Semantic Query Federation over Heterogeneous Databases

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

23 July 2026

Abstract

A shared catalog can tell a query engine where tables live. It cannot, by itself, tell the engine whether two source fields mean the same thing, whether a source is complete, or whether a remote comparison has the same collation and null behavior as the canonical query. Those gaps matter whenever one answer draws on several live systems. We study a provisional CatDB contract for this harder problem: typed, mapping-aware federation rather than syntactic catalog federation.

The contract begins by separating a schema mapping from the instructions for reading data through it. Each source declares a Global-as-View (GAV), Local-as-View (LAV), Generalized Local-as-View (GLAV), or another bounded read profile. Coverage, authority, connector behavior, scalar and bag semantics, snapshot identity, failure policy, and reconciliation are part of that profile. A plan may push an operator only when the remote operation is exact or a named local residual repairs the difference. The result says whether the answer is exact, certain, partial, approximate, stale, or failed. A required source failure never appears as an empty complete result.

A customer-and-order example spans PostgreSQL, SQLite, and an immutable enrichment file. It exposes a collation mismatch, a cross-source join, overlapping coverage, an identity decision, and three availability policies. The example is hand worked. It is not evidence of a running backend. We use it to state rewrite, coverage, capability, join-placement, snapshot, reconciliation, and provenance obligations, together with falsification fixtures and an evaluation plan against native queries, R*-style distributed planning, Calcite, Trino, Ontop, and a polystore baseline.

The present result is a research contract, not an implemented federation system. The repository has bounded schema and mapping prerequisites, but no accepted query profile, read-mapping semantics, source selector, federation planner, connector executor, result envelope, heterogeneous differential run, or P5 performance measurement. All CatDB execution and performance claims therefore remain OPEN or OBSTRUCTED. The candidate contribution is the combined, versioned obligation layer. Mediation, view-

based rewriting, source capabilities, pushdown, distributed optimization, ontology-based access, SPARQL federation, and polystore execution are established prior art.

Contents

1 Introduction

1.1 The problem in plain language

An analyst asks for July orders over $100 from customers whose canonical name is "Ångström Labs," together with a support tier. Customer records live in PostgreSQL, orders in SQLite, and support tiers in a small signed JSON export. The sources disagree about identifiers, letter case, Unicode normalization, nulls, number encodings, and observation time. One customer also appears under two source identifiers. It is straightforward to send three requests and join the returned rows. The hard part is deciding what that joined answer means.

A conventional catalog can register the sources and their columns. A federated engine can push a filter, pull intermediate rows, and choose a join site. Mature systems already do those things. The unresolved question for CatDB is narrower: can a versioned semantic mapping state enough about direction, coverage, loss, authority, and source behavior that every physical choice carries a checkable answer contract? The contract must still work when one comparison remains local, one source is incomplete, or one required source fails.

This is the difference between typed mapping-aware federation and syntactic catalog federation. A syntactic catalog knows that `crm.customers.name` is text and that a connector accepts a predicate. A mapping-aware plan also needs the canonical attribute, the rows the source covers, the comparison semantics, and the rule for combining assertions from another source. The metadata does not remove any physical work. It limits which plans may claim a particular meaning.

1.2 Research question and bounded thesis

The research question is:

> How can a canonical query be rewritten through typed, versioned mappings into source-native and local plan stages without hiding semantic mismatch, incomplete coverage, snapshot skew, reconciliation, or source failure?

The bounded thesis is an ENGINEERING HYPOTHESIS:

> Given a fixed canonical-query semantics, explicit read-direction mappings, declared source coverage and authority, connector-version-specific semantic capabilities, and explicit reconciliation, snapshot, movement, and failure policies, CatDB can attempt a typed rewrite. Exactness is a per-plan obligation. When it cannot be established, the planner rejects the plan or labels approximation, incompleteness, staleness, and missing contributions in the result.

The verb "attempt" matters. A valid schema functor does not imply that a canonical query can be answered from a source instance. A connector that accepts SQL syntax does not imply an exact pushdown. A cheap join plan does not become admissible if it violates collation, residency, snapshot, or authority policy.

1.3 Contributions of this provisional paper

This paper makes no systems-result claim. Its contributions are author-side research artifacts:

1. a canonical query bundle and result-envelope contract that keeps mapping, source, capability, policy, snapshot, failure, reconciliation, and provenance revisions visible;
2. a direction discipline that distinguishes schema mappings from GAV, LAV, GLAV, and other read interpretations;
3. operator-level dispositions for exact pushdown, exact local residual, approved approximation, and unsupported behavior;
4. paper proofs and counterexamples delimiting what follows from those definitions, without claiming connector correctness;
5. a concrete heterogeneous example that covers exact, partial, failed, and stale-fallback outcomes;
6. a current implementation audit with every P5 execution claim marked OPEN or OBSTRUCTED; and
7. a fixture, falsification, baseline, and reproducibility plan for a future executable result.

1.4 Nonclaims

This paper does not claim that category theory solves distributed query optimization; that mappings are generally invertible; that every source is complete; that all pushdown is exact or beneficial; or that federation eliminates data movement. It does not identify a vector of source snapshots with one global transaction. It does not treat duplicate removal as identity reconciliation, vector similarity as identity, or returned values as lineage. It does not claim universal support across SQL, API, file, graph, document, or vector systems. It does not claim native performance, lower transfer volume, fewer calls, or exclusion of CatDB from a runtime hot path.

1.5 Paper organization

Section 2 fixes the evidence vocabulary. Section 3 positions the contract against federation, mediation, view rewriting, distributed optimization, OBDA, SPARQL federation, and polystores. Section 4 gives the running example. Section 5 defines the query and result semantics. Sections 6 to 8 cover mapping direction, source selection, and pushdown. Sections 9 to 12 cover placement, source states, reconciliation, provenance, and heterogeneous connectors. The remaining sections audit present evidence, specify the implementation and experiment, state limitations, and list open questions.

2 Evidence vocabulary and current status

2.1 Two independent status dimensions

The paper uses an evidence class and an operational status. A source-state word such as `PROPOSED` in an earlier design record does not promote a claim. Evidence classes are:

- PROVED for a complete paper derivation within stated assumptions;
- MACHINE-CHECKED for a named statement accepted by a pinned proof kernel with its assumption audit;
- EXPERIMENTALLY SUPPORTED for reviewed, reproducible measurements of the exact claim;
- SUPPORTED BY PRIOR LITERATURE for a scoped statement established by a cited primary source or official specification;
- ENGINEERING HYPOTHESIS for a precise proposed behavior with a falsification test;
- OPEN CONJECTURE for a precise unresolved mathematical claim; and
- SPECULATIVE for an idea whose validation path is not yet specific.

Operational status is one of PROVED, CONDITIONAL, COMPUTATIONAL, OBSTRUCTED, or OPEN. A claim can be SUPPORTED BY PRIOR LITERATURE and only CONDITIONAL for CatDB because literature about another system does not implement this one. An implementation can be COMPUTATIONAL without being a theorem.

2.2 Evidence cut

The evidence cut is 23 July 2026. The current workspace contains Rust crates for core values, semantic IR, schema validation, mapping validation and composition, finite migration, and a CLI. It also contains a Haskell reference project and selected Lean schema, path, and mapping declarations. The accepted P1 evidence is a prerequisite only.

The workspace does not contain `catdb-query`, `catdb-connector`, `catdb-planner`, `catdb-storage-sqlite`, or `catdb-storage-postgres`. The fixture contract explicitly excludes query-rewrite fixtures. No accepted `query-profile-v1` or `read-mapping-profile-v1` file exists. No P5 result, call trace, byte trace, source-native comparison, failure envelope, or benchmark is present.

2.3 What this paper can prove

The paper can prove consequences of its definitions and construct counterexamples. For example, one schema mapping can support distinct read interpretations, so the mapping alone cannot determine a unique answer. Likewise, if "exact" is defined to exclude required-source failure, a failed required source cannot honestly yield exact status. These paper results do not prove that a future compiler, connector, database, API, or telemetry system follows the definitions.

Table 1: Current claim status at the manuscript evidence cut.

Claim	Present evidence	Operational status	Boundary
Schema/path/mapping prerequisite	Bounded Rust/Haskell agreement and selected Lean laws from P1	COMPUTATIONAL CONDITIONAL	No query, source, or connector semantics
Typed canonical query	No accepted query profile or checker	OBSTRUCTED	P2 semantic gate is missing
Rewrite through mappings	Schema mappings exist; read interpretation does not	OBSTRUCTED	P4 direction gate is missing
Sound source selection	No coverage model or selector	OBSTRUCTED	Completeness cannot be stated
Exact operator pushdown	No connector or conformance record	OPEN	Syntax acceptance is insufficient
Cross-source plan execution	No planner or executor	OPEN	No backend claim
Partial and stale outcomes	Paper contract only	OBSTRUCTED	No result/failure envelope
Federated provenance	No P5 provenance path	OPEN	P13 dependency remains
Transfer, calls, latency, or scale	No P5 execution or measurement	OPEN	No performance claim is admissible

3 Prior art and candidate CatDB delta

3.1 Federated databases and mediators

Federated database systems have long addressed distributed, heterogeneous, and autonomous sources [22]. Mediator architectures separate a higher-level information interface from underlying sources [31]. Information Manifold used a mediated domain model and source descriptions to select sources and answer high-level queries [15]. TSIMMIS supplied a common data model, mediator query language, and wrappers for heterogeneous information [10].

These systems make "one query over many sources" an established problem, not a CatDB invention. They also make source selection, wrapper behavior, and semantic heterogeneity direct baselines rather than implementation details to omit.

3.2 Source descriptions and query answering using views

Levy, Rajaraman, and Ordille describe source contents and access restrictions declaratively, then use those descriptions for source pruning and multi-source plans [17]. Vassalos and Papakonstantinou formalize heterogeneous-source query capabilities and mediator decomposition [27]. The GAV/LAV/GLAV vocabulary, query answering using views, and exact versus contained rewriting distinctions are mature [16, 13]. MiniCon is a direct algorithmic baseline for contained conjunctive-query rewriting [20]. Data-exchange work supplies universal-solution and certain-answer foundations for dependency-based settings [9].

Consequently, a typed mapping is not novel merely because it describes a source relative to a canonical schema. P5 must identify the admitted query and mapping fragment, the intended answer relation, and the additional version, failure, reconciliation, and provenance obligations.

3.3 Garlic and capability-aware optimization

Garlic is close prior art for middleware over diverse sources, source-specific modules, wrapper-exposed capabilities, and native source exploitation [6]. Its optimizer combines rule and cost information across sources [12], and capabilities-based rewriting constructs plans whose component requests respect source abilities [19]. Later Garlic work embeds that federation model in DB2 [14].

The candidate CatDB delta cannot be "connectors declare capabilities" or "the optimizer pushes work." It must be a narrower, demonstrated relationship among typed mapping obligations, semantic capability profiles, versioned result status, reconciliation, and provenance.

3.4 Distributed planning and adaptable execution

Classical optimizers already treat access paths and join order as physical choices [21]. R* evaluates distributed execution sites, shipping, fetch-matches, semijoins, and communication estimates [18]. Volcano supplies extensible and parallel operator mechanisms [11]; Eddies demonstrates adaptive query routing [3]. Query shipping, data shipping, bind joins, semijoin reduction, and cost-based site selection are therefore non-novel.

3.5 Modern adapter and connector systems

Apache Calcite provides a modular relational algebra, adapter conventions, rules, and cost-based optimization over heterogeneous sources [4, 2]. Trino documents connector-specific predicate, projection, aggregation, join, limit, and Top-N pushdown, and recommends plan inspection to verify whether pushdown occurred [26, 24]. Its PostgreSQL documentation also records collation-sensitive cases where remote string predicates are not pushed [25]. This is a useful precedent for semantic capabilities rather than backend-wide booleans.

3.6 OBDA, SPARQL federation, and polystores

Ontop is direct prior art for mapping-based virtual semantic access and SPARQL-to-SQL rewriting over relational sources [5]. R2RML standardizes relational-to-RDF mappings [28]. SPARQL 1.1 defines a multiset query algebra [30], while its federated-query recommendation specifies `SERVICE` and `SERVICE SILENT` behavior [29]. BigDAWG spans heterogeneous engines using islands, shims, and casts [8]. CatDB cannot claim novelty for virtual access, remote graph patterns, failure-sensitive remote algebra, or cross-engine movement.

3.7 Portable physical representations and verified rewrites

Arrow gives a standardized typed columnar memory layout [1]; Substrait supplies a portable plan specification [23]. HoTTSQL demonstrates that real SQL rewrite proofs can be mechanized for a declared semantics [7]. These sources constrain two further claims. A typed interchange format is not itself a semantic proof, and an abstract rewrite theorem does not verify a connector or database execution.

3.8 Candidate contribution

The candidate contribution is the combined contract:

1. immutable canonical and source schema revisions;
2. versioned read-direction mappings with coverage, loss, and authority;
3. operator-level connector capabilities with semantic conformance;
4. plans retaining residual, movement, failure, snapshot, and policy obligations;
5. typed reconciliation that preserves uncertainty;
6. a result envelope linking values to mappings, policies, snapshots, plans, failures, and provenance; and
7. one semantic intent whose physical realization may be federated, materialized, or hybrid.

No individual item is claimed as new. Whether the combination supplies a measurable systems contribution remains OPEN.

4 Running example: customers, orders, and support tiers

4.1 Canonical query

The canonical schema revision D^{17} has three relations:

$$\begin{aligned}\mathsf{Customer}(&cid : \mathsf{CustomerId}, name : \mathsf{UnicodeText}, region : \mathsf{Region}),\\ \mathsf{Order}(&oid : \mathsf{OrderId}, customer : \mathsf{CustomerId}, total : \mathsf{Cents}, placedAt : \mathsf{Instant}),\\ \mathsf{Tier}(&customer : \mathsf{CustomerId}, tier : \mathsf{TierCode}, observedAt : \mathsf{Instant}).\end{aligned}$$

The paper query Q_{July} asks for each July 2026 order over 10,000 cents whose customer's canonical case-folded name equals `ångström labs`, together with the customer's support tier. It orders by `total` descending and `oid` ascending and returns at most ten rows. The tier is required in the strict profile. In the explicit-partial profile, the order contribution may be returned without tier values only if the result envelope names the missing contribution.

Listing 1: Illustrative canonical syntax. No parser or executor currently accepts this text.

```
FROM Customer c
JOIN Order o ON o.customer = c.cid
REQUIRE Tier t ON t.customer = c.cid
WHERE canonical_name_token(c.name) = "angstrom-labs-v1"
  AND o.total > cents(10000)
  AND o.placedAt >= instant("2026-07-01T00:00:00Z")
  AND o.placedAt < instant("2026-08-01T00:00:00Z")
SELECT c.cid, c.name, o.oid, o.total, t.tier
ORDER BY o.total DESC, o.oid ASC
LIMIT 10
```

The listing is reader-facing notation, not an accepted grammar. The semantic query must be represented by one immutable typed IR before an executable P5 claim can begin.

4.2 Three source descriptions

PostgreSQL source S_P stores customers. Its local key is an integer, names use a database collation that does not certify the canonical Unicode case-folding relation, and region may be null. SQLite source S_O stores orders. Customer references are text such as `cust/17`, totals are integer cents, and timestamps are UTC text in the admitted fixture profile. An immutable JSON file S_T supplies support tiers with identifiers such as `crm-017`. Its content digest is its snapshot identity.

The first name uses an awkward encoding on purpose. The PostgreSQL profile does not certify that remote case folding matches the canonical comparison. The source may apply a safe coarse filter, but the final name test stays local. We leave the transfer byte count blank because no connector has measured it.

Table 2: Hand-worked source rows. These are paper data, not connector output.

Source	Local ID	Name or customer ref.	Value	Observation
PostgreSQL	17	decomposed Unicode form of Ångström Labs	region `CR`	transaction snapshot σ_P
PostgreSQL	18	`Angstrom Labs`	region `US`	same σ_P
SQLite	`o-501`	`cust/17`	12,500 cents	4 Jul, 16:30 UTC
SQLite	`o-502`	`cust/17`	8,000 cents	5 Jul, 09:00 UTC
SQLite	`o-503`	`cust/18`	20,000 cents	6 Jul, 12:00 UTC
JSON tier	`crm-017`	governed link candidate to customer 17	`gold`	digest σ_T

4.3 Read interpretations

The first executable target uses closed GAV definitions:

$$\begin{aligned}
\mathsf{Customer}(\mathsf{cid}(p.id), \mathsf{nfc}(p.name), p.region) &\leftarrow S_P.\mathsf{customers}(p),\\
\mathsf{Order}(o.oid, \mathsf{cidFromRef}(o.customerRef), o.cents, \mathsf{parseUTC}(o.time)) &\leftarrow S_O.\mathsf{orders}(o),\\
\mathsf{Tier}(\mathsf{approvedLink}(t.customerRef), t.tier, \mathsf{parseUTC}(t.observedAt)) &\leftarrow S_T.\mathsf{tiers}(t).
\end{aligned}$$

Each function is part of the versioned read profile. In particular, approvedLink is not string parsing or duplicate elimination. It consults an authority-approved reconciliation artifact. If no approved link exists, the tier row remains unresolved.

The customer source claims complete coverage for active customers in regions CR and US at σ_P. The order source claims complete July coverage for the represented store at σ_O, but not for marketplace orders. The tier file claims complete support-tier coverage only for the identifiers listed in its signed export. A result may therefore be exact only relative to the declared store-and-tier coverage domain. It is not an enterprise-wide customer-order answer.

4.4 Illustrative typed plan

Figure 1 shows a legal candidate plan if all capability and policy checks succeed. The diagram describes placement; it does not report an executed backend plan.

4.5 Required observability

The future fixture trace must expose the fields in Table 3. Hand-counted paper rows can define a golden logical result. Only the executor may supply observed calls, bytes, retries, and timings. A blank value is preferable to a fabricated measurement.

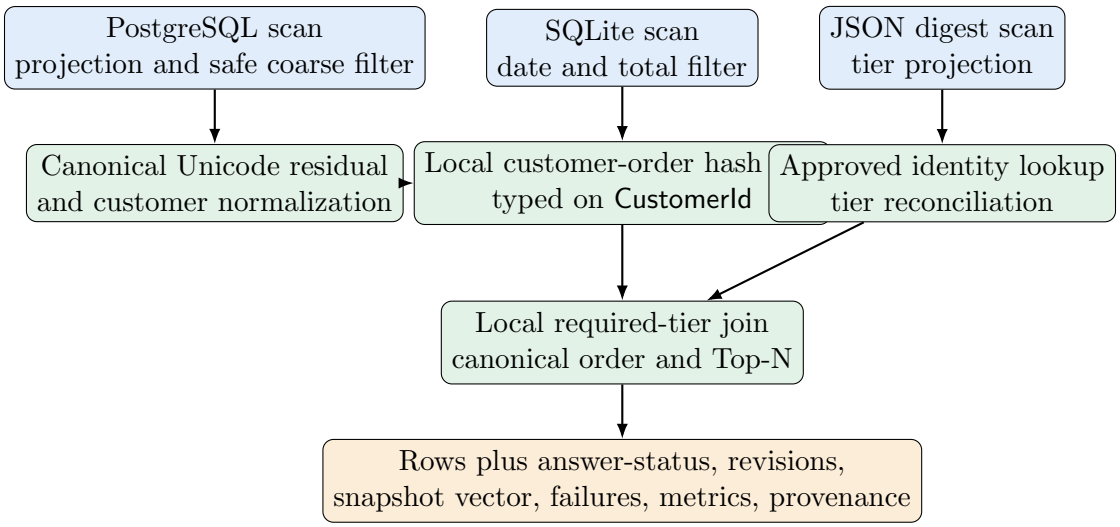

Figure 1: Candidate placement for the paper example. Every edge must carry a typed row schema, multiplicity, ordering claim, source/snapshot identity, and observed metrics when execution exists.

Table 3: Observation contract for the running example.

Stage	Placement	Required observations	Present status
Customer scan	PostgreSQL	calls, rows scanned/returned, request/response bytes, remote plan, snapshot	OPEN; no connector
Order scan	SQLite	calls, rows scanned/returned, bytes, native plan, snapshot	OPEN; no connector
Tier scan	immutable file	opens/calls, bytes, parsed/rejected rows, content digest	OPEN; no connector
Residual	CatDB local	input/output rows, CPU, memory, canonical predicate revision	OPEN; no executor
Cross-source joins	CatDB local in this candidate	input/output rows, hash bytes, equality profile, reconciliation decisions	OPEN; no planner
Result envelope	CatDB result boundary	status, missing contributions, snapshots, plan digest, provenance reference	OBSTRUCTED; no profile

4.6 Three failure outcomes

When the tier file is required but unavailable, `strict` returns an `error`. It cannot return the order rows with an exact label. `explicit_partial` may return the customer-order contribution, but the envelope must identify the missing tier and the resulting coverage gap. `declared_stale_fallback` may use a materialized tier export only when the result carries its identity, source snapshot, age, and policy authorization. The source rows may be the same in all three cases, but the failure contracts make the operational results different.

5 Formal query and result semantics

5.1 Canonical query profile

Definition 5.1 (Canonical query profile)**.** A canonical query profile $\mathcal{S}$ fixes the admitted typed operators and their denotation, including bag or set multiplicity, scalar domains, null and missing values, equality, collation, numeric and temporal rules, error behavior, ordering, and snapshot interpretation.

The first profile should be closed and deliberately small. Scans, selections over pure predicates, projections, inner equijoins, and an exact ordered limit are plausible. Aggregation, outer joins, volatile functions, floating point, and open-ended source functions should remain unsupported until separately specified.

Definition 5.2 (Read interpretation)**.** For source schema S_k, canonical schema D, and source state W_k, a versioned read interpretation is

$$\mathcal{R}_k = (S_k, D, M_k, \delta_k, \gamma_k, \alpha_k, \kappa_k),$$

where M_k is a schema mapping or dependency artifact, δ_k is a GAV, LAV, GLAV, or separately defined read direction, γ_k is coverage, α_k is authority and identity policy, and κ_k contains scalar conversion, multiplicity, null, collation, time, and error semantics.

Definition 5.3 (Canonical query bundle)**.** The complete planner input is

$$\mathcal{Q} = (Q_D, D, \mathcal{M}, \mathcal{S}, \mathcal{C}, \mathcal{P}, \mathcal{F}, \mathcal{R}), \tag{1}$$

where Q_D is the typed query, D the canonical revision, $\mathcal{M}$ the read interpretations, $\mathcal{S}$ the query semantics, $\mathcal{C}$ the connector capabilities, $\mathcal{P}$ the movement and authorization policy, $\mathcal{F}$ the availability policy, and $\mathcal{R}$ the reconciliation contract.

No element of Equation (1) may be silently supplied by a connector. Credentials can be referenced separately, but any setting that changes denotation or admissibility belongs in versioned plan identity.

5.2 Physical plan and obligations

The planner returns

$$(\mathcal{P}_h, \mathcal{O}) = \mathsf{Plan}(\mathsf{Rewrite}(\mathcal{Q})),$$

where $\mathcal{P}_h$ is a typed physical DAG and $\mathcal{O}$ is an obligation set. Nodes include source-native queries, API requests, file scans, exchanges, materializations, bounded local residuals, joins, reconciliation, normalization, provenance attachment, and explicit failure-policy branches.

Table 4: Obligations retained by a federation plan.

ID	Required question
`O-TYPE`	Are canonical, source, exchange, residual, and result fields well typed?
`O-MAP`	Does the rewrite implement the declared read interpretation?
`O-COVER`	Does the selected source set justify the requested completeness label?
`O-CAP`	Is every remote fragment exact under the connector, backend, version, configuration, and session profile?
`O-RESIDUAL`	Does local residual evaluation restore the canonical denotation within a declared resource bound?
`O-JOIN`	Does placement preserve multiplicity, equality, null, collation, and snapshot semantics?
`O-RECONCILE`	Are normalization, identity, authority, and conflict decisions explicit and versioned?
`O-SNAPSHOT`	Which source states contributed, and what cross-source consistency relation, if any, is established?
`O-FAILURE`	What happens on timeout, malformed rows, source errors, cancellation, and partial output?
`O-PROV`	Can the result be traced to sources, mappings, policies, plan stages, and snapshots at the promised granularity?
`O-POLICY`	Are residency, authorization, source-load, freshness, and movement constraints satisfied?

5.3 Exact closed-world correctness

Let $\mathsf{Build}_{\mathcal{M}}(\vec{W})$ be the canonical instance determined by the read interpretations over a declared source-world vector $\vec{W}$. Let $[\![Q_D]\!]_I$ be the canonical denotation on instance I, and let $\mathsf{Norm}_{\mathcal{S}}$ normalize physical encodings without erasing semantic distinctions. Exactness requires

$$\mathsf{Norm}_{\mathcal{S}}\left(\mathsf{Exec}(\mathcal{P}_h, \vec{W}, \vec{\sigma})\right) = [\![Q_D]\!]_{\mathsf{Build}_{\mathcal{M}}(\vec{W})}\,. \tag{FED-EXACT}$$

Equality is multiset equality unless $\mathcal{S}$ explicitly selects set semantics. Order is compared only when the canonical query declares one. Generated SQL parsing, remote acceptance, expected rows in one example, or reduced transfer does not establish Equation (FED-EXACT).

5.4 Incomplete sources and certain answers

For incomplete source descriptions, let $\mathsf{Legal}_{\mathcal{M}}(\vec{W})$ be the canonical worlds consistent with the read theory. In an admitted set-valued conjunctive-query fragment,

$$\mathsf{cert}(Q_D, \mathcal{M}, \vec{W}) = \bigcap_{I \in \mathsf{Legal}_{\mathcal{M}}(\vec{W})} [\![Q_D]\!]_I \,. \tag{2}$$

The result may be `certain_complete` if it equals Equation (2), or `certain_sound` if it is a subset. Bag-valued certain answers, aggregates, negation, inequalities, nulls, and nonmonotone operators require separate semantics. This paper does not generalize Equation (2) to them. In particular, the intersection in Equation (2) is the standard set-valued possible-world construction. Bag multiplicities do not inherit that construction without an additional information order over multiplicities: taking a set intersection erases counts, while taking pointwise minimum counts is a separate semantic choice that this profile has not justified. The initial certain-answer fragment therefore remains set-valued even though closed-world exactness in Equation (FED-EXACT) may use bags.

5.5 Answer status and envelope

Definition 5.4 (Answer status)**.** An answer status is one of `exact`, `certain_complete`, `certain_sound`, `partial`, `approximate`, `stale`, or `error`. Each status has a profile-specific admission predicate over discharged obligations, source outcomes, snapshots, and reconciliation.

Every future executable result must contain at least:

1. query-profile and canonical-schema revisions;
2. all read-mapping revisions and direction profiles;
3. source-description and capability revisions;
4. logical and physical plan identities;
5. per-source snapshot or immutable-content identity;
6. answer status, coverage, and missing contributions;
7. failures, retry observations, and fallbacks;
8. observed calls, rows, bytes, timings, and residual work;
9. reconciliation revision and decisions; and
10. a result- or row-level provenance reference under the P13 contract.

Proposition 5.5 (Strict failure exclusion)**.** *Under a profile in which a source contribution is required and* `strict` *defines exact admission as successful observation of every required contribution, failure of that source is incompatible with* `exact`*.*

Proof. Let K_{req} be the required source set and K_{ok} the sources successfully observed at admitted snapshots. By the profile definition, exact admission implies $K_{\text{req}} \subseteq K_{\text{ok}}$. If a required source $k \in K_{\text{req}}$ fails, then $k \notin K_{\text{ok}}$, so the inclusion is false. The execution may return `error`, or another policy may admit `partial` or `stale`; it cannot satisfy this exact predicate. □

Status. PROVED for the stated profile definition; connector compliance and failure detection remain OPEN.

6 Mapping direction: GAV, LAV, GLAV, and bounded profiles

6.1 A schema arrow is not an access program

A schema mapping

$$F_k : S_k \longrightarrow D$$

assigns source objects and arrows to canonical objects and paths while preserving declared equations. It does not determine whether a canonical predicate is defined by source views, a source is described as an incomplete view of the canonical world, or a dependency creates existential values. It also says nothing by itself about coverage, duplicate behavior, authority, closed-world assumptions, or failure.

Proposition 6.1 (Read-direction underdetermination)**.** *There is no function from a bare schema mapping $F : S \to D$ to a unique canonical-query answer semantics for arbitrary source states.*

Proof. Take schemas S and D, each with one unary relation symbol, and let F map the source symbol to the canonical symbol. Consider source state $W = \{a\}$. A GAV read interpretation may define $D.P(x) \leftarrow S.R(x)$ under complete closed-world coverage, yielding $[\![P(x)]\!] = \{a\}$. A LAV interpretation may instead state that $S.R(x)$ is a sound but incomplete view of $D.P(x)$. Legal canonical worlds then include $\{a\}$ and $\{a, b\}$; the certain answer to $\neg P(b)$, if that nonmonotone query were admitted, differs from the GAV closed-world answer, and global completeness is not implied even for positive queries. Both interpretations use the same object assignment F. Since their legal-world and absence semantics differ, F does not determine a unique answer semantics. □

Status. PROVED as an underdetermination counterexample. It is not a theorem about a completed CatDB mapping language.

6.2 GAV

Under GAV, canonical relations are defined as queries over source relations:

$$D.P(\bar{x}) \leftarrow \phi_S(\bar{x}).$$

If P has defining source queries $\phi_1, \ldots, \phi_n$, the read profile must select one of three explicit combination rules:

$$[\![P]\!]_{\mathsf{bag}} = \biguplus_{i=1}^{n} [\![\phi_i]\!]\,, \qquad [\![P]\!]_{\mathsf{set}} = \bigcup_{i=1}^{n} [\![\phi_i]\!]\,,$$

or disjoint, which uses the set union only after a declared coverage obligation proves the source contributions pairwise disjoint. Under bag union, an overlapping tuple contributes the sum of its source multiplicities. Under set union, duplicate elimination is part of the declared canonical denotation. A failed disjointness obligation invalidates the plan rather than silently choosing either rule. Canonical conjunctive or SPJ queries can often be rewritten by unfolding. This is the first proposed CatDB profile because it avoids a general view-rewriting search. It still requires rules for unions of defining sources, completeness, overlap, multiplicity, generated identifiers, nulls, authority, and source failure. Unfolding into a logical source expression does not establish exact native lowering.

6.3 LAV

Under LAV, each source is described as a view over the canonical schema:

$$S.R(\bar{x}) \leftarrow \psi_D(\bar{x}).$$

Answering Q_D requires finding a rewriting using available views. Source selection and containment are part of this problem. When the views are incomplete, certain or maximally contained answers may be the defensible target, depending on the declared fragment [13, 20]. Reversing F or textually substituting a mapping is not an LAV algorithm.

6.4 GLAV

A GLAV dependency may relate a source query to a canonical query with existential values:

$$\forall \bar{x}\ (\phi_S(\bar{x}) \rightarrow \exists \bar{y}\, \psi_D(\bar{x}, \bar{y}))\,.$$

This expressiveness brings chase, labeled-null, termination, universal solution, and certain-answer questions. General GLAV federation is outside the first implementation target. Positive and unsupported fixtures may describe a bounded fragment, but a GAV-only compiler cannot be reported as a GLAV implementation.

6.5 Required direction record

Table 5: Minimum fields of a read-mapping record.

Field	Required meaning
`direction`	`gav`, `lav`, `glav`, or a separately specified functorial/data-exchange operation

Field	Required meaning
`query_fragment`	Source and canonical languages in which rewrite and answer semantics are defined
`coverage`	Complete, sound-incomplete, conditionally complete, or unknown
`closed_world scope`	Predicates and conditions for which absence may mean false
`keys` and `identity`	Source-local keys and canonical identity assumptions
`existentials`	Whether generated values occur and how they are represented
`multiplicity`	Set or bag interpretation and overlap behavior
`loss`	Dropped distinctions and non-reconstructible values
`authority`	Assertions the source may contribute or override
`revision`	Immutable schema, mapping, coverage, and policy identifiers

Missing direction or coverage leaves source access OBSTRUCTED.

7 Source descriptions, coverage, and source selection

7.1 Candidate-source rule

A source is a candidate for a canonical subgoal only when all of the following hold:

1. its read mapping covers the required canonical objects, paths, attributes, and predicates;
2. its coverage condition is compatible with the query condition;
3. available bindings satisfy the source access pattern;
4. exact remote execution or a permitted bounded residual exists;
5. source authority permits the fact contribution;
6. freshness and snapshot claims meet policy;
7. authorization, locality, and movement rules permit access; and
8. availability state is compatible with the selected failure profile.

A rejected source needs a stable reason. Silent omission is not a source selection algorithm.

7.2 Coverage obligation

Let $\mathsf{Deriv}(Q, \mathcal{M}, \vec{W})$ be the derivations admitted by the read theory, and K' the selected source set. A complete result requires

$$\forall d \in \mathsf{Deriv}(Q, \mathcal{M}, \vec{W}), \qquad \mathsf{sources}(d) \subseteq K', \tag{3}$$

or a stated containment, authority, or redundancy result that makes omitted sources irrelevant. If Equation (3) cannot be shown, the answer must be restricted to a declared coverage domain or marked incomplete.

7.3 Overlap is not identity

Two sources can contribute the same fact, distinct legitimate duplicates, or conflicting assertions. The planner needs a disjointness proof, exact canonical identity, bag-union rule, authority precedence, temporal or regional partition, reconciliation outcome, or explicit conflict. `UNION DISTINCT` is not a universal repair.

Proposition 7.1 (Distinct cannot implement general reconciliation)**.** *There are source overlaps for which duplicate elimination merges legitimate distinct facts, and overlaps for which it fails to merge semantically equivalent facts.*

Proof. For the first case, let two legitimate orders share all projected values after their source-local order identifiers are omitted. Set-based duplicate elimination collapses two facts to one. For the second, let two customer rows refer to one governed canonical customer but encode names as `Ångström Labs` and `ANGSTROM LABS`. Syntactic duplicate elimination retains both. Reconciliation therefore requires identity and projection information that row equality alone does not supply. □

Status. PROVED by counterexample; P7 reconciliation behavior remains OPEN.

7.4 Source-selection trace

Every plan explanation should list candidate, accepted, rejected, unavailable, and redundant sources; coverage predicates; mapping/capability revisions; binding patterns; containment or authority evidence; estimated contribution; and the reason the selected set supports the requested answer status. This trace is part of semantic debugging. Whether it reduces repair effort is an experimental question, not a present result.

8 Capability contracts and pushdown

8.1 Stable dispositions

Definition 8.1 (Operator disposition)**.** For one operator or scalar primitive under one connector/backend/version and semantic profile, the planner returns exactly one of:

1. `exact_pushdown`;
2. `exact_local_residual`;
3. `approved_approximation`; or
4. `unsupported`.

The second disposition may include an exact safe remote prefilter followed by local completion. The approximation disposition requires a separately approved error or recall contract. Unknown capability is not exact; it maps to `unsupported` until conformance evidence supports another outcome.

Capability identity includes connector code, backend product and version, configuration, session settings, and relevant source-schema metadata. Self-reported metadata without boundary tests is STRUCTURAL-ONLY.

8.2 Predicate pushdown

An exact predicate record covers comparison and Boolean operators; null and missing behavior; three-valued logic; collation and Unicode normalization; numeric coercion, precision, overflow, NaN, and infinity; temporal zones and precision; parameter binding; volatility; and error behavior. The running example's PostgreSQL comparison lacks a canonical-collation guarantee, so the name test stays local.

Proposition 8.2 (Exact safe prefilter plus residual)**.** *Let p be a canonical row predicate and r a remote predicate such that $p(x) \Rightarrow r(x)$ for every source row after admitted decoding. If the local residual evaluates p exactly on every row returned by r, then filtering remotely by r and locally by p equals filtering locally by p, with bag multiplicities preserved.*

Proof. Take any decoded row occurrence x. If $p(x)$ holds, the implication gives $r(x)$, so the remote stage retains the occurrence and the residual retains it. If $p(x)$ does not hold, the residual removes the occurrence whether or not $r(x)$ holds. Therefore each input occurrence has the same output multiplicity in both plans. □

Status. PROVED for pure total predicates and admitted decoding. Connector truthfulness, remote errors, and finite resource bounds remain OPEN.

8.3 Projection pushdown

Projection is exact only when fields needed by later joins, residual filters, ordering, reconciliation, and provenance remain available; computed expressions preserve the scalar profile; and omitting a field does not change authorization or error behavior. Column pruning, nested-field projection, and computed projection are separate capabilities.

8.4 Aggregation pushdown

An aggregation capability must state grouping equality and collation, null grouping, bag multiplicity, `DISTINCT`, empty-input behavior, filtered aggregates, overflow and decimal precision, order-sensitive functions, and partial/final decomposition. A source that can compute a partial aggregate does not thereby support exact final aggregation. Aggregation is outside the first query-profile target in the current implementation roadmap.

8.5 Join pushdown

A join capability records join kind, equality, null matching, casts, collation, precision, multiplicity, duplicate-column behavior, co-location, transaction scope, order, estimates, and native-plan evidence. Cross-source join pushdown is possible only when the selected engine can access both inputs under the declared semantics and policy. General backend support for SQL joins is not enough.

8.6 Limit and Top-N pushdown

An unordered limit is not an ordered top n. Exact Top-N pushdown requires agreement on order keys, directions, null placement, collation, numeric and temporal comparison, ties, and every residual operator that can alter membership. A limit cannot move below a filter, join, union, or reconciliation stage when that move can exclude final members.

Proposition 8.3 (Cost cannot repair inexact semantics)**.** *If a pushed operator u differs from the canonical operator v on an admitted input and no later residual repairs that difference, no cost ranking can make the resulting plan exact.*

Proof. On the admitted witness input, the physical result contains the output of u, while the canonical denotation contains the output of v, and these outputs differ. Cost changes only the ranking among physical alternatives; it does not change either denotation. Thus Equation (FED-EXACT) fails. □

Status. PROVED as a semantic separation. Plan costs and connector behavior are not evaluated here.

8.7 Capability conformance

Each exact capability needs positive fixtures, semantic-boundary fixtures, negative cases, source-native oracles, version/configuration coverage, recorded commands and parameters, and downgrade behavior. Conformance must be rerun after a relied-on source or connector revision changes. A plan cache whose capability dependency changes is stale even if its SQL text is unchanged.

9 Cross-source join placement and movement

9.1 Admissible alternatives

For $R \bowtie S$ on different systems, the planner may consider:

- query shipping to an engine that can access both inputs;
- shipping a reduced R or S input to the other source;
- a local join after reduced scans;
- a batched bind join;

- exact semijoin reduction;
- execution in a third engine;
- or a versioned materialization that satisfies freshness and semantic identity.

Approximate Bloom filters can reduce movement if false positives are repaired by an exact join. False negatives are incompatible with an exact plan.

9.2 Policy precedes cost

Residency, authorization, purpose limitation, source load, no-staging rules, freshness, consistency, provenance retention, and capability checks prune the plan space before cost ranking. P14 may later optimize among admissible plans. P5 owns the semantic and capability conditions that define admissibility.

9.3 Required measurements

For each executed alternative, record planning time; calls and retries; request and response bytes; rows scanned, returned, exchanged, and processed locally; first-row and total latency; source load where observable; local CPU and peak memory; temporary storage; rate-limit waits; reconciliation and provenance cost; and estimate error. A transfer-reduction statement is inadmissible without a named baseline and observed bytes.

The running example currently fixes only a candidate local-join placement. It has no measured alternative, no cost model, and no planner regret.

10 Snapshot, availability, retry, and failure semantics

10.1 A vector of source states

Without a distributed transaction or another coordination protocol, execution observes

$$\vec{\sigma} = (\sigma_1, \ldots, \sigma_n),$$

not one atomic global snapshot. Individual entries may be database transaction snapshots, file content digests, API ETags or cursors, materialization revisions, coordinated checkpoints, or `unknown`.

Proposition 10.1 (Snapshot-vector nonimplication)**.** *A vector of independently obtained source snapshot identifiers does not, by itself, establish that the observations belong to one atomic global state.*

Proof. Consider two sources A and B and a transfer whose debit commits at A before its credit commits at B. An execution can observe σ_A after the debit and σ_B before the credit. Both identifiers are valid local snapshots, yet their combination is not a state produced by an atomic cross-source transaction. Establishing a stronger relation requires additional coordination or history evidence. □

Status. PROVED by an anomaly history; no CatDB snapshot implementation exists.

10.2 Failure policies

Table 6: Minimum availability profiles.

Policy	Behavior	Status
`strict`	A required-source or required-stage failure returns an error and no complete-answer claim.	OBSTRUCTED
`explicit_partial`	Successful contributions may be returned with partial status, missing sources, unsatisfied coverage, warnings, and provenance.	OBSTRUCTED
`declared_stale fallback`	A failed live contribution may be replaced by an authorized materialization whose identity, snapshot, and freshness are returned.	OBSTRUCTED

Optionality must be declared before failure. The planner cannot relabel a required source as optional after a timeout.

10.3 SPARQL federation as a narrow precedent

SPARQL 1.1 Federated Query directs remote work with `SERVICE`. Failure without `SILENT` fails the query; `SERVICE SILENT` changes the algebraic outcome [29]. CatDB should not copy silent failure as a default. The standard instead illustrates why remote failure is part of query semantics rather than an afterthought.

10.4 Retry discipline

Retries require an idempotent read or stable request identity, fixed or explicitly renewed snapshot behavior, bounded attempts, rate-limit handling, duplicate-response protection, cancellation propagation, and attempt-level provenance. A retry that silently advances to a later source state changes the result's snapshot contract. Retrying after partial output also requires a deduplication rule based on request/row identity rather than position alone.

10.5 Error taxonomy

The plan distinguishes planning rejection, capability rejection, authentication or authorization failure, source timeout, rate limit, transport failure, malformed row, conversion failure, remote query error, local residual error, cancellation, reconciliation conflict, and provenance-write failure. Each error is classified as row-, source-, stage-, or query-fatal. A required source timeout is not a zero-row scan.

11 Result reconciliation and provenance

11.1 Reconciliation operators

Federation does not end when rows arrive. Reconciliation may normalize scalars and units, translate source identifiers, apply deterministic equivalence, record candidate links, enforce authority precedence, preserve contradictions, perform temporal supersession, quarantine malformed values, or create generated canonical objects. These actions must be named typed operators or postconditions. They must not be hidden in transport code.

Possible or probable identity remains uncertain until an authority rule approves it. The running example's `crm-017` to customer 17 link is therefore a versioned decision with evidence, status, and revision. Removing duplicate result rows would neither establish nor adequately document that decision.

11.2 Required reconciliation evidence

A reconciled row should preserve source-local identifiers; canonical identity, if established; decision revision; supporting evidence and authority; unresolved alternatives; contradiction state; and a reversible history reference when applicable. P7 owns the full identity model. P5 must carry the decision without converting uncertainty into ordinary equality.

11.3 Provenance boundary

The result envelope must reference derivation evidence at the granularity promised by P13. At minimum, a future P5 run needs source records or source query/result digests, source snapshots, schema and mapping revisions, plan stages, residual and reconciliation decisions, and output identity. Returning the right values does not establish lineage. Likewise, an abstract provenance type does not show that connectors emit complete evidence or that storage overhead is acceptable.

11.4 Plan and result identity

A plan digest depends on every revision that affects denotation or admissibility:

$$\mathsf{pid} = H(Q_D, D, \mathcal{M}, \mathcal{S}, \mathcal{C}, \mathcal{P}, \mathcal{F}, \mathcal{R}).$$

Physical statistics can have a separate revision, but their source and age remain visible. Credentials need not enter the digest as secrets, although authorization scope and policy identity do. A result identity additionally depends on $\vec{\sigma}$, observed failures, fallbacks, and reconciliation outcomes.

12 SQL, API, and file connector contracts

12.1 Connector boundary

Connectors may own authentication, transport, dialect rendering, request execution, streaming, cancellation, and source telemetry. They may not make undeclared canonical identity, authority, loss, or reconciliation decisions. Those decisions belong to mapping, plan, or policy artifacts.

12.2 SQL sources

A SQL connector record includes product, server version, protocol and driver, identifier and parameter rules, supported types and codecs, null and collation behavior, numeric and temporal semantics, transaction and isolation, timeouts, cancellation, prepared statements, native-plan capture, streaming and fetch size, statistics, staging capabilities, and exact per-operator claims.

SQLite and PostgreSQL are not interchangeable. Their typing, collations, concurrency, isolation, error handling, and plan inspection differ. The first portable subset must state those differences instead of normalizing them away in prose.

12.3 HTTP and API sources

An API profile includes endpoint and revision, method and parameter encoding, binding patterns, filter/projection/sort/limit support, pagination and cursor stability, request/page limits, rate limits, authentication, retry and idempotency, response schema, null and missing encoding, ETags or version behavior, order guarantees, and provenance identifiers. Fetching a bounded API response and filtering locally is a fallback, not predicate pushdown.

12.4 File sources

A file profile includes format revision, content digest or object version, encoding, delimiter and quoting, header/schema policy, null and missing tokens, compression and splittability, row-order significance, partitions, mutation/read isolation, malformed-row policy, and available pruning. CSV and JSON have no universal canonical type or null behavior. Exact conversion requires a versioned schema or an explicit inference profile.

12.5 Protocol-family reuse

A protocol-family adapter may share transport or decoding logic across products while keeping target-specific dialect and capability modules. This is an ENGINEERING HYPOTHESIS, not a present implementation. Conformance must demonstrate that reuse does not hide vendor-specific differences; otherwise the abstraction should narrow.

13 Current CatDB evidence and implementation status

13.1 Existing prerequisite artifacts

The repository currently has:

- Rust crates for core values and semantic IR;
- Rust crates for schema, mapping, migration, and the CLI;
- finite schema, path, mapping, instance, migration, loss, and diff representations and checks;
- a shared fixture manifest for semantic-core cases;
- a Haskell reference project for the admitted prerequisite slice; and
- Lean schema, path, and mapping declarations with a separate coverage audit.

The accepted P1 boundary reports 24 cross-language fixtures, 16 independently computed semantic outcomes, eight structural-only outcomes, no Rust/Haskell disagreements, and seven deterministic properties over 1,550 generated cases. Those figures are cited here only as prerequisite scope. They establish no canonical-query, connector, or federation behavior.

13.2 Absent P5 artifacts

No current artifact implements:

- a canonical query AST, profile, typing judgment, or denotation;
- GAV, LAV, GLAV, or another read interpretation;
- source descriptions, coverage, authority, or source selection;
- connector capability manifests or conformance;
- logical rewrite or physical federation planning;
- SQL, API, CSV, or JSON query execution;
- exchange, local residual, or cross-source join execution;
- result/failure/snapshot envelopes;
- P5 reconciliation or provenance;
- paired native/federated result comparison; or
- P5 calls, rows, bytes, plan, latency, throughput, or scale measurement.

Table 7: Implementation claims and exact present status.

Claim	Status	Required promotion evidence
Canonical query typing	OBSTRUCTED	Accepted profile, checker, positive/negative fixtures, independent denotation
GAV unfolding	OBSTRUCTED	Read profile, Rust/Haskell result agreement, bounded proof targets
LAV/GLAV query answering	OPEN	Stated fragment, algorithm, certain/contained answer oracle, unsupported cases
Exact pushdown	OPEN	Connector/version profile, conformance, native oracle, downgrade path
Local residual	OPEN	Resource bound and exact result equivalence
Join placement	OPEN	Enumerated admissible alternatives, cost model, executed comparisons
Partial/stale results	OBSTRUCTED	Failure profile, result envelope, injected failures, provenance
Heterogeneous execution	OPEN	SQLite, PostgreSQL, file/API connectors and differential run
Performance or reduced movement	OPEN	Correctness-passing preregistered measurements against named baselines

13.3 Why mapping code is not federation evidence

Mapping validation can show that object and path assignments are typed and preserve declared equations. Federation additionally requires a query denotation, a read direction, source coverage, physical capability, snapshot, failure, reconciliation, and result semantics. None follows from successful mapping composition. The current mapping slice is therefore a dependency, not a partial P5 runtime.

14 Fixture and implementation roadmap

14.1 Semantic gate before connectors

The next artifacts, in order, are:

1. `query-profile-v1`, fixing the finite canonical algebra and denotation;
2. `read-mapping-profile-v1`, selecting closed GAV first and delimiting LAV/GLAV;
3. source-description, coverage, capability, result, failure, snapshot, and reconciliation profiles;
4. a new immutable fixture schema and manifest;

5. independent Rust and Haskell finite evaluators;
6. generated and hand-worked counterexamples; and
7. selected Lean rewrite laws only after the profiles stabilize.

Writing connectors first would encode accidental SQL behavior before the canonical semantics exists.

14.2 Implementation sequence

The proposed sequence is:

1. in-memory canonical oracle;
2. SQLite vertical query path;
3. PostgreSQL portable-subset parity;
4. connector capability manifests and conformance;
5. CSV and JSON with immutable content identity;
6. deterministic local HTTP fixture server;
7. heterogeneous customer-order federation;
8. result/failure/snapshot/provenance integration; and
9. only then broader connectors, adaptive planning, or scale claims.

Each stage must retain unsupported cases. An empty crate, a trait declaration, generated SQL string, or mocked connector response does not count as the stage's result.

14.3 Fixture families

Table 8: Required P5 fixture families.

Family	Minimum purpose
`query_typing`	Accepted and rejected canonical queries with stable diagnostics
`read_mapping`	Exact GAV cases, bounded LAV/GLAV cases, and direction errors
`source_selection`	Candidate, redundant, incomplete, unavailable, and authority-rejected sources
`capability`	Exact, residual, approximate, unsupported, and unknown operator behavior
`logical_rewrite`	Normalization and source-fragment generation
`physical_plan`	Remote stages, exchanges, local residuals and joins, reconciliation

Family	Minimum purpose
`result_envelope`	Exact, certain, partial, stale, approximate, and error outcomes
`dialect`	SQLite/PostgreSQL rendering, binding, scalar, collation, and error boundaries
`api`	Binding patterns, pagination, rate limits, retry, snapshot, malformed results
`file`	CSV/JSON schema, codecs, digest, mutation, pruning, and malformed rows
`equivalence`	Canonical, source-native, and federated normalized results
`metrics`	Calls, rows, bytes, timings, estimates, and source/local work

14.4 Boundary and negative fixtures

Required boundaries include empty input, duplicate rows, all-null groups after aggregation is admitted, decimal overflow, NaN and infinity where supported, Unicode composed and decomposed strings, locale-sensitive sorting, daylight saving gaps and overlaps, timestamp precision loss, JSON missing versus null, CSV empty versus quoted empty, duplicate API pages, cursor change on retry, source timeout before and after partial output, cancellation, overlapping authority, possible identity, and repeated unordered limits.

Negative fixtures must reject unspecified mapping direction, LAV treated as GAV, unknown coverage used for completeness, unknown capability treated as exact, collation mismatch without a residual, incorrect null equality, unordered limit treated as Top-N, limit below a residual, unstable API cursor, file mutation during scan, required-source timeout under strict mode, partial data labeled exact, stale fallback without freshness, global snapshot claimed from independent sources, contradiction erased during reconciliation, and source-native divergence.

14.5 Generative properties

Within the finite profile, generate properties for output-schema preservation, remote-plus-residual equivalence, strict failure status, source-order permutation under unordered bag semantics, capability disablement, plan dependency identity, missing-contribution completeness, and agreement between connector telemetry and envelope counts within a declared tolerance. A counterexample must be minimized and retained. Generated passing cases support only the explored model bound.

15 Experimental evaluation and falsification plan

15.1 Research questions

Table 9: Future P5 research questions.

ID	Question
P5-RQ1	Does mapping-aware rewriting preserve the declared canonical denotation?
P5-RQ2	Does capability-aware pushdown reduce rows, bytes, calls, or latency without changing result or error semantics?
P5-RQ3	Which admissible join placement performs best as size, selectivity, latency, bandwidth, skew, and rate limits vary?
P5-RQ4	Are incompleteness, source failure, snapshot skew, and staleness labeled correctly?
P5-RQ5	What planning, execution, storage, and maintenance overhead comes from typed mapping, reconciliation, status, and provenance?
P5-RQ6	Does the combined contract improve semantic diagnosis or repair relative to selected federation baselines?

15.2 Baselines

The minimum baseline registry contains hand-written source-native queries; hand-written orchestration; pull-all/local evaluation; a simple filter/project-then-local-join heuristic; an R*-style communication-aware planner or simulator; Apache Calcite in an exact adapter configuration; Trino over mutually supported connectors; Ontop for an applicable mapping-based virtual-access fragment; and BigDAWG or another reproducible polystore workload where feasible. A baseline that cannot be run is related work, not a measured competitor.

15.3 Workload axes

Vary source count and kind, mapping complexity, GAV overlap, source size, predicate and join selectivity, skew, network latency and bandwidth, API page size and rate limit, availability, snapshot skew, duplicate and conflict rate, reconciliation mode, provenance granularity, and warm/cold plan and result caches. The first required systems population is SQLite, PostgreSQL, CSV, JSON, and one deterministic HTTP source. It does not represent universal heterogeneity.

15.4 Correctness before performance

For every seed:

1. freeze or identify source states;
2. evaluate the canonical oracle over the constructed canonical instance;
3. run source-native oracle queries;
4. compile and execute the CatDB candidate;

5. normalize encodings without erasing null, missing, or error states;
6. compare bags, scalar values, declared order, and errors;
7. compare answer status, coverage, missing-source, snapshot, and reconciliation metadata;
8. verify capability and residual decisions;
9. verify provenance references; and
10. retain inputs, plans, commands, outputs, measurements, and failures.

A run with a semantic or provenance mismatch is a failed correctness result. Its timing does not enter a favorable performance summary.

15.5 Measurements

Report rewrite and planning time; plan-cache state; connector calls, retries, and cancellations; source rows scanned and returned; request, response, and exchange bytes; local rows processed; first-row and total latency; source and local CPU/memory where observable; native plan text or digest; estimate error; reconciliation time; provenance overhead; result-status overhead; failure and fallback outcome; and stale age. Report units and distinguish measured, estimated, and unavailable fields.

15.6 Join-placement study

For each workload, enumerate admissible alternatives and record the selected plan, best observed plan, regret, planning overhead, calls, transferred rows/bytes, latency, source load, and policy-pruned plans. Include small-local to large-remote, large-local to small-remote, high- and low-selectivity joins, skewed keys, API bind-join limits, a co-located PostgreSQL join, a SQLite/PostgreSQL local join, and reused versus refreshed materialization.

15.7 Failure study

Inject unavailability before planning, timeout before the first row, timeout after partial rows, malformed values, authentication failure, rate limiting, duplicate API pages, cursor change, mutation between pages, PostgreSQL transaction abort, SQLite lock, stale materialization, capability downgrade, and reconciliation conflict. Test `strict`, `explicit_partial`, and `declared_stale_fallback` separately.

15.8 Ablations

Compare physical federation without typed mappings; mappings without capability checks; mappings and capabilities without reconciliation; those features without provenance; the full profile; and the full profile with materialization disabled. An ablation that permits wrong answers is a semantic stress condition, not a fair high-performance competitor.

15.9 Reproducibility and statistics

Preregister immutable source, baseline, data, environment, and schedule artifacts. Use a detached worktree or container from the recorded commit. Record operating system, hardware, database and connector versions, collations, time zone, isolation, network shaping, compiler flags, and cache stratum. Retain failed runs and report exclusions. Independent experimental units are fresh source placements and data seeds on named machines; repeated queries inside one process are nested measurements.

Correctness endpoints are categorical. Performance claims require predeclared estimands, thresholds, confidence intervals, and multiplicity treatment. The current repository contains no P5 observation to analyze.

15.10 Falsification conditions

The bounded hypothesis is falsified or must narrow if:

1. a compiled plan disagrees with the canonical oracle;
2. a complete label omits a required source;
3. an exact capability changes bag, null, collation, numeric, temporal, ordering, or error semantics;
4. an unsupported operation is silently approximated;
5. source failure is indistinguishable from empty data;
6. stale fallback omits identity or freshness;
7. independent snapshots are labeled globally atomic without evidence;
8. calls, rows, and bytes cannot be measured accurately enough to audit pushdown or placement;
9. manual source-specific code remains necessary per query inside the claimed fragment;
10. the combined contract supplies no checkable delta over mediator, OBDA, federation, or polystore baselines; or
11. any performance statement fails its preregistered threshold after mapping, reconciliation, provenance, residual, and failure costs are included.

16 Limitations and threats to validity

16.1 This is a contract paper

The main limitation is that the federation path does not exist yet. The definitions can catch an incoherent claim before code is written, and the counterexamples rule out a few tempting shortcuts. They cannot tell us whether the interfaces are usable, whether connector manifests remain accurate, or whether a planner will find good plans.

16.2 The query fragment is not frozen

P2 has not supplied an accepted canonical query profile, and P4 has not supplied an accepted read-direction profile. The equations in this paper are therefore a manuscript contract. Later profile work may narrow the operator set, change error behavior, or reject a proposed mapping form. Any such change reopens dependent claims.

16.3 Heterogeneity is bounded

SQLite, PostgreSQL, CSV, JSON, and one deterministic HTTP source cover useful differences, but they do not represent warehouses, graph engines, document stores, vector systems, streams, or vendor APIs in general. Results from the first source set must remain source- and version-specific.

16.4 Coverage can be hard to establish

Completeness and authority often depend on organizational facts that source schemas do not encode. A machine-readable coverage predicate may still be wrong or stale. The contract can make the assumption visible and test some consequences; it cannot manufacture evidence that a business source is complete.

16.5 Connector conformance is finite

Boundary fixtures cannot cover every backend setting, locale, extension, data value, or version. Exact capability claims must name the tested profile and downgrade when it changes. Property tests and differential runs support a bounded population, not universal connector correctness.

16.6 Partial answers remain application-sensitive

An explicit partial label prevents one kind of deception, but it does not make the data fit for every decision. Some applications should reject every partial answer. Other queries have no useful partial interpretation at all. Availability policy therefore belongs to the query or product contract.

16.7 Snapshots may remain incomparable

Even precise local snapshot identifiers may lack a meaningful cross-source order. The result can report the vector and still be unsuitable for an invariant that needs one coordinated state. The paper offers no distributed transaction protocol.

16.8 Reconciliation depends on governance

Typed plans can carry identity and authority decisions, but they do not decide business meaning. Human review, policy, and reversible uncertainty remain necessary. P7 and P13 are substantive dependencies, not optional metadata extensions.

16.9 Performance and maintenance benefits are unknown

The proposed obligations cost something: more planning, tests, metadata, storage, and runtime checks. Better diagnostics may come with higher latency, and conventional federation may still win once every cost is counted. This evidence cut supports no claim about native performance, transfer reduction, scale, or maintenance.

16.10 Prior-art coverage may still be incomplete

The comparison covers central mediator, view-rewriting, federation, OBDA, SPARQL, adapter, distributed-optimizer, and polystore sources. A final novelty claim requires a narrower systematic review and expert audit. The combined contract is a candidate contribution, not a settled novelty result.

17 Open and obstructed questions

17.1 Open questions

1. Which finite query fragment gives both a useful heterogeneous example and tractable exact semantics?
2. How should coverage, closed-world scope, and authority compose across mapping revisions?
3. Which connector capabilities can be derived from a formal backend model, and which can only be tested?
4. What local-residual resource bound avoids disguising pull-all execution as pushdown?
5. Which join alternatives belong in P5 before P14 owns a general cost model?
6. What P13 provenance granularity is useful at tolerable overhead?
7. How should plan-cache identity separate semantics, capabilities, statistics, authorization, and credentials?
8. Can mapping-aware explanations measurably shorten fault diagnosis or repair compared with Calcite, Trino, and hand-written orchestration?
9. Which Ontop, Calcite, Trino, and BigDAWG workloads can be reproduced fairly under one source population?
10. When should the planner prefer a stale materialization over an explicit partial live answer?

17.2 Obstructed questions

Exact rewriting is obstructed until the query denotation and read direction are approved. Completeness is obstructed until coverage and closed-world assumptions are represented.

Aggregate, join, and Top-N pushdown are obstructed until multiplicity, null, collation, numeric, temporal, order, and error profiles are fixed. Partial correctness is obstructed until the result and failure envelope is fixed. Global consistency claims are obstructed without coordination evidence. Execution and performance claims are obstructed by the absent query, planner, connector, telemetry, and benchmark path.

18 Conclusion

Federation is often described as a connectivity problem. That description is too small. A catalog can expose several systems through one query language while leaving meaning, completeness, identity, and failure for the user to infer. A typed schema mapping helps, but it does not choose a read direction or prove that a native operator matches the canonical one.

The stricter target starts with a versioned read interpretation for every source, including coverage and authority. Each operator is exact, residual, approximate, or unsupported. Plans retain movement, snapshot, reconciliation, failure, and provenance obligations. Results report their status and the reason for it.

The customer-and-order example makes the boundary concrete. The name comparison stays local because PostgreSQL has not certified the canonical collation. The tier link is a governed identity decision rather than a deduplication trick. A missing tier source fails under strict mode, produces an identified partial answer under explicit-partial mode, or uses a named stale materialization under the fallback profile. The same source rows do not justify the same answer status under all three policies.

Nothing here establishes a running federated backend. The repository has useful schema and mapping prerequisites, but no query semantics, read-mapping profile, planner, connector executor, result envelope, or P5 experiment. A defensible next milestone is smaller: one closed GAV query profile with an independent finite oracle. SQLite and PostgreSQL differential execution, an immutable file or deterministic API source, and failure-aware result provenance come after that.

The intended claim remains narrow. CatDB may connect typed, immutable semantic mappings to capability-aware physical choices while making the conditions of an answer inspectable. Established mediator, view-rewriting, federation, OBDA, SPARQL, distributed-optimizer, and polystore work supplies the baseline. Implementation and comparison must decide whether the combined contract earns a systems contribution.

A Claim-status ledger

Table 10: Claim status and promotion evidence.

Claim	Evidence class	Operational status	Promotion or retained boundary
Federation, mediators, source descriptions, GAV/LAV/GLAV, view rewriting, capabilities, pushdown, distributed placement, OBDA, SPARQL federation, and polystores predate CatDB	SUPPORTED BY PRIOR LITERATURE	CONDITIONAL	Retain cited fragment and system assumptions
A bare schema mapping does not determine one read semantics	PROVED	PROVED	Theorem 6.1; no runtime implication
Strict required-source failure excludes exact status under the declared profile	PROVED	PROVED	Theorem 5.5; detection and enforcement remain open
Safe prefilter plus exact residual preserves bag-filter results	PROVED	PROVED	Theorem 8.2; pure total predicates only
Cost ranking cannot repair semantic inexactness	PROVED	PROVED	Theorem 8.3
Independent source snapshots do not imply one atomic global state	PROVED	PROVED	Theorem 10.1; no coordination protocol supplied
Bounded schema/mapping prerequisite agreement exists	COMPUTATIONAL	COMPUTATIONAL	Accepted P1 prerequisite fixture/review boundary; no P5 promotion
Selected schema/path/mapping laws are kernel checked	MACHINE-CHECKED	CONDITIONAL	Exact P1 prerequisite declarations only; no query or connector coverage
Canonical query typing and denotation	ENGINEERING HYPOTHESIS	OBSTRUCTED	Accepted query profile, independent oracle, fixtures
GAV rewriting through read mappings	ENGINEERING HYPOTHESIS	OBSTRUCTED	Accepted direction/coverage profile and cross-language agreement
LAV/GLAV query answering	ENGINEERING HYPOTHESIS	OPEN	Bounded fragment, algorithm, certain/contained oracle
Capability-aware exact pushdown	ENGINEERING HYPOTHESIS	OPEN	Versioned conformance and native equivalence
Cross-source join placement	ENGINEERING HYPOTHESIS	OPEN	Planner, alternatives, policy gate, measurements
Explicit partial and stale results	ENGINEERING HYPOTHESIS	OBSTRUCTED	Failure/result profile and injected-failure evidence
Typed reconciliation improves correctness or diagnosis	ENGINEERING HYPOTHESIS	OPEN	P7 contract, baseline comparison, controlled study

Claim	Evidence class	Operational status	Promotion or retained boundary
P5 provenance is complete at a stated granularity	ENGINEERING HYPOTHESIS	OPEN	P13 semantics, connector capture, audit and overhead
Pushdown reduces calls, rows, bytes, or latency	ENGINEERING HYPOTHESIS	OPEN	Correctness-passing, preregistered baseline measurements
CatDB remains outside the row hot path or scales to large workloads	SPECULATIVE	OPEN	Native plans, scale tiers, resource and boundary evidence
Universal heterogeneous federation	NONCLAIM	no promotion path	Explicitly excluded

B Result-envelope field contract

Table 11: Minimum executable result envelope.

Field	Meaning
`answer_status`	Exact, certain-complete, certain-sound, partial, approximate, stale, or error
`query_revision`	Canonical query and profile identity
`schema_revision`	Immutable canonical schema identity
`mapping_revisions`	Every read mapping and direction profile used
`capability revisions`	Connector/backend/version capability records used
`selected_sources`	Planned and actually contacted sources
`missing_sources`	Required or optional sources not observed, with reasons
`source_snapshots`	Per-source transaction snapshot, cursor, ETag, digest, materialization, or unknown
`coverage`	Claimed domain and unsatisfied predicates, regions, or periods
`reconciliation revision`	Identity, authority, normalization, and conflict rules
`warnings`	Approximation, malformed rows, retries, fallback, and freshness notices
`plan_digest`	Stable logical/physical plan identity with semantic dependencies
`metrics`	Calls, retries, rows, bytes, timings, resource use, estimates, and unavailable markers
`provenance_ref`	Link to result- or row-level derivation evidence under the declared P13 profile

C Running-example acceptance matrix

Table 12: Future acceptance cases for the customer/order example.

Case	Expected status	Required source behavior	Required evidence
All three sources available; admitted snapshots and coverage; local name residual	`exact` within declared domain	PostgreSQL, SQLite, file succeed	Native/federated bag equality; residual trace; all revisions
Tier source missing under strict	`error`	Required file unavailable	No complete rows; failure identity and attempts
Tier source missing under explicit partial	`partial`	Customer/order succeed; tier fails	Missing tier contribution, unsatisfied coverage, successful-source provenance
Tier source missing with authorized materialization	`stale`	Named fallback succeeds	Materialization revision, source snapshot, age, policy
PostgreSQL remote collation claimed exact without conformance	rejected	Capability mismatch	Stable capability diagnostic
Remote coarse name filter plus exact local canonical residual	`exact` if all other obligations pass	Safe remote superset	Theorem 8.2 premises and differential case
Unknown mapping direction	rejected	No read interpretation	Stable direction diagnostic
Tier identity remains possible, not approved	partial, conflict, or rejected per profile	No canonical identity	Uncertainty preserved; no string-based merge
SQLite mutates between pages/scans without accepted snapshot	rejected or nonexact	Snapshot drift	Old/new snapshot identities and drift diagnostic
Unordered limit pushed below residual	rejected	Membership can change	Counterexample retained

D Reproducibility checklist for a future P5 result

A claim-bearing run must retain:

1. clean source commit/tree and all dependency-lock digests;
2. accepted query, mapping, scalar, coverage, capability, policy, failure, reconciliation, and provenance profile digests;

3. fixture schema, manifest, case digests, workload generator, and seeds;
4. Rust/Haskell source and generated comparison output;
5. Lean theorem names, source digest, axiom inventory, and placeholder audit for any machine-checked claim;
6. database and API/file images, versions, schema, configuration, collation, time zone, and snapshot identities;
7. generated logical and physical plans, native queries, parameters, native plans, and source responses;
8. exact result normalization and oracle-comparison output;
9. calls, rows, bytes, retries, errors, timings, resource use, and unavailable telemetry markers;
10. baseline source, version, capability, configuration, and validation;
11. environment, network, cache, randomization, repetition, stopping, exclusion, and statistical records;
12. all failed and partial runs;
13. reconciliation decisions and authority evidence; and
14. provenance linking every published result to the preceding artifacts.

Golden fixture values are comparison targets. An implementation that reads and returns them does not satisfy the semantic contract.

References

[1] Apache Arrow project. Columnar format. Official format specification, version 1.5, 2026. Accessed 2026-07-23.

[2] Apache Calcite project. Adapters. Official project documentation, 2026. Accessed 2026-07-23.

[3] Ron Avnur and Joseph M. Hellerstein. Eddies: Continuously adaptive query processing. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 261–272, 2000.

[4] Edmon Begoli, Jesús Camacho-Rodríguez, Julian Hyde, Michael J. Mior, and Daniel Lemire. Apache calcite: A foundational framework for optimized query processing over heterogeneous data sources. In *Proceedings of the 2018 International Conference on Management of Data*, pages 221–230, 2018.

[5] 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.

[6] Michael J. Carey, Laura M. Haas, Peter M. Schwarz, Manish Arya, William F. Cody, Ronald Fagin, Myron Flickner, Allen W. Luniewski, Wayne Niblack, Dragutin Petkovic, John Thomas, John H. Williams, and Edward L. Wimmers. Towards heterogeneous multimedia information systems: The Garlic approach. In *Proceedings of the Fifth International Workshop on Research Issues in Data Engineering: Distributed Object Management*, 1995.

[7] Shumo Chu, Konstantin Weitz, Alvin Cheung, and Dan Suciu. HoTTSQL: Proving query rewrites with univalent SQL semantics. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 510–524, 2017.

[8] Jennie Duggan, Aaron J. Elmore, Michael Stonebraker, Magda Balazinska, Bill Howe, Jeremy Kepner, Sam Madden, David Maier, Tim Mattson, and Stan Zdonik. The BigDAWG polystore system. *SIGMOD Record*, 44(2):11–16, 2015.

[9] 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.

[10] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Vasilis Vassalos, and Jennifer Widom. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

[11] Goetz Graefe. Volcano—an extensible and parallel query evaluation system. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):120–135, 1994.

[12] Laura M. Haas, Donald Kossmann, Edward L. Wimmers, and Jun Yang. Optimizing queries across diverse data sources. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 276–285, 1997.

[13] Alon Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[14] Vancho Josifovski, Peter Schwarz, Laura M. Haas, and Eileen Lin. Garlic: A new flavor of federated query processing for DB2. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 524–532, 2002.

[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] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, pages 251–262, 1996.

[18] Lothar F. Mackert and Guy M. Lohman. R* optimizer validation and performance evaluation for distributed queries. In *Proceedings of the 12th International Conference on Very Large Data Bases*, pages 149–159, 1986.

[19] Yannis Papakonstantinou, Ashish Gupta, and Laura M. Haas. Capabilities-based query rewriting in mediator systems. *Distributed and Parallel Databases*, 6(1):73–110, 1998.

[20] Rachel Pottinger and Alon Y. Halevy. Minicon: A scalable algorithm for answering queries using views. In *Proceedings of the 26th International Conference on Very Large Data Bases*, pages 484–495, 2000.

[21] P. Griffiths Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, pages 23–34, 1979.

[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] Substrait project. Substrait specification. Official specification, 2026. Accessed 2026-07-23.

[24] Trino Software Foundation. Connector development. Official Trino documentation, 2026. Accessed 2026-07-23.

[25] Trino Software Foundation. Postgresql connector. Official Trino documentation, 2026. Accessed 2026-07-23.

[26] Trino Software Foundation. Pushdown. Official Trino documentation, 2026. Accessed 2026-07-23.

[27] Vasilis Vassalos and Yannis Papakonstantinou. Describing and using query capabilities of heterogeneous sources. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 256–265, 1997.

[28] W3C RDB2RDF Working Group. R2RML: RDB to RDF mapping language. W3c recommendation, World Wide Web Consortium, 2012.

[29] W3C SPARQL Working Group. SPARQL 1.1 federated query. W3c recommendation, World Wide Web Consortium, 2013.

[30] W3C SPARQL Working Group. SPARQL 1.1 query language. W3c recommendation, World Wide Web Consortium, 2013.

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