Cost-Based Planning for Semantic Integration: A Falsifiable Contract for Federated, Cached, and Materialized Execution

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

23 July 2026

Provisional status: OBSTRUCTED

Abstract

A federated planner can find a low-latency route that returns the wrong answer contract. It may push a comparison to a source with different collation, reuse a cache created under an old mapping, choose a fast vector index whose recall is too low, or protect user latency by shifting damaging work into a production source. Ordinary cost models do not excuse those mistakes, and a larger weighted sum does not repair them.

This paper gives a provisional planning contract for semantic integration in CatDB. A request binds a logical query to immutable schema and mapping revisions, scalar semantics, source coverage, freshness, consistency, provenance, reconciliation, disclosure, and approximation profiles. Candidate plans include virtual federation, remote computation, local execution after transfer, semantic-cache reuse, partial and full materialization, incremental refresh, full recomputation, shared transient results, and runtime adaptation. The planner first rejects plans that cannot satisfy the request. It then predicts a typed outcome vector with natural units and uncertainty: latency, network and storage traffic, local and remote work, memory, source load, freshness, coordination, provenance overhead, approximation quality, and failure behavior. A versioned objective chooses among feasible plans or returns a probe, fallback, no-feasible-plan result, or abstention.

The paper also specifies decision certificates, adversarial counterexamples, calibration and benchmark protocols, baselines, semantic-feature ablations, and a staged Rust and backend roadmap. The possible contribution is narrow: whether versioned semantic metadata can improve constrained plan decisions after its collection and maintenance costs are counted. Cost-based optimization, federated cost models, capability-aware pushdown, adaptive query processing, semantic caching, materialized views, multiple-query optimization, incremental maintenance, and vector index tradeoffs are prior art.

This is not a systems result. The current repository has no P14 optimizer, candidate enumerator, cost calibrator, connector statistics contract, materialization planner, adaptive executor, vector planner, or P14 benchmark run. It contains no evidence for global optimality or billion-row behavior. Every CatDB execution, calibration, plan-quality, adaptation, and scale claim in this manuscript is therefore OPEN, OBSTRUCTED, or UNRESOLVED.

Contents

1 Introduction

1.1 A cheap plan can still be inadmissible

Consider an analyst who asks for recent orders from reconciled customers, with a support tier and the nearest matching product description. Customer records are in PostgreSQL. Orders are in a rate-limited service. Support tiers come from a signed daily export. Product vectors sit in a local approximate index. A prior result is cached.

Several plans look plausible. PostgreSQL can join and aggregate locally. The service can accept batches or one key at a time. The cached result can answer most of the date range. The vector index can return candidates quickly. A materialized view can avoid repeated reconciliation. An optimizer could attach a latency estimate to each choice and pick the smallest one.

That is where the trouble starts. The cache predates a currency mapping change. The service has a concurrency budget that is more important than one analyst's response time. The signed export is fresh by file timestamp but old by source frontier. The approximate index misses some products after a metadata filter. A row-only comparison will not expose the lost provenance or source coverage. The cheapest row-producing plan may not satisfy the request.

The planning problem has two distinct stages. First decide whether a physical strategy preserves the requested result envelope. Only then compare its predicted outcomes with those of other admissible strategies. This ordering is the central discipline of the paper.

1.2 Research question

The research question is:

> Given a fixed semantic request and a finite set of physical alternatives, how should CatDB choose among federation, remote or local execution, caching, partial or full materialization, incremental refresh, full recomputation, and adaptive execution without trading away result semantics or hiding uncertain costs?

The bounded thesis is an ENGINEERING HYPOTHESIS:

> For a fixed logical query, semantic revision, result profile, policy bundle, execution environment, and finite candidate set, a future CatDB planner can reject infeasible strategies, predict a versioned vector of resource and service outcomes for each remaining strategy, and choose or abstain under an explicit objective. Every selected strategy must preserve the requested semantics, provenance, coverage, freshness, consistency, and approximation profile.

This thesis does not promise the globally optimal plan. Candidate enumeration is bounded, measurements are incomplete, sources change, and remote internals may remain hidden. The intended result is a replayable constrained decision, not an oracle over all possible executions.

1.3 Contributions and boundary

This provisional manuscript contributes:

1. a semantic request and full result-envelope model;
2. a feasibility predicate that precedes cost comparison;
3. a typed outcome vector with units, scope, uncertainty, and explicit exclusions;
4. a decision algebra for choice, probing, fallback, infeasibility, and abstention;
5. a decision certificate that ties each prediction to immutable statistics, capabilities, model revisions, and policies;
6. a unified decision space covering live federation, caching, materialization, incremental maintenance, shared work, adaptation, and vector search;
7. falsifiers, baselines, ablations, calibration tests, and adversarial workloads; and
8. a dependency-aware implementation roadmap for Rust and heterogeneous backends.

The manuscript does not report an optimizer, benchmark, global optimality result, or billion-row experiment. It does not claim that putting more terms in a cost equation is novel.

1.4 Organization

Section 2 fixes the evidence boundary. Section 3 reviews established work. Sections 4 to 6 define requests, candidates, and admission. Sections 7 and 8 define predicted outcomes and decisions. Sections 9 to 11 treat persistent state, adaptation, and vector search. Section 12 gives worked counterexamples. Section 13 specifies the future empirical test. Section 14 gives the implementation order. The paper ends with limitations, exact nonclaims, open questions, and reproducibility requirements.

2 Evidence vocabulary and repository status

2.1 Evidence labels

Label	Meaning
SUPPORTED BY PRIOR LITERATURE	A primary source establishes the stated result under its own model. It is not evidence that CatDB implements the result.
PAPER REASONING	A definition, calculation, or finite counterexample follows inside this paper. It is not a machine-checked theorem or runtime result.
ENGINEERING HYPOTHESIS	A falsifiable proposal for a future implementation or study.
OPEN	The claim has a specified evidence path, but that evidence does not exist in the repository.
OBSTRUCTED	A prerequisite semantic or systems contract is missing, so the claim cannot yet be tested honestly.
UNRESOLVED	The intended contract, fixture, or scale gate has not been fixed.
REJECTED AS WRITTEN	The wording cannot be supported and must be narrowed or removed.
NONCLAIM	An explicit statement that the paper does not establish the listed result.

2.2 Current repository audit

The repository contains schema, mapping, migration, and other prerequisite work owned by earlier papers. It does not contain a `catdb-stats`, `catdb-cost`, `catdb-planner`, `catdb-connector`, `catdb-materialization`, or `catdb-distribution` implementation for P14. There is no physical candidate language, capability conformance suite, statistics snapshot, cost-model revision, optimizer trace, predicted-versus- observed report, plan-choice benchmark, or vector planning adapter.

Layer	Status	Missing evidence
semantic request	OBSTRUCTED	Accepted P2 and P5 query/result profile
candidate plan space	OBSTRUCTED	Physical plan and connector contracts
feasibility predicate	OBSTRUCTED	Capability, policy, P6, P12, and P13 interfaces
typed cost vector	OPEN	Runtime observations and unit-checked model code

Layer	Status	Missing evidence
statistics provenance	OPEN	Snapshot schema, expiry, and telemetry records
calibration	OPEN	Training and held-out observation bundles
plan selection	OPEN	Enumerator, ranker, oracle, and traces
cache and materialization	OBSTRUCTED	P6 semantic and recovery contracts
adaptive execution	OPEN	Safe state transfer and replan runtime
vector choice	OPEN	Quality-aware connector and ground truth
backend execution	OBSTRUCTED	Heterogeneous connectors and result envelopes
benchmark evidence	OPEN	Preregistered runs and retained failures
semantic-feature novelty	UNRESOLVED	Physical-only and capability-only ablations
billion-row scale	UNRESOLVED	No physical billion-row run

The status of the manuscript is therefore OBSTRUCTED. Definitions below state a proposed contract. They do not promote any implementation claim.

3 Prior art and novelty boundary

3.1 Conventional and distributed optimization

System R established statistics-driven access-path and join-order selection [28]. Volcano and Cascades established extensible physical operators, properties, transformation rules, memoized alternatives, and guided optimizer search [12, 13]. Calcite provides a mature modular framework for heterogeneous adapters, metadata, rules, and cost-based planning [5]. End-to-end plan quality can depend sharply on cardinality estimates and cost-model behavior, which is why q-error alone is not a sufficient evaluation [20].

Distributed communication cost is also old. The R* validation study measured message, I/O, and CPU estimates while comparing relation shipping, fetch-matches, semijoins, and other distributed alternatives [21]. Garlic exposed source-specific capability rules inside a heterogeneous optimizer [15]. SemaGrow and later federated SPARQL work use source metadata, cardinality, communication, and placement [7, 27]. Trino exposes a current engineering baseline for connector statistics, join enumeration, distribution, pushdown, and adaptive plan changes [32].

Evidence status. SUPPORTED BY PRIOR LITERATURE. Cost-based access-path selection, distributed cost models, capability-aware pushdown, and adapter-based heterogeneous planning are not CatDB contributions..

3.2 Adaptation and feedback

Eddies route tuples adaptively as selectivities and rates change [3]. ANAPSID addresses endpoint delay and burstiness with adaptive federated operators [1]. Progressive optimization introduces runtime checkpoints and reoptimization [24]; LEO feeds execution observations back into future estimates [30]. Bao shows how a learned component can steer an existing optimizer while retaining the underlying engine [23]. Learned cardinality estimation provides a relevant baseline for correlated joins [18].
Evidence status. SUPPORTED BY PRIOR LITERATURE. Adaptation, feedback, learned estimation, and learned plan steering are established. P14 must test a semantic and evidence delta, not claim adaptation itself..

3.3 Caching, sharing, and materialization

Multiple-query optimization dates at least to Sellis [29]. Semantic caching, query containment, remainder queries, and replacement policies were studied explicitly by Dar et al. [8]. Materialized views can appear as ordinary costed rewrite alternatives [11]. DynaMat and later dynamic materialization work address workload-driven creation, refresh, storage, and retirement [19, 25]. Materialized-view maintenance has a broad taxonomy [14]; DBToaster and DBSP show sophisticated incrementalization approaches [2, 6].
Evidence status. SUPPORTED BY PRIOR LITERATURE. Caching, remainder queries, shared work, dynamic view management, and incremental maintenance are not new..

3.4 Freshness, provenance, and vector search

Continuous consistency and probabilistically bounded staleness show that freshness and consistency have multiple dimensions and guarantee levels [4, 34]. PERM and OneProvenance show that provenance can be compiled, extracted at different granularities, and measured as work rather than assumed free [10, 26].

Product quantization, HNSW, GPU search, DiskANN, Faiss, and Milvus expose different build, update, memory, I/O, latency, and quality regimes [9, 16, 17, 22, 31, 33]. Vector selection cannot be reduced to one operator cost.

3.5 Publication-safe novelty statement

The plausible P14 contribution is not a larger equation. It is the following testable integration:

> A versioned feasibility and decision-certificate contract in which semantic equivalence, source coverage, policy, freshness, consistency, provenance, and approximation determine admissibility; physical and service outcomes retain their units and uncertainty; and semantic statistics must show incremental value over physical and capability metadata after their own overhead is included.

That contribution remains UNRESOLVED until the physical-only, capability-aware, and semantic-feature ablations produce retained evidence. A null result must remain publishable.

4 Semantic requests and result envelopes

Definition 4.1 (Semantic request)**.** A planning request is the immutable tuple

$$\mathcal{R} = (Q, \Sigma, \Lambda, \Gamma, \Phi, \Omega, \mathcal{A}, W),$$

where:

- Q is a typed logical query;
- Σ is a schema and mapping revision bundle;
- Λ fixes null, bag, numeric, collation, conversion, temporal, and error semantics;
- Γ fixes source coverage, authority, and availability policy;
- Φ fixes freshness and consistency;
- Ω fixes provenance, reconciliation, retention, access, and disclosure;
- $\mathcal{A}$ fixes exactness, approximation, and vector quality; and
- W fixes the objective and resource budgets.

Evidence status. OBSTRUCTED by the P2 and P5 semantic profiles and the P13 provenance profile..

Definition 4.2 (Result envelope)**.** For admitted source state I, the denotation of a request is

$$\begin{aligned}\mathsf{Sem}(\mathcal{R}, I) = (&\text{Data}, \text{Provenance}, \text{Coverage}, \text{Freshness},\\ &\text{Consistency}, \text{Approximation}, \text{Diagnostics}).\end{aligned}$$

The data component includes the declared row or value semantics. The remaining components record obligations that cannot be inferred from visible rows.

Proposition 4.3 (Visible-row equality is insufficient)**.** *There exist two results with equal visible data bags that do not satisfy the same semantic request.*

Proof. Let two results contain the same visible row. Let the first result include all required sources at the requested frontiers and row provenance. Let the second omit one required source, use an older mapping revision, or omit the required provenance. Their Data components are equal on this instance, but at least one other component of the result envelope differs. Therefore the second result does not satisfy the same request. □

Evidence status. PAPER REASONING. This proposition follows from the result-envelope definition. It is not a runtime theorem..

Requirement 4.4 (Immutable semantic identity)**.** The request identity changes whenever a component that can change denotation, admissibility, or required evidence changes. A query text hash alone is not a semantic request identity.

At minimum, the identity covers query, schema, mapping, source, scalar semantics, coverage, authority, freshness, consistency, provenance, reconciliation, disclosure, approximation, and vector-model revisions.

5 Candidate plan space

Definition 5.1 (Candidate plan)**.** A candidate physical plan is

$$p = (L, \text{placement}, \text{exchange}, \text{cache}, \text{materialization}, \text{refresh}, \text{provenance}, \text{vector}, \text{recovery}),$$

where L is a physical operator graph. Each component refers to immutable logical, semantic, connector, capability, statistic, cost-model, policy, and generated-fragment revisions.

Evidence status. OPEN. No such plan representation exists in the current P14 repository surface..

5.1 Virtual federation

A virtual plan reads sources during the query and combines remote and local operators. Its inputs include request setup, authentication, pushed capabilities, output cardinality and width, round trips, transfer, local residual work, source pressure, snapshot behavior, transformations, reconciliation, provenance, and coordination.

Virtual execution avoids persistent-copy maintenance. It still pays network, remote work, source availability, partial failure, and reconciliation costs.

5.2 Remote computation

A remote-compute plan pushes filters, projections, aggregates, joins, Top-N, or vector search to a source and returns a reduced result. It is admissible only when the connector declares the operation and the generated fragment preserves the request's semantic profile.

"Push down as much as possible" is not a decision rule. A remote join can expand its result. A remote collation can disagree with the canonical comparison. A pushed aggregate can consume a source's concurrency budget even when it lowers user latency.

5.3 Local execution after transfer

A local plan transfers base or partially reduced data and executes under locally controlled operators. It may offer predictable semantics, better instrumentation, reusable intermediate state, or protection for a constrained source. It may also incur large transfer, violate residency, lose remote indexes, increase local memory and I/O, delay the first row, or expose snapshot skew. Local execution is a costed candidate, not a universally safe fallback.

5.4 Cache and materialization alternatives

A semantic-cache candidate can be an exact hit, a contained hit plus a remainder query, a union of compatible regions, or a miss. Its compatibility key covers query region, semantic revisions, source identity and frontiers, scalar behavior, reconciliation, provenance, retention, disclosure, approximation, embedding model, and storage format.

A partial materialization covers declared sources, predicates, partitions, columns, aggregates, semantic mappings, provenance profiles, or time windows. The plan must identify uncovered regions and provide a completion plan when the request requires complete results.

A full materialization stores a declared result envelope at a declared source frontier. The semantic definition remains authoritative. The stored object is a replaceable physical artifact.

5.5 Refresh and recomputation

Incremental refresh is a candidate only for a P6-admitted query and change fragment. Its cost includes delta capture, deduplication, ordering, state, indexes, provenance, checkpoint publication, compaction, replay, and semantic invalidation. Full recomputation remains both a competing plan and the reference result for maintenance correctness.

5.6 Shared and adaptive execution

Transient materialization can share scans, joins, aggregates, transformations, reconciliation, or provenance among queries. The plan must account for batching delay, interference, fairness, cancellation, cleanup, snapshot compatibility, and disclosure.

An adaptive plan can change routing, join order, placement, vector effort, or materialization after an observation. It must specify its trigger, replacement set, state-transfer semantics, treatment of already emitted results, snapshot preservation, provenance continuity, and added decision overhead.

Strategy	Required semantic evidence	Distinct cost exposure
virtual federation	source coverage, exact/residual pushdown, snapshots	remote calls, transfer, source load, partial failure
remote compute	connector capability and scalar equivalence	remote CPU/I/O, expanded output, service pressure
local execution	movement permission and snapshot interpretation	transfer, serialization, local CPU/I/O/memory
semantic cache	containment and complete compatibility key	lookup, remainder, invalidation, storage, staleness
partial materialization	region coverage and valid completion	mixed live and stored work, uncovered region, provenance merge
full materialization	definition and frontier identity	build, refresh, storage, recovery, serving
incremental refresh	admitted delta and checkpoint contract	delta capture, state, index, provenance, replay
full recomputation	full-read and swap contract	source read, rebuild, barrier, replacement
shared transient result	compatible policy and snapshot profiles	wait, interference, spill, cancellation
adaptive replacement	state and result continuity	observation, replan, state transfer, duplicate prevention

6 Feasibility before cost

Definition 6.1 (Feasibility)**.** For request $\mathcal{R}$, plan p, and environment E, $\mathsf{Feasible}(p, \mathcal{R}, E) = \mathsf{true}$ only if:

1. p is well typed;
2. every rewrite and lowering is admitted by the semantic profile;
3. every pushed fragment is supported by its connector;
4. unsupported work remains an explicit residual or stable error;
5. source coverage and availability policy are satisfied;
6. access, movement, residency, disclosure, and retention policy permit each operation;
7. the freshness envelope satisfies Φ;
8. the implementation satisfies the named consistency profile;
9. the provenance path can produce Ω;
10. exact and approximate operators satisfy $\mathcal{A}$;
11. cache and materialization dependencies match;
12. snapshot, cancellation, and recovery obligations can be honored; and
13. known hard resource limits are not exceeded.

Unknown capability, cache compatibility, or policy satisfaction is not true. The permitted states are true, false, or a structured unknown that requires a probe, conservative fallback, infeasibility result, or abstention.
Evidence status. OBSTRUCTED by P2, P5, P6, P12, P13, and connector contracts..

Proposition 6.2 (Hard constraints cannot be repaired by a finite penalty)**.** *If a request declares a property as a hard constraint, assigning its violation a finite scalar penalty does not preserve that constraint for arbitrary objective weights.*

Proof. Let the violating plan have physical score a, let the admissible plan have score $b > a$, and let the violation penalty be finite M. Any rescaling or objective composition that makes the physical difference exceed M ranks the violating plan first. Filtering the violating plan before ranking avoids this dependence on scale. □

Evidence status. PAPER REASONING. The result is a consequence of the declared two-stage decision rule..

6.1 Plan equivalence

Two plans may be compared as alternatives only within the same request:

$$p_1 \equiv_{\mathcal{R}} p_2 \iff \forall I \in \text{Admitted}(\mathcal{R}), \quad \mathsf{Norm}(\mathsf{Run}(p_1, I)) = \mathsf{Norm}(\mathsf{Run}(p_2, I)).$$

For exact requests, equality covers the full normalized result envelope. For approximate requests, both plans must satisfy the same declared quality contract. General plan equivalence is undecidable in unrestricted languages; the initial system therefore needs a bounded language, reviewed rewrites, connector conformance, finite fixtures, differential tests, selected formal results, and explicit unsupported outcomes.

6.2 Hard constraints and soft objectives

Result semantics, source coverage, authorization, residency, disclosure, freshness bounds, consistency, required provenance, minimum vector quality, and memory safety are hard unless the request explicitly selects another profile. After admission, soft objectives may include first-row and total latency, tail risk, throughput, transfer, calls, local and remote work, source pressure, storage, refresh work, failure exposure, uncertainty, and plan simplicity. The request must define their ordering.

7 Typed outcomes, units, and exclusions

Definition 7.1 (Predicted outcome vector)**.** For request features x, environment E, statistics S, and model revision θ,

$$\widehat{\mathbf{Y}}(p) = \mathsf{Predict}_\theta(p, x, E, S)$$

contains

$$(\widehat{T}_{\text{first}}, \widehat{T}_{\text{total}}, \widehat{B}_{\text{net}}, \widehat{B}_{\text{read}}, \widehat{C}_{\text{local}}, \widehat{C}_{\text{remote}}, \widehat{M}_{\text{peak}}, \widehat{L}_{\text{source}}, \widehat{F}, \widehat{K}, \widehat{P}, \widehat{A}, \widehat{R}).$$

Each component has a natural unit, scope, statistic lineage, and uncertainty description.

The vector covers first-row and total latency, network and storage traffic, local and remote compute, peak memory, source load, freshness, consistency and coordination, provenance, approximation quality, and failure or retry outcomes. It is not collapsed into a universal unitless score. A scalar utility is valid only when W supplies a versioned and reviewable conversion.

Evidence status. OPEN. No calibrated P14 outcome model exists..

7.1 Latency and transfer

For physical graph G_p, a first-order latency decomposition is

$$T_{\text{total}}(p) = T_{\text{plan}} + T_{\text{admit}} + T_{\text{setup}} + T_{\text{critical}}(G_p) + T_{\text{finalize}}.$$

Parallel branches are represented by the critical path, not blindly summed. Queueing, remote service, transfer, local execution, transformation, reconciliation, provenance, coordination, spill, retry, and materialization work appear on the path when applicable.

For exchange edge e,

$$T_{\text{net}}(e) \approx T_{\text{connect}}(e) + N_{\text{rt}}(e)T_{\text{rtt}}(e) + \frac{B_{\text{wire}}(e)}{\text{Throughput}(e)} + T_{\text{ser}}(e) + T_{\text{de}}(e).$$

The approximation is calibrated by payload, concurrency, protocol, compression, region, and time window. A throughput constant is not exported outside its calibration domain without widened uncertainty or a probe.

7.2 Source load

For source s, the planner retains

$$\begin{aligned}\mathbf{L}_s(p) = (&\text{calls}, \text{concurrency}, \text{queue\_ms}, \text{cpu\_ms}, \text{io\_bytes},\\ &\text{rows\_scanned}, \text{bytes\_returned}, \text{writes}, \text{rate\_limit\_risk}).\end{aligned}$$

Source load can be a hard cap, an ordered objective, or a priced resource under a supplied conversion. It does not disappear inside user latency. When a source does not expose CPU or I/O, those fields remain unknown or bounded. They are never inferred as exact values from response time.

7.3 Transformation, reconciliation, and provenance

An initial transformation family for named class g is

$$C_{\text{transform}}(g) = \alpha_g + \beta_g n_{\text{in}} + \gamma_g b_{\text{in}} + \delta_g n_{\text{out}}.$$

This is a calibratable family, not a law. Reconciliation includes candidate generation, evidence lookup, scoring, decision, conflict handling, and provenance. Mapping and reconciliation features are recorded by named operator and semantic revision.

For provenance profile ω,

$$\mathbf{C}_{\text{prov}}(p, \omega) = (T_{\text{capture}}, T_{\text{normalize}}, T_{\text{persist}}, T_{\text{query}}, B_{\text{wire}}, B_{\text{store}}, N_{\text{tokens}}, N_{\text{nodes}}, N_{\text{edges}}).$$

Artifact references, row lineage, value origin, how-provenance, and reconciliation evidence are different profiles. Plans with unequal provenance profiles are not equivalent baselines.

7.4 Freshness and consistency

Freshness is a vector:

$$\begin{aligned}\mathbf{F} = (&\text{source\_age}_i, \text{frontier\_lag}_i, \text{definition\_age}, \text{mapping\_age},\\ &\text{reconciliation\_age}, \text{provenance\_profile\_age}, \text{materialization\_age}).\end{aligned}$$

A result may be fresh by materialization time and stale by mapping or source frontier. Wall-clock age records clock assumptions and uncertainty. Version lag is separate.

Consistency profiles are categorical contracts such as one-source snapshot, per-source snapshots with visible skew, a causally compatible frontier, bounded staleness, serializable materialization refresh, or best effort with explicit partial metadata. Their wait, message, retry, lock, and availability costs are measurable. The profile itself is not a millisecond penalty.

7.5 Core units

Dimension	Unit	Observation boundary
planning time	ns or ms	parse end to executable plan publication, by phase
candidate counts	count	generated, feasible, pruned, and costed
first and total latency	ms	user request boundary
queue and setup	ms	per site and remote call
calls and round trips	count	per source and operation
network payload	bytes	each direction, compressed and raw if visible
serialization	CPU ns, bytes	encode and decode separately
source work	rows, bytes, CPU ns, I/O ops	actual values when exposed
local work	rows, bytes, CPU ns, I/O ops	per physical operator
memory	bytes, byte-ms	peak and occupancy definition
temporary storage	byte-ms	spill and transient results
persistent storage	byte-seconds	data, indexes, provenance, metadata
transformation	rows, bytes, CPU ns	per named mapping operator
reconciliation	candidates, comparisons, CPU ns	evidence and conflicts included
provenance	CPU ns, bytes, graph counts	capture, persist, normalize, query
coordination	messages, waits, bytes	named mechanism
freshness	ms and versions behind	per source and artifact
cache coverage	fraction plus region	hit, remainder, stale, rejected
refresh	rows, bytes, CPU ns, I/O	delta and full separately
failures and retries	count, added ms	stable class and retry policy
vector effort	visits, probes, distances	index-specific parameters
vector quality	recall@k, precision@k, ratio	exact or declared reference

7.6 Excluded quantities

Semantic correctness, authorization, residency, disclosure, and required quality remain feasibility conditions. Human reconciliation effort, business value, money, energy, and carbon

are excluded until a measured and versioned conversion exists. Source truth or reputation is not inferred from provenance. Unobserved remote work remains unknown. Availability probability requires a calibrated failure taxonomy. Timed-out executions remain censored. Vector proximity does not prove identity. Global optimality is excluded under incomplete information and bounded enumeration.

8 Statistics, uncertainty, and decisions

8.1 Statistic records

Every statistic used by a decision records:

- statistic ID, type, value, unit, and uncertainty;
- source, connector, relation, column, expression, or operator scope;
- schema, mapping, reconciliation, provenance, and capability revisions when relevant;
- observation method, sample, time window, and source frontier;
- environment and workload domain;
- expiry and invalidation rules; and
- whether the value was measured, sampled, declared, inferred, or bounded.

Physical statistics include cardinality, distinct counts, null fractions, histograms, correlations, widths, index properties, queue state, transfer curves, and operator observations. Semantic statistics may include mapping selectivity, conversion expansion, reconciliation candidate fan-out, accepted-match fan-out, provenance expansion, cache compatibility frequency, invalidation rate, and residual selectivity. The semantic set is optional research input until its benefit is demonstrated.

8.2 Capability manifests

A connector capability is a versioned contract, not a Boolean `supports_pushdown`. It names admitted operators, scalar semantics, limits, null and collation behavior, parameter limits, snapshot and cancellation behavior, statistics availability, instrumentation, rate limits, and known failure classes. Unknown behavior remains unknown.

8.3 Decision algebra

Definition 8.1 (Planner result)**.** For finite candidate set $\mathcal{P}$,

$$\mathsf{Decide}(\mathcal{R}, \mathcal{P}, S, E, \theta) \in \{\mathsf{Choose}(p, c), \mathsf{Probe}(g), \mathsf{Fallback}(h), \mathsf{NoFeasible}(d), \mathsf{Abstain}(u)\}.$$

Here c is a certificate, g a bounded observation plan, h a named conservative strategy, d an infeasibility diagnosis, and u an uncertainty diagnosis.

A proposed default objective is lexicographic:

1. admit no known semantic or policy violation;
2. satisfy freshness, consistency, quality, and resource limits;
3. minimize the probability of missing the latency objective;
4. minimize predicted p95 total latency;
5. minimize source pressure;
6. minimize transfer and local resource use; and
7. prefer the simpler and better-observed plan inside a declared indifference region.

This default is an ENGINEERING HYPOTHESIS. Another request may supply another versioned ordering. Hidden weights are prohibited. Source pressure with a nonlinear failure boundary is not left as a soft fifth-place preference: a declared rate-limit, concurrency-pool, quota, or saturation threshold is a hard resource limit in step 2. Plans that may cross it are rejected, probed, or assigned a policy-approved fallback before latency ranking.

8.4 Probe, fallback, and abstention

A probe is bounded by calls, rows, bytes, time, source load, and disclosure. Its expected information gain cannot override feasibility. Examples include a remote `EXPLAIN`, sampled cardinality query, small transfer calibration, capability handshake, or exact vector sample.

A fallback is named and reviewed. It might use local exact execution, disable sharing, require a fresh rebuild, choose a conservative join distribution, or return an explicit unsupported result. "Use the default" is not a reproducible fallback.

Abstention is correct when feasible candidates cannot be distinguished safely inside the uncertainty and objective contract. It returns the missing statistics, stale models, out-of-domain features, or conflicting objectives that prevented a choice.

8.5 Decision certificate

Each $\mathsf{Choose}(p, c)$ result records:

- request and semantic-profile digests;
- feasible and rejected candidate IDs, with every rejection reason;
- predicted outcome vector and intervals for each finalist;
- objective profile and hard constraints;
- statistic, capability, and telemetry references;
- model revision and calibration-domain label;
- winning criterion and runner-up delta;

- probes, fallback paths, and out-of-distribution signals;
- cache or materialization compatibility evidence;
- approximation and quality requirement;
- source-load budget and adaptive checkpoints;
- expiry conditions; and
- a stable replay explanation.

The certificate explains a decision. It does not prove that the chosen plan will be fastest.

Proposition 8.2 (Safe dominance)**.** *For two feasible plans under the same request, plan p_1 dominates p_2 only if p_1 is no worse on every ordered objective and strictly better on at least one, with uncertainty handled by the declared comparison rule.*

Proof. This is the Pareto dominance relation restricted to one semantic equivalence class and one objective profile. The restriction prevents a cheaper but weaker result envelope from dominating an admissible plan. □

9 Planning over a materialization horizon

9.1 Horizon cost

Let H be a decision horizon, Q_H the query arrivals, and U_H the data and semantic changes. The realized materialization cost is

$$\begin{aligned} C_H(m) = {} & C_{\text{build}}(m) + C_{\text{serve}}(Q_H, m) + C_{\text{maintain}}(U_H, m) \\ & + C_{\text{invalidate}}(U_H, m) + C_{\text{storage}}(m, H) + C_{\text{recovery}}(m, H) + C_{\text{staleness}}(m, H). \end{aligned}$$

The staleness term exists only when the request supplies a legitimate loss function. Otherwise staleness remains a hard constraint.

The comparison is

$$\text{NetBenefit}_H(m) = C_H(\text{best nonmaterialized}) - C_H(m).$$

The horizon, arrivals, updates, initial cache state, and terminal value are declared. Stationarity is not assumed.

9.2 Incremental versus full refresh

For change batch Δ,

$$C_{\text{inc}}(\Delta) = C_{\text{capture}} + C_{\Delta\text{eval}} + C_{\text{state}} + C_{\text{index}} + C_{\text{prov}} + C_{\text{checkpoint}} + C_{\text{recovery risk}},$$

while

$$C_{\text{full}} = C_{\text{source read}} + C_{\text{recompute}} + C_{\text{prov rebuild}} + C_{\text{swap}} + C_{\text{checkpoint}}.$$

Incremental refresh is selected only when P6 admits it and the constrained objective favors it. Delta size alone is insufficient.

9.3 Adaptive materialization state

A future policy may move an artifact through

$$\mathsf{Absent} \to \mathsf{Building} \to \mathsf{Ready} \to \mathsf{Refreshing} \to \mathsf{Stale} \to \mathsf{Retiring} \to \mathsf{Absent},$$

with explicit Failed and Blocked states. Admission uses reuse probability, build cost, maintenance, storage, invalidation, recovery, policy compatibility, and uncertainty. Retirement considers future value and deletion obligations. Exploration is bounded and semantically feasible. Its traffic is labeled and included in overhead.
Evidence status. OBSTRUCTED by P6. No materialization policy or result exists for P14..

10 Adaptive execution

An adaptive checkpoint observes actual cardinality, rates, queueing, memory, spill, source failures, transfer, or vector candidate quality. A replacement plan is admitted only under the same request. The checkpoint records which rows have been emitted, which source frontiers were observed, which state can be reused, and how provenance continues across the boundary.

Requirement 10.1 (Adaptive continuity)**.** For an exact request, adaptation must not duplicate, omit, or reorder results contrary to the request. It must not silently change source coverage, freshness, consistency, provenance, disclosure, or approximation.

Feedback used for future estimates is separated from confirmatory evaluation. Training observations carry workload time, source, query template, semantic revision, and environment. Held-out executions are not fed back before their reported predictions are frozen.
Evidence status. OPEN. Adaptive query processing is prior art, but no CatDB state-transfer or feedback-isolation implementation exists..

11 Vector-aware planning

A vector candidate records distance and normalization, embedding model and revision, count and dimension, exact or approximate index, parameters, quantization, reranking, filter placement, candidate count, distance evaluations, build and update behavior, memory, persistent bytes, latency, throughput, quality, exact verification, and stale-embedding policy.

For a probabilistic quality claim,

$$\Pr(\mathsf{Quality}(p) \geq q_{\min}) \geq 1 - \epsilon$$

is valid only when calibrated for the declared population. Otherwise the planner reports sample evidence and an out-of-domain label. Exact flat search, HNSW, IVF, product quantization, GPU search, and disk-based ANN are different physical families, not settings of one abstract latency coefficient.

Metadata filtering requires particular care. Prefilter, integrated-filter, and postfilter strategies may have different recall. Quality is measured against filtered ground truth. The exact reference applies the identical metadata predicate first and then computes exact

distances over precisely that eligible population; an unfiltered exact search is not a valid oracle for a filtered request. A recent exact delta may be required when the index frontier lags the source.
Evidence status. OPEN. The repository contains no P14 vector connector, ground-truth bundle, or calibrated selection result. No vector result is evidence of semantic identity..

12 Worked examples and falsifiers

12.1 Running order query

Return the last 24 hours of accepted orders for reconciled customers, attach a support tier, and find the nearest canonical product description. The request requires all three authoritative sources, source age below five minutes for orders, row provenance, a causally compatible customer and tier frontier, and recall@10 of at least 0.98 for vector candidate generation followed by exact verification.

Candidate p_1 pushes customer filtering and aggregation to PostgreSQL, batches the order service, joins the signed tier export locally, and uses the local vector index. Candidate p_2 reuses a result cached yesterday and issues a remainder query. Candidate p_3 reads a partial materialization and completes the recent window from live sources.

The cached data in p_2 looks right on the hand-worked rows, but the cache predates a cents-to-dollars mapping correction and contains artifact-level provenance rather than row provenance. It is infeasible and never reaches latency comparison. Plan p_3 has the lowest median estimate, but its materialization frontier is unknown after a failed refresh. It requires a probe or rebuild. Only p_1 is presently admissible, though it can still lose its latency objective. The decision certificate records that distinction.

12.2 Correlated predicates

Counterexample 12.1. Two individually selective predicates are strongly correlated. An independence estimate underpredicts a join input by several orders of magnitude and chooses a remote nested loop.

The required behavior is to retain estimate provenance, expose correlation uncertainty, compare a conservative or learned estimator, and permit a probe or adaptive checkpoint. The claim is falsified if reported prediction intervals repeatedly exclude the actual cardinality on held-out correlated queries.

12.3 Remote join expansion

Counterexample 12.2. A source supports join pushdown, but a many-to-many remote join returns more bytes than transferring one filtered input and joining locally.

Capability support cannot force placement. The planner enumerates both plans, estimates output width and cardinality, includes source CPU and returned bytes, and keeps remote scalar semantics visible.

12.4 Rate-limited service

Counterexample 12.3. An API returns few rows but requires one request per key. A single query is fast; concurrent traffic triggers throttling and retries.

Returned bytes are a poor proxy for remote work. The source profile includes calls, batching, round trips, concurrency, queueing, rate-limit risk, retry, and the operational source objective. A low-latency plan can be infeasible because it harms the source.

12.5 Fast but stale cache

Counterexample 12.4. A cached result matches the current fixture rows but predates a mapping revision and lacks the required provenance profile.

The cache is rejected with both incompatible revisions named. Rebuild, live execution, or a stable error remain candidates. SQL text and parameters are not a complete cache key.

12.6 Update burst crossover

Counterexample 12.5. Incremental refresh wins for small deltas. A burst then changes most join keys, and index plus provenance maintenance costs more than full recomputation.

The model compares incremental, partial, and full refresh using delta size, fan-out, state, random I/O, and provenance. A static "incremental" flag is a falsifier.

12.7 Identity and provenance expansion

Counterexample 12.6. A reconciliation revision splits one canonical entity into two. Output rows remain few, but join fan-out and alternative provenance derivations grow sharply.

Statistics are bound to the reconciliation revision. Candidate and accepted fan-out, evidence lookup, conflict work, and provenance structure are measured. A fixed per-row semantic multiplier is inadequate.

12.8 Consistency barrier

Counterexample 12.7. Remote scans are fast, but one lagging source blocks the common frontier required by the request.

The plan includes barrier wait and failure behavior. A policy-permitted older or partial frontier can be a different request, but it is not silently substituted for this one.

12.9 Filtered vector recall

Counterexample 12.8. ANN search runs before a metadata filter. Relevant filtered neighbors never enter the candidate set, even though unfiltered recall is high.

The benchmark uses filtered ground truth, compares filter placement, adjusts search effort, and retains an exact fallback. That fallback applies the same metadata predicate before exact distance calculation over the eligible population. Unfiltered recall cannot certify the filtered query.

12.10 Planning-time explosion

Counterexample 12.9. An eleven-way heterogeneous join with placement, cache, and materialization alternatives takes longer to optimize than a conservative plan takes to run.

Planning obeys a time and candidate budget. The trace records pruning, termination, fallback, and regret against deeper search. Advertised latency includes planning.

12.11 Negative semantic-feature result

Counterexample 12.10. Collecting mapping selectivity, reconciliation fan-out, and provenance expansion costs more than it saves on a stable simple workload.

The physical-only baseline remains available. The full treatment includes collection, storage, invalidation, and inference. A null or negative semantic feature result is reported.

13 Evaluation and calibration protocol

13.1 Research questions

The confirmatory study asks:

1. Does feasibility filtering prevent semantic and policy violations that cost-only baselines admit?
2. Are cardinality, latency, transfer, source load, provenance, refresh, and vector-quality predictions calibrated on held-out workloads?
3. Does the chosen plan have lower constrained regret than a simple baseline over the enumerated candidate set?
4. Do semantic features add value beyond physical statistics and connector capabilities after metadata overhead?
5. When do cache, partial/full materialization, incremental refresh, or shared work cross over against live execution?
6. Can probes, fallback, and abstention reduce harmful confident choices under drift?

The primary null hypotheses are that feasibility filtering does not reduce violations, predictions are not acceptably calibrated, plan ranking does not beat the named baseline, and semantic features provide no net improvement.

13.2 Baselines and treatments

System or treatment	Required comparison role
native backend plans	backend-local execution and optimizer behavior

System or treatment	Required comparison role
R*-style reconstruction	communication-aware distributed alternatives using documented reconstruction assumptions
Calcite	modular heterogeneous rules, metadata, and cost planning
Trino	connector statistics, pushdown, join distribution, fallback, and adaptive behavior
physical-only CatDB	physical statistics without semantic features
capability-aware CatDB	physical statistics plus connector capabilities
full semantic CatDB	capabilities plus mapping, reconciliation, provenance, freshness, and consistency features
oracle over enumerated plans	best observed feasible candidate, computed only where all required candidates can be executed
conservative fallback	fixed reviewed plan chosen without learned ranking

A historical-system reconstruction does not inherit published performance. Every reconstruction records source revision, deviations, and exclusions.

13.3 Ablations

The ablation ladder removes one class at a time:

1. physical statistics only;
2. add connector capability metadata;
3. add source-load state;
4. add mapping and transformation features;
5. add reconciliation fan-out;
6. add freshness and consistency features;
7. add provenance features;
8. add cache and materialization state;
9. add vector-quality features;
10. add uncertainty, probing, and abstention; and
11. add feedback or learned ranking.

Each feature class includes its collection, storage, invalidation, planning, and execution overhead. A feature is not credited when it merely shifts cost outside the measured boundary.

13.4 Workload

The workload spans single-source, two-source, and heterogeneous many-source topologies. Query shapes include selections, projections, aggregates, star and chain joins, skewed many-to-many joins, Top-N, cache containment, materialized completion, incremental refresh, reconciliation, provenance expansion, and filtered vector search.

Distributions vary selectivity, correlation, skew, width, update rate, fan-out, network conditions, queueing, source limits, cache state, semantic revision, frontier lag, failure, and vector density. Workload dynamics include stable, step-change, burst, drift, and recovery periods. Policy cases include movement denial, source-load caps, exactness, minimum vector quality, required provenance, and bounded freshness.

Physical data scales begin with hand-solvable fixtures, then controlled 10^4, 10^6, and 10^8 row inputs where the backend and query permit. No scale is inferred from logical expansion or extrapolation. Billion-row language requires an actual physical billion-row run for every named workload and backend. This paper contains no such evidence.

13.5 Correctness first

For each request, an independent reference evaluator produces normalized data, provenance, coverage, freshness, consistency, approximation, and diagnostic expectations at the same source frontier. Candidate plans are admitted before timing. Warmup and measured repetitions still rerun correctness checks, because cache, adaptation, and refresh bugs can be state-dependent.

An oracle over plans is computed only when every candidate in the declared set can be executed under the same request. Failed and timed-out candidates remain observations. Timeouts are right-censored, not rewritten as successful executions at the timeout value.

13.6 Calibration and ranking

Cardinality reports include q-error, signed log error, interval coverage, error by operator and template, and tail behavior. Latency and resource models report absolute and relative error, interval coverage, calibration curves, and failure-class confusion where applicable. Plan ranking reports top-1 accuracy, pairwise order accuracy, constrained regret, tail regret, and policy-violation rate.

Probabilistic quality or deadline claims use reliability diagrams and proper scores. Vector plans report recall@k, filtered recall, latency, throughput, memory, build, update lag, and exact-verification cost. The reference result uses the request's identical metadata predicate before exact distance calculation; otherwise filtered recall is not reported. Predictions are frozen before held-out outcomes enter feedback.

13.7 Experimental discipline

Runs use paired randomized blocks over fixed request and environment states. The protocol declares warmup, repetition count, seeds, cache state, connector versions, source frontiers, failure injection, timeouts, exclusions, and stopping rules. Reports include intervals and

practical-effect thresholds, not only significance tests. A confirmatory model and analysis are frozen before the retained evaluation set is executed.

13.8 Acceptance gates

A future empirical claim requires:

1. zero unreported result-envelope mismatches;
2. zero silent policy weakening;
3. complete candidate and decision certificates;
4. no missing units or fabricated remote metrics;
5. held-out calibration inside preregistered tolerances;
6. constrained-regret and violation reporting against all named baselines;
7. explicit metadata overhead;
8. retained failures, timeouts, and censored observations;
9. separate vector quality and latency;
10. source-load measurements where exposed;
11. source, query-template, and temporal holdouts; and
12. immutable raw artifacts sufficient to replay one full decision.

Evidence status. OPEN. No P14 benchmark was executed for this paper..

14 Implementation roadmap

14.1 Dependency order

P14 depends on:

- P2 for physical compilation, semantic profiles, residual operators, and plan identity;
- P5 for federation, coverage, connector capability, snapshots, partial failure, and result envelopes;
- P6 for materialization definitions, incremental admissibility, checkpoints, invalidation, and recovery;
- P12 for routing and placement constraints;
- P13 for provenance profiles and measurable provenance operators; and
- connector and backend slices for observable execution.

These dependencies are contracts, not package names to invent prematurely. P14 remains blocked until a finite profile is frozen.

14.2 Staged Rust plan

Stage	Deliverable	Exit condition
I0	freeze profile	accepted finite request, plan, capability, result, and objective contracts
I1	fixtures and oracle	six to ten hand-solvable plan sets with independent expected feasibility and cost calculations
I2	statistics	typed statistic records, revision binding, expiry, invalidation, and deterministic serialization
I3	cost model	unit-checked outcome components, uncertainty, unknown values, and observation lineage
I4	planner	finite enumeration, feasibility filter, objective ordering, probe/fallback/abstain, and decision certificate
I5	connector telemetry	SQLite, PostgreSQL, and service capability conformance with stable unsupported behavior
I6	cache/materialization	exact compatibility, completion plans, horizon costs, refresh alternatives, and invalidation
I7	adaptation	checkpoints, safe replacement, state transfer, feedback isolation, and replay
I8	vector profile	exact and approximate alternatives, quality contract, filter placement, update frontier, and ground truth
I9	benchmark harness	immutable inputs, randomized blocks, raw observations, analysis, and report generation
I10	hardening boundary	crash, cancellation, concurrency, resource caps, security, observability, and upgrade behavior

14.3 Backend measurement plan

SQLite supplies a deterministic local baseline for scans, joins, indexes, sorts, aggregates, memory, spill, and materialization files. PostgreSQL adds server-side statistics, plans, CPU and I/O observations where available, concurrency, cancellation, snapshots, and pushdown. An HTTP or service connector exposes batching, setup, pagination, rate limits, queueing, retries, timeouts, and incomplete remote metrics. Heterogeneous federation combines these with transfer, reconciliation, provenance, consistency, and partial failure.

The materialization backend records build, serve, refresh, invalidate, recover, and retire paths. A vector backend records index construction, updates, deletes, compaction, search parameters, quality, memory, persistent bytes, filtered behavior, and exact verification.

14.4 Formalization boundary

Suitable early formal targets are finite-plan feasibility filtering, lexicographic decision determinism, unit-tag preservation, cache-key dependency monotonicity, dominance, and decision-

certificate replay. Formalization should not attempt to prove calibrated latency, connector behavior, vector recall on unseen data, production recovery, or global optimizer optimality.

A small Lean model would verify only its declarations and assumptions. It would not verify the Rust runtime or calibrate a cost model.

15 Limitations

The contract is much easier to state than to implement. General query equivalence is unavailable, connectors rarely expose complete semantics, and remote services can hide the work that matters most to their operators. Even a careful outcome vector depends on calibration domains that can drift. Probing consumes the same source budgets the planner is trying to protect.

The finite candidate set is another hard boundary. A plan can be best among the enumerated alternatives and still be poor. Search pruning may remove the eventual winner. An oracle computed by executing every candidate is practical only for small or controlled workloads, and winner selection on noisy runs creates its own bias.

Materialization requires a workload horizon that is never known exactly. Semantic changes can invalidate both stored results and the statistics used to choose them. Incremental refresh can lose to recomputation after fan-out, random I/O, provenance, checkpoint, and recovery are counted. The current paper offers no implementation evidence for any of those decisions.

Freshness and consistency remain profile-specific. A probabilistic staleness model is not a deterministic guarantee. Per-source timestamps do not prove a common snapshot. Provenance adds work whose size depends on representation and granularity. Vector quality depends on the population, filter, embedding, index, update frontier, and ground truth.

The most important experimental risk is a null semantic-feature result. Physical statistics and capability metadata may explain nearly all plan quality in the tested workloads. Collecting mapping, reconciliation, and provenance features may cost more than they save. That outcome would narrow the paper to a feasibility and evidence contract. It would not justify hiding the ablation.

Finally, this author pass has no independent paper review. AGY was deliberately not invoked. The manuscript should not be treated as reviewed or publication ready.

16 Exact nonclaims

The following statements are binding on this provisional manuscript:

1. CatDB does not currently implement the optimizer specified here.
2. This manuscript contains no CatDB benchmark result.
3. CatDB does not invent cost-based query optimization.
4. CatDB does not invent distributed or federated cost models.
5. CatDB does not invent capability-aware pushdown.

6. CatDB does not invent adaptive query processing or feedback-based optimization.
7. CatDB does not invent semantic caching, remainder queries, materialized views, multiple-query optimization, dynamic view management, or incremental view maintenance.
8. A larger cost equation is not by itself a research contribution.
9. The proposed planner does not guarantee global optimality under incomplete information or bounded enumeration.
10. A chosen plan is not guaranteed to be the fastest realization.
11. Cost prediction does not prove semantic correctness.
12. Identical visible rows do not prove equal provenance, freshness, consistency, source coverage, policy, or approximation.
13. Semantic correctness, authorization, residency, disclosure, freshness, consistency, provenance, and approximation are not traded away by low estimated latency.
14. Pushdown does not always reduce movement, latency, or source load.
15. Federation does not remove network, remote compute, source availability, coordination, or partial-failure costs.
16. Caching and materialization do not remove invalidation, refresh, storage, recovery, or staleness costs.
17. Incremental refresh is not always cheaper than full recomputation.
18. Provenance is not free, and one reported overhead does not generalize across profiles.
19. Freshness is not one cache timestamp.
20. A probabilistic staleness estimate is not a deterministic guarantee.
21. Vector similarity does not establish identity or semantic equivalence.
22. Approximate vector latency is not compared across unequal quality contracts as if it were equivalent.
23. Connector-reported statistics are not assumed complete, current, or accurate.
24. Remote CPU or I/O is not fabricated when a source does not expose it.
25. Financial, energy, carbon, business-value, and human-decision costs are excluded unless an explicit measured conversion is supplied.
26. Learned or adaptive models do not remove the need for conventional baselines, holdouts, uncertainty, or fallback.
27. A small Lean model would not verify the Rust runtime or calibrate the cost model.
28. A local or synthetic result would not establish production behavior.

29. A successful 10^8-row run would not establish billion-row scale.
30. No scale claim may rely on extrapolation or logical row counts.
31. A historic-system reconstruction cannot inherit the published system's performance.
32. A null semantic-feature result must be reported rather than hidden.

17 Open questions

1. What is the smallest P2 and P5 language that supports an independent full result-envelope oracle?
2. Which semantic dependencies belong in plan identity and which can safely expire without changing admissibility?
3. How should conflicting latency, source-load, freshness, and storage objectives be expressed without hidden weights?
4. When should uncertainty trigger a probe, fallback, or abstention?
5. Which remote metrics can be measured safely, and which must remain unknown?
6. How should censored timeouts enter calibration and ranking?
7. What workload horizon makes adaptive materialization identifiable rather than hindsight-driven?
8. How should semantic-definition changes invalidate learned features?
9. Can one decision certificate replay across connector upgrades, or must every capability revision create a new candidate identity?
10. Which provenance representations expose enough structure for cost prediction without requiring provenance computation first?
11. How should filtered vector quality be calibrated under drift and update lag?
12. How much candidate search is justified before planning dominates the query?
13. Which baselines can be reconstructed faithfully without inheriting unsupported historical performance?
14. What practical-effect threshold justifies semantic-statistics overhead?
15. Can an independent team reproduce one complete decision from raw inputs and certificates?

18 Conclusion

The proposed planner has a modest job description. It receives a semantic request, rejects physical strategies that cannot satisfy it, predicts the observable outcomes of the remaining

finite candidates, and either chooses with a replayable certificate or says why it cannot. That design keeps a fast but stale cache, a damaging remote pushdown, and a low-recall vector plan out of the same comparison class as an admissible result.

The research question is whether semantic metadata improves those constrained decisions after its own costs are counted. Established optimizers already provide search, statistics, capabilities, distributed costs, adaptive processing, caching, materialized views, feedback, and vector access paths. CatDB must earn any narrower claim with physical-only and capability-aware baselines, held-out calibration, adversarial workloads, and immutable decision traces.

No such result exists today. The current status is OBSTRUCTED. The next evidence gate is a hand-solvable finite profile with six to ten candidates, typed outcome vectors, an independent oracle, and deterministic choice, probe, fallback, infeasibility, and abstention behavior. Until that gate passes, the optimizer, benchmark, global-optimality, and billion-row claims remain absent.

A Claim anchors

Anchor	Future claim shape	Present status
P14-C01	feasibility filtering prevents declared semantic and policy violations	OBSTRUCTED
P14-C02	predictions retain units and explicit exclusions	OPEN
P14-C03	outcome intervals calibrate on held-out workloads	OPEN
P14-C04	ranking lowers constrained regret against baselines	OPEN
P14-C05	semantic features add net value after overhead	UNRESOLVED
P14-C06	cache reuse preserves complete compatibility	OBSTRUCTED
P14-C07	materialization crossover is predicted on held-out horizons	OBSTRUCTED
P14-C08	source-load constraints prevent harmful placement	OPEN
P14-C09	freshness and consistency profiles are never silently weakened	OBSTRUCTED
P14-C10	provenance-aware planning compares equal profiles	OBSTRUCTED
P14-C11	adaptation recovers without result-envelope change	OPEN
P14-C12	vector choices satisfy declared quality and freshness	OPEN

Anchor	Future claim shape	Present status
P14-C13	abstention reduces harmful confident decisions	OPEN
P14-C14	shared work improves total cost without unfair delay	OPEN
P14-C15	bounded physical-scale claims match executed inputs	UNRESOLVED

B Decision-certificate sketch

```
DecisionCertificate {
  request_digest,
  semantic_profile_digest,
  environment_digest,
  candidate_set_digest,
  feasible_candidates[],
  rejected_candidates[{ id, reason, evidence }],
  finalists[{
    id,
    prediction_vector,
    intervals,
    statistic_refs[],
    capability_refs[],
    calibration_domain
  }],
  objective_profile,
  hard_constraints[],
  chosen_or_abstained,
  winning_rule,
  runner_up_delta,
  probe,
  fallback,
  out_of_distribution_signals[],
  cache_materialization_evidence[],
  source_load_budget,
  adaptation_checkpoints[],
  expiry_conditions[],
  model_revision
}
```

The sketch is a proposed artifact shape, not an implemented serialization. Exact field types, canonicalization, and digest rules belong to I0.

C Falsification checklist

The central hypothesis is rejected or narrowed if any retained test shows:

1. plans admitted as equivalent produce different normalized result envelopes;

2. movement, residency, freshness, consistency, provenance, disclosure, or approximation policy is violated;
3. unsupported pushdown is silently approximated;
4. cache reuse ignores a semantic dependency or source frontier;
5. virtual and materialized modes disagree at the same request and frontier;
6. incremental and full refresh disagree at the same request and frontier;
7. vector latency is ranked without its quality requirement;
8. incompatible units are combined through undocumented weights;
9. estimate provenance is absent or cannot be replayed;
10. held-out prediction intervals are materially miscalibrated;
11. selected plans repeatedly lose to the conservative baseline;
12. semantic metadata has no net benefit after overhead;
13. source work shifted by pushdown is omitted;
14. provenance is measured under a weaker profile;
15. freshness is reduced to cache age;
16. feedback leaks held-out outcomes;
17. out-of-domain inputs are reported as calibrated;
18. adaptation duplicates or omits results;
19. planning dominates execution without bounded search; or
20. the proposed CatDB delta is fully explained by physical statistics and connector capabilities.

D Reproduction artifact layout

```
runs/<run-id>/
  manifest.json
  environment.json
  source-frontiers.json
  requests/
  candidates/
  capabilities/
  statistics/
  models/
  decisions/
  observations/
  correctness/
```

```
provenance/
vector-ground-truth/
failures/
analysis/
report/
checksums.txt
```

Every report cell links to raw observations, candidate plans, decision certificates, and a stable request identity. The repository manuscript does not contain such a run.

References

[1] Maribel Acosta, Maria-Esther Vidal, Tomas Lampo, Julio Castillo, and Edna Ruckhaus. ANAPSID: An adaptive query processing engine for SPARQL endpoints. In *The Semantic Web, ISWC 2011*, pages 18–34, 2011. doi: 10.1007/978-3-642-25073-6_2. URL https://doi.org/10.1007/978-3-642-25073-6_2.

[2] Yanif Ahmad, Oliver Kennedy, Christoph Koch, and Milos Nikolic. DBToaster: Higher-order delta processing for dynamic, frequently fresh views. *Proceedings of the VLDB Endowment*, 5(10):968–979, 2012. doi: 10.14778/2336664.2336670. URL https://doi.org/10.14778/2336664.2336670.

[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. doi: 10.1145/335191.335420. URL https://doi.org/10.1145/335191.335420.

[4] Peter Bailis, Shivaram Venkataraman, Michael J. Franklin, Joseph M. Hellerstein, and Ion Stoica. Probabilistically bounded staleness for practical partial quorums. *Proceedings of the VLDB Endowment*, 5(8):776–787, 2012. doi: 10.14778/2212351.2212359. URL https://doi.org/10.14778/2212351.2212359.

[5] 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. doi: 10.1145/3183713.3190662. URL https://doi.org/10.1145/3183713.3190662.

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

[7] Aris Charalambidis, Antonis Troumpoukis, and Stasinos Konstantopoulos. SemaGrow: Optimizing federated SPARQL queries. In *Proceedings of the 11th International Conference on Semantic Systems*, 2015. doi: 10.1145/2814864.2814886. URL https://doi.org/10.1145/2814864.2814886.

[8] Shaul Dar, Michael J. Franklin, Björn Jónsson, Divesh Srivastava, and Michael Tan. Semantic data caching and replacement. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, pages 330–341, 1996. URL https://www.vldb.org/conf/1996/P330.PDF.

[9] Facebook AI Research. Faiss index selection and cost parameters. Official project documentation, 2026. URL https://github.com/facebookresearch/faiss/wiki/Faiss-indexes. Accessed 2026-07-23.

[10] Boris Glavic and Gustavo Alonso. PERM: Processing provenance and data on the same data model through query rewriting. In *Proceedings of the 25th IEEE International Conference on Data Engineering*, pages 174–185, 2009. doi: 10.1109/ICDE.2009.15. URL https://doi.org/10.1109/ICDE.2009.15.

[11] Jonathan Goldstein and Per-Åke Larson. Optimizing queries using materialized views: A practical, scalable solution. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 331–342, 2001. doi: 10.1145/376284.375706. URL https://doi.org/10.1145/376284.375706.

[12] Goetz Graefe. Volcano—an extensible and parallel query evaluation system. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):120–135, 1994. doi: 10.1109/69.273032. URL https://doi.org/10.1109/69.273032.

[13] Goetz Graefe. The cascades framework for query optimization. *IEEE Data Engineering Bulletin*, 18(3):19–29, 1995. URL https://www.sigmod.org/publications/dblp/db/journals/debu/Graefe95a.html.

[14] Ashish Gupta and Inderpal Singh Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Engineering Bulletin*, 18(2):3–18, 1995. URL https://www.vldb.org/dblp/db/journals/debu/GuptaM95.html.

[15] 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. URL https://www.vldb.org/conf/1997/P276.PDF.

[16] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33 (1):117–128, 2011. doi: 10.1109/TPAMI.2010.57. URL https://doi.org/10.1109/TPAMI.2010.57.

[17] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572. URL https://doi.org/10.1109/TBDATA.2019.2921572.

[18] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. Learned cardinalities: Estimating correlated joins with deep learning. In *9th Biennial Conference on Innovative Data*

Systems Research, 2019. URL https://www.vldb.org/cidrdb/2019/learned-cardinalities-estimating-correlated-joins-with-deep-learning.html.

[19] Yannis Kotidis and Nick Roussopoulos. DynaMat: A dynamic view management system for data warehouses. *ACM Transactions on Database Systems*, 26(4):400–434, 2001. doi: 10.1145/503099.503100. URL https://doi.org/10.1145/503099.503100.

[20] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. How good are query optimizers, really? *Proceedings of the VLDB Endowment*, 9(3):204–215, 2015. doi: 10.14778/2850583.2850594. URL https://www.vldb.org/pvldb/vol9/p204-leis.pdf.

[21] 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. URL https://www.vldb.org/conf/1986/P149.PDF.

[22] Yu A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020. doi: 10.1109/TPAMI.2018.2889473. URL https://doi.org/10.1109/TPAMI.2018.2889473.

[23] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. Bao: Making learned query optimization practical. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1275–1288, 2021. doi: 10.1145/3448016.3452838. URL https://doi.org/10.1145/3448016.3452838.

[24] Volker Markl, Vijayshankar Raman, David E. Simmen, Guy M. Lohman, Hamid Pirahesh, and Miso Cilimdzic. Robust query processing through progressive optimization. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 659–670, 2004. doi: 10.1145/1007568.1007642. URL https://doi.org/10.1145/1007568.1007642.

[25] Thanh Phan and Wen-Syan Li. Dynamic materialization of query views for data warehouse workloads. In *Proceedings of the 24th IEEE International Conference on Data Engineering*, 2008. doi: 10.1109/ICDE.2008.4497452. URL https://doi.org/10.1109/ICDE.2008.4497452.

[26] Fotis Psallidas et al. OneProvenance: Efficient extraction of dynamic coarse-grained provenance. *Proceedings of the VLDB Endowment*, 16(12):3662–3675, 2023. doi: 10.14778/3611540.3611555. URL https://doi.org/10.14778/3611540.3611555.

[27] Muhammad Abdul Qudus, Muhammad Saleem, Axel-Cyrille Ngonga Ngomo, and Young-Koo Lee. An empirical evaluation of cost-based federated SPARQL query processing engines. *Semantic Web*, 2021. doi: 10.3233/SW-200420. URL https://doi.org/10.3233/SW-200420.

[28] 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. doi: 10.1145/582095.582099. URL https://doi.org/10.1145/582095.582099.

[29] Timos K. Sellis. Multiple-query optimization. *ACM Transactions on Database Systems*, 13(1):23–52, 1988. doi: 10.1145/42201.42203. URL https://doi.org/10.1145/42201.42203.

[30] Michael Stillger, Guy M. Lohman, Volker Markl, and Mokhtar Kandil. LEO: DB2's learning optimizer. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 19–28, 2001. URL https://www.vldb.org/conf/2001/P019.pdf.

[31] Suhas Jayaram Subramanya, Fnu Devvrit, Rohan Kadekodi, Ravishankar Krishnaswamy, and Harsha Vardhan Simhadri. DiskANN: Fast accurate billion-point nearest neighbor search on a single node. In *Advances in Neural Information Processing Systems 32*, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/09853c7fb1d3f8ee67a61b6bf4a7f8e6-Abstract.html.

[32] Trino Software Foundation. Optimizer documentation: Statistics, cost-based optimizations, pushdown, and adaptive plan optimizations. Official Trino documentation, version 483, 2026. URL https://trino.io/docs/current/optimizer/cost-based-optimizations.html. Accessed 2026-07-23.

[33] Jianguo Wang, Xiaomeng Yi, Ruoyu Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yutian Zou, Jiachun Long, Yufei Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Rui Jiang, Yi Wei, and Charles Xie. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627, 2021. doi: 10.1145/3448016.3457550. URL https://doi.org/10.1145/3448016.3457550.

[34] Haifeng Yu and Amin Vahdat. Design and evaluation of a continuous consistency model for replicated services. In *Proceedings of the Fourth Symposium on Operating Systems Design and Implementation*, pages 305–318, 2000. URL https://www.usenix.org/conference/osdi-2000/design-and-evaluation-continuous-consistency-model-replicated-services.