliation decision connects them. Equal email strings do not merge customers.

7.2 Adequate domain

The proposed domain revision $D_{\text{good}} =$ `CommerceDomain@3` contains:

- a `Customer` object with fields `canonicalId`, `displayName`, `normalizedEmail`, `tier`, `lifecycle`, `riskState`, and `sourceRef`;
- an `Order` object with fields `canonicalOrderId`, `placedBy`, `amountUsd`, and `sourceRef`; and
- typed relationships and provenance back to source records.

The domain distinguishes:

$$\begin{aligned} \texttt{tier} &\in \{\texttt{standard}, \texttt{preferred}\}, \\ \texttt{lifecycle} &\in \{\texttt{active}, \texttt{inactive}\}, \\ \texttt{riskState} &\in \{\texttt{clear}, \texttt{hold}, \texttt{unknown}\}. \end{aligned}$$

These distinctions are included because at least one selected consumer uses them. The domain is bounded to commerce; it is not an enterprise-wide claim.

7.3 Consumers

$T_1 =$ `SupportView@5` requests an operational customer view with canonical identity status, display name, lifecycle, risk state, and recent orders. It requires freshness of five minutes for the store source.

$T_2 =$ `OrderFacts@8` requests analytical facts with normalized customer tier, order amount, source, and risk state. Aggregation remains a downstream query or explicit residual rather than a hidden mapping primitive.

$T_3 =$ `CustomerAPI@2` requests one nested JSON document per customer with an order array. Nesting and array multiplicity require a structural transformation beyond the current Slice 1 mapping language.

7.4 Base mappings

The mediated design proposes:

$$F_1 : S_1 \to D_{\text{good}}, \quad F_2 : S_2 \to D_{\text{good}}, \quad F_3 : S_3 \to D_{\text{good}},$$
$$G_1 : D_{\text{good}} \to T_1, \quad G_2 : D_{\text{good}} \to T_2, \quad G_3 : D_{\text{good}} \to T_3.$$

Map	Main assignments	Obligation	Provisional verdict
F_1	integer IDs remain source references; cents convert to USD; A/I/V normalize into lifecycle and tier	exact decimal policy; V does not imply identity	conditional
F_2	nested buyer path maps to Customer; string state normalizes; amount uses USD	partial buyer coverage; privacy policy	conditional/partial
F_3	URI paths map to Customer and Order; contact email and numeric lifecycle normalize; code 9 maps to risk hold	email normalization, enum authority, and URI stability	conditional
G_1	operational fields and recent-order relationship	identity status and freshness	reconciliation-dependent
G_2	order facts, tier, source, and risk state	multiplicity and analytical snapshot	conditional
G_3	customer document and nested orders	object synthesis, array policy, fresh/stable keys	unsupported in Slice 1

Table 2: Proposed mappings in the running example. No row execution has been implemented.

7.5 Dense and sparse demand

The favorable dense relation is:

$$A_{\text{dense}} = \{1, 2, 3\} \times \{1, 2, 3\}.$$

It has nine direct edges and six mediated base mappings. The comparison is not 9 versus 6 total artifacts: the domain, tests, identity policy, exceptions, generated plans, and operations still count.

The sparse relation is:

$$A_{\text{sparse}} = \{(1,1), (1,2), (2,2), (3,3)\}.$$

It has four direct edges but still requires all six base mappings if the same domain-mediated topology is used. This case must appear in any evaluation.

7.6 Pair-specific exception

The marketplace contract bars the support consumer from seeing `contactEmail`, although the analytical consumer may use an approved one-way domain token. The marketplace-to-support pair also accepts a 30-minute snapshot even though T_1's normal policy requests five minutes. These rules belong to:

$$E_{2,1} = \{\text{email suppression}, \text{30-minute freshness waiver}\}.$$

Assigning either rule solely to S_2, D, or T_1 would change other pairs. The exception therefore counts against the reuse claim.

7.7 Reconciliation-dependent identity

A marketplace buyer and a partner account have the same normalized email. This is evidence for a candidate match, not known equivalence. The example records:

$$r_{23} = (\text{possible match}, 0.78, \text{email evidence}, \text{no approving authority}).$$

Until an authorized identity decision exists, the operational consumer must display two source-local identities or mark the result reconciliation-dependent. The mapping layer may reference the candidate but cannot merge the records.

7.8 Poor domain counterexample

For contrast, let D_{poor} collapse lifecycle, tier, and risk into:

$$\texttt{customerStatus} \in \{\texttt{good}, \texttt{inactive}\}.$$

It maps `preferred` and `active` to `good`. It also maps partner risk-hold code 9 to `inactive`. Consumer T_2 needs to distinguish a risk hold from ordinary inactivity. After F_3 maps through D_{poor}, G_2 cannot recover that distinction. The pair $(3, 2)$ therefore receives a concrete lossy verdict against the direct oracle. An opaque pair-specific function would conceal the failed factorization instead of repairing it.

7.9 Change graph

The example includes five controlled changes:

1. S_2 renames `contactEmail` to `primaryEmail` without changing meaning; only F_2 should require semantic repair if locality holds.

2. T_3 adds `sourceSystem`; only G_3 should change because D_{good} already preserves `sourceRef`.

3. D_{good} splits `riskState` into policy and fraud dimensions; all three ingress maps and the consumers that use risk may change.

4. The marketplace privacy agreement changes; $E_{2,1}$ changes even though the three schemas do not.

5. An authority approves or rejects r_{23}; the shared reconciliation record and affected outputs change without changing schema composition.

The study records predicted and actual impact sets for each change. The source-local and consumer-local cases test the favorable hypothesis. The domain, exception, and identity cases measure its costs.

7.10 Serialized artifact sketch

The following sketch shows proposed interchange syntax. It is not an executable fixture:

```
{
  "factorization_id": "market-to-support-v1",
  "source_schema": "MarketJSON@12",
  "domain_schema": "CommerceDomain@3",
  "consumer_schema": "SupportView@5",
  "source_mapping": "F2@6",
  "consumer_mapping": "G1@4",
  "schema_composite": "G1@4 o F2@6",
  "data_profile": null,
  "verdict": "unsupported",
  "diagnostic": "FORWARD_OPERATOR_NOT_SELECTED",
  "exceptions": ["E21-privacy@2", "E21-freshness@1"],
  "identity_refs": ["r23@candidate"],
  "physical_plan": null
}
```

The null operator and unsupported verdict record what is missing. Claiming forward execution before the profile exists would misstate the current evidence.

8 Mapping taxonomy, composition, and factorization

8.1 Field mappings

A field mapping relates a source attribute to a target expression. Its artifact records the two declarations, scalar domains, totality, null and missing policy, conversion version, units, locale, precision, rounding, validation constraints, and provenance. A same-domain attribute projection can be exact. A cast can be partial; unit conversion can lose precision; a default can synthesize information not present at the source.

Current Rust mapping expressions are limited to a target path followed by one attribute of the same scalar type. Constants, casts, general scalar functions, multi-input expressions, nullable fields, and unit conversion are outside Slice 1.

8.2 Path mappings

A source generator may map to a typed target path:

$$f : A \to B \quad \mapsto \quad p : F(A) \rightsquigarrow F(B).$$

Current Rust code computes path substitution during schema mapping composition. That is COMPUTATIONAL for the named finite fixtures and direct-equation fragment. It does not specify how records are joined, whether path navigation is total, how multiplicity changes, or what happens when the physical source has zero or several matches.

8.3 Structural transformations

Flattening, nesting, split, merge, embedding, repeated-group construction, join, union, aggregation, pivot, and object-graph construction are different operations. They cannot be grouped under one untyped "structural map." Each operation needs:

- input and output type;
- cardinality and multiplicity behavior;
- generated-identity policy;
- constraint preservation;
- loss and error behavior;
- provenance construction; and
- an explicit exact, residual, or unsupported classification.

The G_3 nesting in the running example is therefore unsupported in the current profile even though a hand-written document builder would be easy to write. Hiding that builder inside a mapping would make the language appear closed by definition and invalidate the reuse accounting.

8.4 Enum normalization

An enum artifact includes a source code-set revision, domain code-set revision, lookup relation, totality, unknown and deprecated-code behavior, effective time, authority, and reversibility. Many-to-one normalization is a possible intentional loss. One-to-many normalization is ambiguous unless context or authority selects an alternative.

In the running example, `V`, `premium`, and numeric code 2 can map to `preferred` only under an approved business rule. An unseen code is not silently null. Partner code 9 must retain the `riskState=hold` distinction needed by T_2.

8.5 Object synthesis

Object synthesis creates a target object not already identified with a source object. It can require a deterministic key, fresh or Skolem identifier, grouping, multi-source join, collision policy, existence dependency, and provenance. Data exchange provides established semantics for fresh values and universal solutions [10]. A bounded Σ-like construction is another possible basis.

Status. OBSTRUCTED.

General synthesis is not implemented. It remains obstructed until fresh identity, quotient or canonicalization, collision, and provenance semantics are fixed. The external API's customer document cannot receive an exact verdict merely because a deterministic JSON serializer could be written.

8.6 Identity reconciliation

Mapping two source schema objects to one domain object is a schema-level assignment. Asserting that two records denote the same customer is a record-level identity decision. The latter needs evidence, confidence, authority, timestamp, ownership, contradiction handling, and reversible history.

P4 consumes identity decisions; P7 defines their broader semantics. Equal fields or vector similarity may generate candidates but cannot establish identity. The factorization record references versioned decisions and marks outputs reconciliation-dependent when authority is absent.

8.7 Information loss

A conservative loss analyzer should check object identification, arrow collapse, attribute omission, unpopulated required targets, enum collapse, precision or unit loss, cardinality reduction, filtering, aggregation, identifier replacement, null or default introduction, and provenance truncation.

The output must distinguish:

concrete loss a checked witness identifies the source distinction and consumer requirement that cannot both be preserved;

no loss under selected checks no witness was found by the named finite checks; and

unknown the analyzer lacks enough semantics to decide.

"No loss under selected checks" does not mean invertible, fully faithful, business-correct, or universally lossless. Current Rust only represents the loss envelope, so its loss fixtures are STRUCTURAL-ONLY.

8.8 Partial mappings

A partial mapping has an applicability predicate, exact coverage report, outside-domain policy, downstream completeness guarantee, and provenance. Outside cases may be rejected,

quarantined, preserved, or delegated. The policy must be explicit. Current Rust requires complete assignment of schema objects, arrows, and attributes; it does not implement record-level partial mapping.

8.9 Ambiguity

An ambiguity value retains non-equivalent alternatives and their evidence. Examples include a field with two plausible meanings, a split without a discriminant, comparable identity candidates, a context-dependent code, or multiple structurally valid paths. Array order is not authority. A compiler must either request a decision, carry the alternatives to a consumer that accepts them, or return unsupported.

8.10 Closure boundary

Composition is closed only relative to a mapping language. The current finite total path-and-attribute fragment admits some substitutions. More expressive schema-mapping languages have known closure limits [11]. CatDB must publish both positive cases and counterexamples that terminate with stable residual or unsupported diagnostics.

9 Federation, ELT, ETL, and materialization

9.1 Semantic contract versus physical plan

The semantic factorization artifact identifies meaning. A physical plan identifies where and when work occurs. The same admitted denotation might be realized through:

- virtual federation and source-native query rewriting;
- ELT into a warehouse followed by transformations;
- staged ETL into consumer tables;
- complete or partial materialization;
- incremental refresh from change streams; or
- a hybrid with source pushdown and local residuals.

These strategies have different latency, freshness, transfer, availability, retry, and operational properties. A semantic mapping does not erase those differences.

9.2 Physical plan artifact

A physical integration plan records:

1. semantic factorization and data-profile digests;
2. source connector and capability revisions;

3. query or extraction statements and typed parameters;
4. pushdown and residual boundaries;
5. transfer formats and projected fields;
6. target layout, loading, and idempotence policy;
7. snapshot, freshness, and consistency contract;
8. retries, checkpoints, backfill, and failure policy;
9. rows, bytes, calls, latency, and source load; and
10. result and provenance comparison with the semantic oracle.

The plan is versioned separately from F_i, G_j, and E_{ij}. Changing from federation to materialization need not change semantic meaning, but equivalence must be tested under one snapshot and freshness contract.

9.3 Virtual access and standards

Ontop is direct prior art for mapping-aware virtual access [3]. R2RML permits materialized or virtual RDF production from relational sources [26]; RML extends the mapping style to heterogeneous inputs [8]. P4 must compare mapping coverage, direction, composition, loss, partiality, versioning, and provenance rather than claim that a canonical virtual layer is new.

9.4 Operational cost is part of the claim

For a fair comparison, C_X includes credentials, extraction protocols, transport, staging, storage, cleaning, deduplication, scheduling, backfill, quality monitoring, retries, idempotence, loading, rate limits, physical optimization, incident handling, and freshness enforcement. Some costs may be common across conditions. They are still measured before deciding that the semantic savings matter.

If the mediated condition moves more data, introduces a central bottleneck, or misses a freshness objective, the result must report that cost. Generated ETL can be a realization of reusable semantics, but it is still ETL and still moves data.

10 Current CatDB evidence

10.1 Rust Slice 1

Status. COMPUTATIONAL.

The verified Rust command:

```
cargo run -p catdb-cli -- fixtures verify fixtures/cases \
  --implementation rust --capability-set slice-1
```

Capability	Current evidence	Boundary
Total object/arrow/attribute assignments	Rust computation	finite profile only
Arrow-to-path endpoint typing	Rust computation	declared paths only
Path-plus-attribute expressions	Rust computation	same-scalar fragment
Direct equation preservation	Rust computation	deliberately incomplete equality procedure
Identity mappings	two computed fixtures	schema mappings, not record identity
Mapping composition	six computed fixtures	bounded substitutions
Loss result representation	two decoded fixtures	structural-only; no loss computation

Table 3: Accepted prerequisite Rust evidence.

reported 24 fixtures: 16 semantic-operation passes, eight structural-only passes, and zero failures. The workspace test command also exited zero at the evidence cut.

The current mapping-composition fixtures are:

- `mapping-composition-equation-not-preserved-error`;
- `mapping-composition-equation-preserved-ok`;
- `mapping-composition-incomplete-map-error`;
- `mapping-composition-mismatch-error`;
- `mapping-composition-right-unit-ok`; and
- `mapping-composition-substitution-ok`.

Two supporting schema-identity fixtures also pass. These results establish neither source integration nor lower maintenance.

10.2 Lean Slice 3

Status. MACHINE-CHECKED.

The pinned Lean build checks:

- `MAP-03`: lawful identity mapping;
- `MAP-04`: closure of lawful mappings under composition; and
- `MAP-05`: left and right units and associativity under the stated extensional equality.

The audited build reported empty axiom inventories for the covered declarations. This label applies only to those Lean statements under their assumptions. The release record remained a release candidate with independent code review pending at the dossier cut.

The Lean results do not prove Rust conformance, parser behavior, complete equation decision, forward data integration, any Σ, Δ, or Π implementation, loss analysis, identity reconciliation, physical plans, the accounting model, or a maintenance effect.

10.3 Structural-only loss evidence

Status. STRUCTURAL-ONLY.

The current loss-envelope representation can process two fixtures:

- `information-loss-object-identification-ok`; and
- `information-loss-incomplete-mapping-error`.

These fixtures do not compute information loss. This distinction is important because the poor-domain counterexample requires a semantic witness that the current code cannot produce.

10.4 Missing systems evidence

No accepted repository artifact currently implements a source connector, domain-to-consumer planner, forward data operator, finite-instance validation, factorization checker, computed loss analyzer, partial or ambiguous mapping, enum conversion, general value expression, object synthesis, identity reconciliation, generated ETL/ELT, virtual integrated query, materialization, integration provenance, Rust/Haskell P4 comparison, or repair-surface study.

In-progress working-tree code is not paper evidence until its release record, native tests, cross-language comparisons where required, and independent code review pass.

11 Fixture and implementation plan

11.1 Freeze the profile first

The first implementation milestone is `integration-profile-v1`. It must freeze:

- schema and mapping revisions and equality;
- factorization artifact schema and verdicts;
- a bounded forward data operator and direction;
- required scalar, null, multiplicity, identifier, error, snapshot, and provenance semantics;
- exact mapping taxonomy subset;
- explicit residual and unsupported behavior; and
- canonical result comparison.

Adding connectors before this profile would create executable behavior without a stable semantic oracle.

11.2 Schema-composition fixtures

The required schema-composition fixture contracts are:

- `integration-factorization-schema-ok`: construct F_i, G_j, and normalized $G_j \circ F_i$;

- `integration-factorization-direct-equal-ok`: compare direct and factored mappings extensionally equal;
- `integration-factorization-direct-different-error`: return a stable nonfactorization witness;
- `integration-composition-version-mismatch-error`: reject mismatched domain revisions;
- `integration-composition-expression-unsupported`: reject composition outside the language;
- `integration-composition-equation-chain-ok`: preserve an equation with the adopted procedure; and
- `integration-composition-equation-unknown`: return unknown when the sound procedure is incomplete.

11.3 Direction and execution fixtures

The required direction and execution fixture contracts are:

- `integration-delta-direction-counterexample`: show that $\Delta_{G \circ F}$ consumes a target instance;
- `integration-sigma-forward-ok`: after profile acceptance, compare direct and sequential outputs;
- Before profile acceptance, `integration-sigma-forward-unsupported` must return `SIGMA_NOT_IN_PROFILE`;
- `integration-virtual-rewrite-ok`: compare canonical and rewritten query results;
- `integration-materialized-result-ok`: compare materialized output with the same oracle; and
- `integration-physical-mode-equivalence`: compare virtual and materialized snapshots at fixed freshness.

11.4 Taxonomy fixtures

The taxonomy suite includes:

- exact field projection and a partial cast;
- a multi-hop typed path;
- total enum normalization, an unmapped code, and a many-to-one loss;
- deterministic synthesis and missing fresh-identifier policy;
- an unapproved identity candidate and an approved equivalence;

- exact partial coverage and two retained ambiguous candidates; and
- required-target and provenance-truncation loss witnesses.

Every fixture names its mapping profile, result equality, expected verdict, and stable diagnostic. Positive syntax-only fixtures cannot substitute for data results.

11.5 Topology and evolution fixtures

Small deterministic manifests cover complete 2×2 and 3×3 graphs, a sparse graph with $|A| < N + M$, one source-local change, one consumer-local change, one domain-wide change, one pair-specific policy, one joint multi-source transform, one poor domain, and one staged domain-version transition.

These fixtures validate implementation behavior. They do not establish maintenance savings.

11.6 Cross-language and formal roles

Rust is the production implementation. Haskell independently computes admitted mapping, loss, partiality, and selected data-operator outcomes from the same serialized fixtures. Agreement is exact over canonical results and stable diagnostics.

Lean verifies only high-value general laws after the profile is stable. A candidate P4 theorem could state an exact factorization property for the bounded operator, but only after its semantics and assumptions are fixed. Lean is not expected to verify connectors, ETL operations, timing, governance, or human repair work.

11.7 Implementation sequence

1. freeze `integration-profile-v1`;
2. add exact factorization fixtures;
3. compute conservative loss results;
4. define partiality and ambiguity;
5. select one forward data semantics;
6. implement the independent Haskell oracle;
7. add only stable, high-value Lean statements;
8. implement one SQLite virtual path and one materialized path;
9. record provenance and physical movement; and
10. run the controlled repair-surface study.

12 Repair-surface experiment

12.1 Question and conditions

Status. ENGINEERING HYPOTHESIS.

The experiment asks:

> For which integration topologies and change distributions does a typed domain-mediated design reduce total correct-repair effort relative to pairwise, GAV, LAV, and Information-Manifold-style baselines?

The conditions are:

1. a direct pairwise mapping family;
2. a GAV-style mediated integration;
3. an LAV-style source-description and query-rewriting integration;
4. an Information-Manifold-style mediated source-description condition;
5. the CatDB typed source-domain-consumer condition.

CQL, Clio, Ontop/R2RML, and a current ETL or ELT tool may be added only when their versions, supported fragments, and tasks can be run comparably. The study must not reconstruct an unavailable system and label the reconstruction as the original.

12.2 Factorial workload

Use:

- $N, M \in \{2, 4, 8, 16\}$;
- complete, medium-density, and sparse connection graphs;
- one preregistered adequate domain and one poor-fit domain;
- mapping chain lengths $1, 2, 4, 8, 16$;
- total, partial, lossy, ambiguous, and unsupported cases;
- relational, JSON or CSV, and one API-shaped source description; and
- identical semantic requirements across conditions.

The poor-fit domain is mandatory. A workload designed only to factor cleanly cannot test the conditional boundary.

12.3 Controlled changes

The study applies the following changes independently:

1. source field rename with stable meaning;
2. source field split;
3. source enum extension;
4. source cardinality change;
5. consumer field addition;
6. consumer aggregation requirement;
7. domain concept rename with stable identity;
8. domain split affecting a subset of consumers;
9. identity-policy revision;
10. pair-specific privacy rule;
11. source replacement;
12. new source;
13. new consumer; and
14. a consumer requirement for a distinction erased by the poor domain.

Every scenario has an intended-output oracle, expected loss/partiality/ambiguity classifications, identity decisions, and expected dependency impact.

12.4 Unit of correct repair

A repair is complete only when:

- the required semantic result is restored;
- schema and mapping validation passes;
- no undeclared loss or ambiguity is introduced;
- identity decisions remain authorized;
- downstream tests pass;
- generated artifacts are current; and
- the physical strategy meets the same snapshot and freshness requirement.

A faster incorrect repair is not counted as a maintenance win.

12.5 Impact prediction

For every change c, record:

$$\text{Impact}(c) = \{\text{schemas, mappings, exceptions, queries, tests, plans,} \\ \text{materializations, policies, provenance rules}\}.$$

Compare predicted impact with the artifacts that actually require repair. Measure false-negative omissions, false-positive review burden, fan-out, transitive failures, and time to a correct green state.

12.6 Primary measurements

Primary endpoints are:

- correct repair completion and wall-clock correct-repair time;
- semantic regressions after repair;
- touched authored artifacts and changed semantic AST nodes;
- failed tests and source-specific branches;
- pair-specific exceptions and residual handwritten code;
- identity and reconciliation decisions;
- domain edits, governance decisions, and review effort;
- generated artifacts rebuilt and query-rewrite changes;
- runtime calls, rows, bytes, latency, and freshness; and
- virtual/materialized result and provenance equivalence.

Artifact count, line count, and generated-code count are secondary. None can stand alone as evidence of lower maintenance.

12.7 Domain and exception accounting

Domain costs include schema edits, semantic decision meetings, mapping migrations, compatibility windows, consumer coordination, identity-policy changes, governance approvals, and rollback or replay work. Every pair-specific exception contributes to E_{ij}, even when stored inside a generic custom-transform field.

The near-linear claim fails for a condition if these costs erase the source-side and consumer-side reuse benefit.

12.8 Correctness and statistical protocol

Before timing a repaired condition:

1. validate every schema and mapping;
2. run the same source instances;
3. compare canonical results;
4. compare loss, partiality, ambiguity, and identity decisions;
5. verify the affected dependency closure;
6. compare required provenance; and
7. exclude or separately report incorrect repairs.

The study preregisters scenario generators, exclusions, and primary endpoints. Scenario authorship is separated from repair where feasible. Condition order is randomized, scenario seeds repeat, repeated changes within a scenario are treated as dependent, and distributions with uncertainty are reported. Good-domain and poor-domain conditions remain separate. Failed runs and raw repair traces are preserved.

12.9 Rejection and narrowing rules

The central hypothesis is rejected or narrowed when:

- required mappings do not factor;
- composition leaves the language and residuals grow with $|A|$;
- direct and mediated data results disagree;
- source-local or consumer-local changes are not local;
- exceptions or identity work grow pairwise;
- domain evolution dominates savings;
- sparse topologies favor direct mappings;
- mapping counts fall without lower correct-repair work;
- excluded physical work erases the apparent benefit; or
- opaque pair code carries the semantics advertised as shared.

No experiment in this section has been run.

13 Current implementation status

P4 is OPEN at the 22 July 2026 evidence cut. The paper specifies a factorization contract, data-flow boundary, accounting model, running example, fixture plan, and controlled experiment. It does not report an implemented integration system.

The present repository has bounded prerequisite mapping evidence:

- Rust computes a finite total mapping and composition subset;
- Lean machine-checks named mapping identity, closure, unit, and associativity laws under audited assumptions; and
- structural loss-envelope values can be processed.

The repository does not yet have:

- an accepted forward data operator;
- direct-versus-mediated factorization execution;
- a P4 Rust/Haskell result comparison;
- computed loss, partiality, ambiguity, synthesis, or reconciliation;
- source connectors or compiled integration plans;
- virtual/materialized equivalence evidence;
- integration provenance; or
- a controlled correct-repair experiment.

Consequently, no $N + M$-style maintenance improvement, end-to-end integration, ETL generation, storage independence, or physical performance claim is experimentally supported.

14 Threats to validity and limitations

14.1 Conditional factorization

The approach is useful only where direct requirements factor through an adequate domain. Joint multi-source transformations, consumer-relative meaning, pair-specific policy, and poor domain design can defeat that factorization. Adding exceptions until every case appears to factor merely recreates pairwise logic behind a hub.

14.2 Topology selection

A complete $N \times M$ graph exaggerates the direct baseline when actual demand is sparse. The experiment therefore measures the real relation A, reports endpoint participation, and includes $|A| < N + M$ cases. Results from dense graphs do not generalize to sparse ones.

14.3 Domain governance

A domain schema needs ownership, review, versioning, migration, compatibility, and dispute resolution. Centralizing semantics can reduce duplication but also create a coordination bottleneck. Data mesh emphasizes bounded domain ownership and warns against a monolithic global model [6, 7]. This paper does not resolve the organizational tradeoff.

14.4 Language closure

The current mapping fragment is deliberately small. Structural transforms, casts, aggregation, synthesis, partiality, and ambiguity can leave it. Richer languages can regain closure at the cost of more complex equivalence, execution, and review. No universal closure claim is plausible.

14.5 Identity is external

Schema mappings do not decide record identity. If reconciliation work grows with source-consumer pairs or consumers use different identity authorities, C_R can erase the mediated advantage. P4 depends on, but does not solve, the P7 identity problem.

14.6 Loss analysis is incomplete

Any finite checker can miss business distinctions not represented in its types and rules. A no-loss verdict must be scoped to named criteria. Consumer requirements and human semantic review remain necessary.

14.7 Data-flow semantics are absent

The largest current limitation is not implementation scale but semantic direction. No accepted forward profile selects a Σ-like, data-exchange, query, or materialization semantics. Schema composition alone cannot supply the missing data transformation.

14.8 Human repair study

Repair-time experiments are vulnerable to learning, order, participant skill, tool familiarity, scenario quality, and review subjectivity. Randomization, repeated seeds, preregistered endpoints, preserved traces, and within-scenario dependence modeling mitigate but do not eliminate these threats.

14.9 Physical execution

Even a semantic win may lose operationally because of transfer, central bottlenecks, materialization cost, stale results, retries, source limits, or service-level objectives. Physical work stays in both cost equations. Neither category theory nor generated plans eliminate distributed-systems tradeoffs.

14.10 Prior-art comparison

Reconstructing GAV, LAV, Clio, CQL, Ontop, or an ETL tool poorly could bias the comparison toward CatDB. Executable baselines must use pinned supported fragments. Historical systems that cannot be run should remain conceptual comparators rather than mislabeled implementations.

15 Open and obstructed questions

15.1 Open questions

1. Should D be one bounded-domain schema, a federation of domains, or an enterprise model?
2. Which extensional equality decides schema and data factorization?
3. Which forward operator gives the smallest useful closed profile?
4. How should pair-specific residuals be represented and versioned?
5. Which constraints compose, and which require snapshot validation?
6. How can domain evolution avoid global repair?
7. Which identity decisions can be shared across consumers?
8. How should partial coverage and ambiguity appear in consumer APIs?
9. What provenance is required for a composed result?
10. Which prior systems can be executed fairly as baselines?
11. Does correct-repair effort improve in the preregistered adequate-domain condition?
12. When does a sparse topology justify direct mappings instead?
13. How should a consumer-required distinction be fed back into domain governance without turning D into a union of pair fields?
14. Can physical virtual and materialized plans share one result oracle under explicit freshness?

15.2 Obstructed questions

Forward execution is obstructed until one operator and its canonicalization are fixed. General synthesis is obstructed by fresh identity, quotient, and provenance semantics. General composition is obstructed until the mapping language and closure boundary are fixed. Virtual/materialized equivalence is obstructed until query, null, multiplicity, snapshot, and freshness profiles exist.

A universal $N + M$ theorem is obstructed in principle by sparse topologies, poor domains, and pair-specific semantics. P4 should not pursue one.

16 Conclusion

Domain-mediated mappings can reduce duplicated semantic definitions, but the conditions are demanding. Every required source-consumer transformation must factor through an adequate domain. The mapping language and selected data operator must remain closed over the required composition, while source and consumer rules remain separable. Exceptions, identity, loss, ambiguity, governance, and physical work must stay in the accounting.

The actual direct baseline is $|A|$, not automatically NM. The mediated base count is $|I_A| + |J_A|$, not literally $N + M$ once domain, reconciliation, tests, exceptions, and operations are included. Dense topologies and local changes may favor the mediated design. Sparse topologies, poor domains, pair-specific policy, and reconciliation-heavy work may favor direct mappings or narrow the claim.

The data-flow boundary adds a separate obligation. Schema mappings $S_i \to D \to T_j$ compose at the schema level, but they do not identify a forward integration operator. In particular, Δ is contravariant. A future CatDB implementation must choose and version a bounded forward construction, data-exchange, query, or materialization semantics, then compare direct and sequential results.

Current evidence establishes only a prerequisite subset: bounded Rust mapping composition and selected Lean mapping laws. No P4 integration path or repair-surface result exists. A defensible next milestone is one complete factorization fixture with a selected forward operator, independent Rust/Haskell agreement, conservative loss and identity status, and equivalent virtual and materialized results. A controlled repair experiment must then test whether fewer mappings produce less correct work.

The intended claim remains narrow: CatDB may move some integration meaning from bespoke pair pipelines into typed, composable, versioned mappings. Federation, ELT, ETL, and materialization remain execution strategies, and data movement and operational tradeoffs remain.

A Formal dependency graph

For one mediated family, define authored semantic vertices:

$$V_{\text{auth}} = \{D, R\} \cup \{F_i \mid i \in I_A\} \cup \{G_j \mid j \in J_A\} \cup \{E_{ij} \mid (i, j) \in A\}.$$

Generated vertices include composites, tests, query rewrites, plans, and materializations. A directed edge $u \to v$ means that changing u can invalidate v. The predicted impact set of change c is the transitive closure from the directly changed vertices, filtered by versioned dependency conditions.

The study compares that prediction with the minimal actual repair set needed to restore correctness. A false negative omits an artifact that needed repair. A false positive forces review of an artifact whose semantics did not change. Both count against the maintenance claim.

B Factorization acceptance checklist

For each $(i, j) \in A$, acceptance requires:

1. exact source, domain, and consumer revisions;
2. mapping-language and equality-profile revisions;
3. normalized $H_{ij} = G_j \circ F_i$;
4. selected instance/data operator and direction;
5. direct oracle or extensional comparison target;
6. closure, residual, or unsupported verdict;
7. typed path, attribute, equation, and constraint evidence;
8. loss, partiality, synthesis, and ambiguity records;
9. governed identity references;
10. pair-specific exceptions;
11. separate physical plan; and
12. result and provenance comparison at one snapshot.

Missing items produce an incomplete or unsupported artifact rather than an implicit successful factorization.

C Claim-promotion checklist

Table 4: Present status and promotion evidence.

Claim	Present status	Required promotion evidence
Base mapping count under factorization	proved	assumptions remain explicit
Sparse topology can have $\|A\| < N + M$	proved	elementary counterexample
Conditional authored source locality	proved	retain strong dependency premises
Selected mapping laws	machine-checked prerequisite	exact Lean declarations and coverage record
Rust mapping composition subset	computational prerequisite	named fixture command and release evidence
Loss analysis	structural-only	computed witnesses and cross-language checks

Claim	Present status	Required promotion evidence
Forward integration	obstructed	accepted operator profile and result equivalence
Factorization checker	open	direct-versus-mediated fixture suite
Virtual/materialized parity	open	same oracle, snapshot, freshness, and provenance
Near-linear maintenance	engineering hypothesis	controlled correct-repair experiment
ETL or movement elimination	nonclaim	no promotion path; excluded by scope

References

[1] Philip A. Bernstein and Sergey Melnik. Model management 2.0: Manipulating richer mappings. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 1–12, 2007.

[2] Kristopher S. Brown, David I. Spivak, and Ryan Wisnesky. Categorical data integration for computational science. *Computational Materials Science*, 164:127–132, 2019.

[3] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. Ontop: Answering sparql queries over relational databases. *Semantic Web*, 8(3):471–487, 2017.

[4] CategoricalData project. Categorical query language (CQL) documentation and tutorial. Official project documentation, 2026. Accessed 2026-07-22.

[5] dbt Labs. dbt semantic layer. Official product documentation, 2026. Accessed 2026-07-22.

[6] Zhamak Dehghani. How to move beyond a monolithic data lake to a distributed data mesh. MartinFowler.com, 2019. Primary practitioner architecture essay; accessed 2026-07-22.

[7] Zhamak Dehghani. Data mesh principles and logical architecture. MartinFowler.com, 2020. Primary practitioner architecture essay; accessed 2026-07-22.

[8] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. Rml: A generic language for integrated rdf mappings of heterogeneous data. In *Proceedings of the 7th Workshop on Linked Data on the Web*, volume 1184 of *CEUR Workshop Proceedings*, 2014.

[9] Ronald Fagin, Laura M. Haas, Mauricio A. Hernández, Renée J. Miller, Lucian Popa, and Yannis Velegrakis. Clio: Schema mapping creation and data exchange. In *Conceptual Modeling: Foundations and Applications*, volume 5600 of *Lecture Notes in Computer Science*, pages 198–236. Springer, 2009.

[10] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[11] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang-Chiew Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Transactions on Database Systems*, 30(4):994–1055, 2005.

[12] Google Cloud. Introduction to lookml. Official product documentation, 2026. Accessed 2026-07-22.

[13] Google Cloud. What is elt (extract, load, and transform)? Official product documentation, 2026. Accessed 2026-07-22.

[14] Laura M. Haas, Mauricio A. Hernández, Howard Ho, Lucian Popa, and Mary Roth. Clio grows up: From research prototype to industrial tool. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, 2005.

[15] Thomas Kirk, Alon Y. Levy, Yehoshua Sagiv, and Divesh Srivastava. The information manifold. In *Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, number SS-95-08 in AAAI Technical Report, pages 85–91, 1995.

[16] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 233–246, 2002.

[17] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 49–58, 2001.

[18] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[19] Patrick Schultz, David I. Spivak, Christina Vasilakopoulou, and Ryan Wisnesky. Algebraic databases. *Theory and Applications of Categories*, 32(16):547–619, 2017.

[20] Patrick Schultz, David I. Spivak, and Ryan Wisnesky. Algebraic model management: A survey. In *Recent Trends in Algebraic Development Techniques*, volume 10644 of *Lecture Notes in Computer Science*, pages 56–69. Springer, 2017.

[21] Patrick Schultz and Ryan Wisnesky. Algebraic data integration. *Journal of Functional Programming*, 27:e24, 2017.

[22] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[23] David I. Spivak. Functorial data migration. *Information and Computation*, 217:31–51, 2012.

[24] David I. Spivak and Ryan Wisnesky. Relational foundations for functorial data migration. In *Proceedings of the 15th Symposium on Database Programming Languages*, pages 21–28, 2015.

[25] Panos Vassiliadis, Zografoula Vagena, Spiros Skiadopoulos, Nikos Karayannidis, and Timos Sellis. Arktos: Towards the modeling, design, control and execution of etl processes. *Information Systems*, 26(8):537–561, 2001.

[26] W3C RDB2RDF Working Group. R2rml: Rdb to rdf mapping language. W3c recommendation, World Wide Web Consortium, 2012.

[27] Gio Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25(3):38–49, 1992.

[28] Ryan Wisnesky, David I. Spivak, Patrick Schultz, and Eswaran Subrahmanian. Functorial data migration: From theory to practice, 2015.