donto: A Bitemporal, Paraconsistent, Evidence-First Knowledge Substrate for the Age of Generative Abundance

Thomas Davis
donto
thomasalwyndavis@gmail.com

***Abstract*—For six decades the scarce, human-bottlenecked step in knowledge-base construction was *generating* typed facts. Guided large language models (LLMs) have removed that scarcity: a frontier model now emits an essentially unbounded, multi-directional stream of evidence-anchored (subject, predicate, object) claims—inventing predicates as it goes—for on the order of 10^{-4} USD each. The governing problem inverts accordingly, from *"how do we generate enough?"* to *"where do we hold an unbounded, contradictory firehose without discarding most of it?"* Vector stores and classical knowledge graphs answer by collapsing the stream: they deduplicate, pick a winner, and invalidate on conflict. We present donto, a knowledge substrate that does the opposite. donto is *bitemporal* (every claim carries valid-time and transaction-time), *paraconsistent* (incompatible claims coexist as legal state, related by a typed argument graph, with identity treated as a reversible hypothesis rather than a destructive merge), and *evidence-first* (every claim is anchored to a source span). Its central design choice is to *emit free and untyped at write time and defer typing, alignment, identity resolution, and joining to query time*. A deferred join is only as good as its key, and a purely lexical key cannot connect synonymous self-minted predicates; we therefore contribute an *embedding fabric* and a *three-signal scheduled alignment engine* (lexical, semantic, and LLM adjudication) that supplies a non-brittle, non-destructive join key while preserving contradictions. We report an implementation on a single commodity host holding $39{,}570{,}770$ live statements over $989{,}550$ distinct predicates (all counts measured on 2026-06-03), and a measured silver-label study showing that semantic candidate generation recalls 99.7% of $9{,}541$ accepted alignments against 66.4% for a lexical-only key—recovering 33.6% of equivalences that the lexical key cannot see at all.**

***Index Terms*—knowledge graphs, large language models, bitemporal databases, paraconsistency, ontology matching, entity resolution, vector search, embeddings, data integration**

I. Introduction

A. Generation was the scarce step; it no longer is

Knowledge-base construction has, for most of its history, been governed by the cost of *generation*: the human or semi-automated effort required to mint a single typed fact. From hand-curated expert systems through to large collaborative graphs such as Wikidata [1], and even in machine-assisted information-extraction pipelines, the binding constraint was the production of high-quality, schema-conformant assertions. Schemas, ontologies, and integration tooling were designed around this assumption of scarcity: because each fact was expensive, it was worth substantial up-front investment to model it correctly, to align it to a controlled vocabulary, and to resolve the entities it referred to before admitting it to the store.

That assumption no longer holds. Petroni et al. showed that pretrained language models already behave as soft knowledge bases, returning relational facts under cloze-style probing [2]. The trajectory since has been one of falling cost and rising volume: recent systems construct very large knowledge bases directly from LLMs at a marginal cost per fact several orders of magnitude below human curation. GPTKB materializes a knowledge base on the order of 10^8 triples elicited from a language model [3], and AutoSchemaKG performs autonomous, schema-free graph construction in which the model invents its own predicates [4]. A guided frontier model can now emit an essentially unbounded, multi-directional space of properties and relations about any entity—and, crucially, *invent the predicate as it goes*—for roughly 10^{-4} USD per claim.

The consequence is an *inversion* of the field's central problem. The hard question is no longer how to generate enough typed knowledge; generation is now cheap and abundant. The hard question is *where to put* an unbounded, self-typed, frequently contradictory, evidence-bearing firehose without throwing most of it away. This is a substrate problem, not an extraction problem, and it is the problem this paper addresses.

B. Existing substrates collapse the stream

The dominant data substrates respond to abundance by *collapsing* it. Vector databases reduce content to nearest-neighbour retrieval over a single embedding per item; semantically distinct or mutually contradictory passages are flattened into proximity, and conflict has no first-class representation. Classical knowledge graphs, including RDF/SPARQL stores [5], [6], are built to enforce consistency: ingestion deduplicates, conflicting values trigger a *pick-a-winner* resolution, and updates *invalidate* prior assertions. Entity resolution and data-matching pipelines [7], [8] and ontology-matching systems [9] are similarly engineered to *merge*: their success metric is the collapse of many surface forms into one canonical entity or property.

Collapse is the right strategy under scarcity, where each fact is trustworthy and expensive and the goal is a clean, queryable consensus. It is the wrong strategy under abundance. When facts are cheap, multi-directional, and drawn from heteroge-

neous sources of varying reliability, the contradictions in the stream are not noise to be removed but *signal to be retained*: they encode disagreement between sources, change over time, and the very uncertainty that a downstream consumer must reason about. A substrate that deduplicates on write discards exactly the structure that makes an abundant stream valuable, and it does so irreversibly.

C. donto's answer: hold everything, defer the join

donto inverts the collapse strategy. Rather than reducing the stream to a consistent consensus at write time, it *holds* the stream and resolves structure at read time. Three commitments make this concrete.

Paraconsistent hold. Contradictory claims coexist as legal state. Where a classical store would invalidate one of two conflicting values, donto retains both and records their relationship in a typed *argument graph* of support, rebuttal, undercut, and qualification edges. This is the database analogue of paraconsistent logic, in which a contradiction does not trivialize the system [10], [11]: donto does not derive arbitrary conclusions from inconsistency; it localizes and preserves it. Identity is likewise treated as a *hypothesis*—a reversible, query-time clustering—rather than a destructive merge, so that an incorrect "same-as" decision is always recoverable.

Bitemporality and non-destructive history. Every statement is a quad extended with two time dimensions: *valid-time*, when the assertion is held to be true in the world, and *transaction-time*, when the system knew it [12], [13]. Retraction and supersession set an upper bound on transaction-time; they never delete. The store is therefore append-only and fully auditable: any past state of knowledge can be reconstructed, and no source is ever silently overwritten.

Evidence-first provenance. Every claim is anchored to the source span that licenses it, through a chain from fact to evidence link to span (surface text and character offsets) to document revision to content-addressed blob. A claim that is not so anchored must be explicitly marked as a hypothesis carrying bounded confidence. Provenance is thus a write-time invariant rather than an optional annotation.

The design principle that ties these together—and that is donto's native strength—is to **emit free and untyped at write time, and defer typing, alignment, identity resolution, and joining to query time**. Extraction is allowed to mint predicates freely; the substrate does not force them into a schema on arrival. In the live system this yields $989{,}550$ distinct predicates, of which $736{,}670$ (74.4%) occur exactly once. We argue this long tail is not a proliferation defect but the *signature of abundance*: it is what an unconstrained, multi-directional generator produces, and it is tamed not by forbidding it at write time but by aligning it at read time.

D. The deferred join is only as good as its key

Deferring the join relocates, but does not remove, the integration problem. At query time a user who asks for the predicate `killed` must be able to reach its synonyms—`murdered`, `slew`, `assassinated`—and the inverse-direction forms minted by the generator, or the deferred join silently under-returns. The key on which this join depends must therefore connect *semantically* related self-minted predicates, not merely lexically similar strings.

A purely lexical key cannot do this, and the failure is sharp rather than marginal. The trigram neighbours of `killed` are confined to forms that share a substring—`killedAt`, `killedBy`, `killedIn`, `killedOn`—while genuine synonyms such as `murdered`, `assassinated`, and `slew` score near zero trigram similarity, far below any usable threshold, yet are embedding-near to `killed` (an illustrative cosine ≈ 0.9). We quantify this gap, with measured trigram values and the corresponding silver-label recall study, once and canonically in Section IV (Table III). Lexical similarity, in short, sees morphology; it is blind to meaning, and meaning is precisely what a deferred join over self-minted predicates requires.

This observation motivates the paper's principal technical contribution. donto maintains an *embedding fabric*: a dense vector (bge-small-en-v1.5, 384-dimensional [14], served via fastembed/ONNX, indexed by pgvector [15] with HNSW [16] under cosine distance) for each join-relevant object—predicate, entity, statement, span, document, and context—refreshed by a single scheduled loop and consulted wherever a join, match, or rank occurs. On top of the fabric runs a *three-signal scheduled alignment engine* that combines lexical similarity (trigram and FTS, which catches morphological variants such as `rdfType` versus `rdf:type`), semantic similarity (embeddings, which catches synonyms and paraphrase), and LLM adjudication (which assigns the *type* of the relation—exact, inverse, sub-property, close, or not-equivalent—and resolves false friends and direction, being the only signal that reads usage). Candidate generation is the cheap recall stage (lexical *union* semantic); LLM adjudication is the precision stage applied to borderline pairs; governance flags and a transitive closure gate which alignments are eligible for query-time expansion; and at query time a hybrid of full-text and vector retrieval, fused by reciprocal rank fusion [17], executes the search.

A non-negotiable constraint preserves donto's identity throughout: the embedding fabric is used to *cluster and rank*, never to *collapse or merge*. Proximity is a hypothesis, alignment is reversible query-time expansion, and contradictions remain held. The fabric supplies a better key for the deferred join without reintroducing the destructive collapse that donto exists to avoid.

E. Contributions

This paper makes the following contributions.

- **A data model for abundance.** A formal, bitemporal, context-scoped, polarity-bearing, provenance-anchored quad model with a typed argument graph and identity-as-hypothesis, governed by explicit write-time invariants (every claim anchored to a source span or marked hypothesis-only; no destructive overwrite), that holds contradictory claims as legal state rather than collapsing them.

- **Emit-free, defer-to-query-time.** A write/read division of labour in which predicates are minted freely at write time and typing, alignment, identity resolution, and joining are deferred to query time, with an analysis of why the resulting long predicate tail (74.4% singletons over $989{,}550$ predicates) is the expected and desirable signature of generative abundance.
- **An embedding fabric and a three-signal scheduled alignment engine.** A maintained dense-vector layer over all join-relevant objects, and a scheduled lexical/semantic/LLM ensemble that supplies a non-brittle, non-destructive, reversible join key—together with a measured silver-label evaluation showing that semantic candidate generation recalls 99.7% of $9{,}541$ accepted alignments versus 66.4% for a lexical-only key, recovering 33.6% of equivalences the lexical key cannot see.
- **An implementation on commodity hardware.** A working system—a Postgres 16 substrate, a Rust HTTP service, a Temporal-backed durable extraction engine, and pgvector/fastembed for the fabric—deployed on a single 4-vCPU/16 GB host, holding $39{,}570{,}770$ live statements with $1{,}886{,}580$ evidence links over $326{,}021$ spans.
- **An empirical study.** A 105-query coverage study across 21 analytical lenses, and a demonstration of paraconsistent retention on real data, including an event that simultaneously holds conflicting casualty counts from different sources together with an explicit discrepancy-recording claim.

The remainder of the paper formalizes the data model and its invariants (Section II), describes the emit-free architecture and query-time resolution (Section III), presents the embedding fabric and the scheduled alignment engine (Section IV), details the implementation (Section V), reports the empirical study (Section VI), positions donto against related work (Section VII), and discusses limitations and concludes (Section VIII).

II. The donto Data Model

This section formalizes the donto store. We first give the unit of record—a bitemporal, context-scoped quad carrying provenance and status—then the evidence chain that anchors it, the polarity and maturity attributes that let mutually incompatible records coexist, and finally a small set of invariants that distinguish donto from conventional knowledge stores. Throughout, we contrast donto's *non-collapsing* discipline with the *collapse-based* behaviour of vector databases and classical knowledge graphs (KGs). All quantitative figures are measured against the live deployment (verified 2026-06-03) and are summarized in Table I.

A. Statements as bitemporal provenance-bearing quads

The atomic unit of donto is the *statement*: a claim that a subject stands in a named relation to an object, asserted within a context and stamped with both world time and system time.

Definition 1 (Statement): A statement is a tuple

$$s = \langle\, id,\ subj,\ pred,\ o,\ ctx,\ \pi,\ \mu,\ V,\ T \,\rangle$$

where id is a surrogate identifier; $subj$ and $pred$ are IRIs/strings; the object $o \in (\mathcal{I} \uplus \mathcal{L})$ is *either* an IRI ($o \in \mathcal{I}$) *or* a typed literal ($o \in \mathcal{L}$), exclusively; ctx is a context IRI; π is a polarity ($+$ assert, $-$ deny); μ is a maturity grade; and V, T are the valid-time and transaction-time intervals defined below.

In the implementation a statement is one row of `donto_statement`. The exclusive choice of object is enforced physically by a check constraint, `(object_iri IS NOT NULL) <> (object_lit IS NOT NULL)`, so that $\mathcal{I}$ and $\mathcal{L}$ are disjoint at the schema level; literals are stored as `jsonb` values carrying a datatype tag. Polarity and maturity are packed into a small integer `flags` column. Crucially, the predicate is an *open* string: extractors mint predicates freely rather than drawing from a fixed vocabulary, which is why the live store holds $989{,}550$ distinct predicates, of which $736{,}670$ (74.4%) occur exactly once. This long tail is intrinsic to the abundance regime and is reconciled at read time rather than write time (Table II).

B. Bitemporality

donto records two orthogonal time axes in the sense of Snodgrass and of Jensen and Snodgrass [12], [13].

Definition 2 (Valid time): The valid time $V \subseteq \mathbb{T}$ of a statement is the (possibly unbounded, possibly open) interval of world time during which the asserted fact is held to be true. V is application data: it is what the claim says about the world.

Definition 3 (Transaction time): The transaction time $T = [\,t_0, t_1)$ of a statement is the interval of *system* time during which the store knew the statement as current. A live statement has $t_1 = \infty$ (an open upper bound); the store ceases to regard a statement as current by setting t_1 to a finite value—never by deleting the row.

In the schema, T is a `tstzrange` and V a `daterange`. The two axes are independent: the same world-fact may be re-asserted, corrected, or contradicted at many system times, and a single system time exposes a complete consistent slice of world history. We write $\mathsf{live}(s) \equiv \big(\mathrm{upper}(T_s) = \infty\big)$ for the current view, and the as-of-τ view is $\{\, s : \tau \in T_s \,\}$. Of the statements written to date, only 281 have been superseded by a finite upper bound, indicating that correction is rare relative to accretion but is exercised by the mechanism rather than simulated.

C. The evidence chain

donto is *evidence-first*: a claim is meaningful only in relation to a retrievable source. The provenance of a statement is a path through four levels of granularity.

Definition 4 (Evidence chain): For a statement s, an evidence link associates s with a *span* $\sigma = \langle text, [a, b)\rangle$, a character-offset window $[a, b)$ together with its surface text, located within a *document revision* r, which is in turn the textual realization of a content-addressed *blob* β:

$$s \xrightarrow{\text{evidence\_link}} \sigma \xrightarrow{\subseteq} r \xrightarrow{\text{realizes}} \beta.$$

Blobs are addressed by the SHA-256 of their bytes, so identical sources deduplicate to one β while remaining citable from many spans.

This chain lets every fact be traced to a verifiable textual locus and back to the raw artifact, supporting the retrospective audit and re-ranking that motivate the substrate. The live store holds $43{,}607$ documents, $326{,}021$ spans, and $1{,}886{,}580$ evidence links. Because some statements carry several links, these links anchor a smaller *set* of distinct statements: $877{,}648$ live statements (2.22%) currently carry at least one evidence link, a coverage figure we report honestly as a current limitation rather than a design endpoint (Section VIII). Spans are many-to-one onto revisions and revisions many-to-one onto blobs, so the chain compresses storage while preserving citation granularity.

D. Polarity, maturity, and coexisting incompatibility

Two attributes govern how a statement participates in the store's contradiction discipline. *Polarity* π distinguishes assertion (s) from denial ($\bar{s}$, the same quad with $\pi = -$), so denial is a first-class recorded act rather than the absence of an assertion. *Maturity* μ grades a statement along a promotion ladder (e.g. hypothesis, candidate, asserted), letting consumers filter by epistemic standing without removing immature claims from the store.

The essential property is that incompatibility does not trigger deletion or overwrite. A claim s and its denial $\bar{s}$, or two claims with literally inconsistent objects, may both be live simultaneously; their relation is recorded explicitly, not resolved by erasure (Property 4). Reconciliation, if any, is deferred to query time and to the argument and identity layers described in Section IV; the data model itself takes no position on which of two conflicting claims is "correct."

E. Invariants

We state the load-bearing invariants as numbered properties. They are what make donto a *paraconsistent*, append-mostly substrate rather than a snapshot of a single consistent world.

Property 1 (I1: anchoring-or-hypothesis): Every statement s either carries at least one evidence link (Definition 4) or is explicitly marked hypothesis-only. An anchorless hypothesis must carry confidence below 0.95. Equivalently, no high-confidence claim may exist without a retrievable source span.

Property 2 (I3: no destructive overwrite): Current state is never mutated in place and rows are never deleted from `donto_statement`. A correction is expressed by bounding the transaction time of the superseded statement ($\mathrm{upper}(T) \leftarrow \tau$) and inserting the replacement with a fresh open T. Hence the full transaction-time history is recoverable: for any past τ, the as-of-τ view is reconstructible exactly.

Property 3 (Identity-as-hypothesis): Co-reference between subjects is never realized by merging or rewriting rows. A proposed sameness is recorded as a reversible identity edge or cluster membership with its own provenance and review status; subjects remain physically distinct, and the proposed identity is consulted as a non-destructive expansion at query time. No write-time operation collapses two entities into one.

Property 4 (Paraconsistent coexistence): For any pair of mutually inconsistent live statements (including a quad and its negative-polarity denial), both may be simultaneously live. Inconsistency is recorded by typed argument edges (support, rebut, undercut, qualify) over statements rather than by removal of either party. The store therefore tolerates local contradiction without trivializing: a contradiction does not license arbitrary inference, in the spirit of paraconsistent logics [10], [11].

Property 2 is the database-level realization of the "never lose data" discipline and is implemented through transaction-time bounding rather than deletion. Properties 3 and 4 are what we mean by saying donto *does not collapse*: where conventional systems force a single surviving value, donto retains all parties and records the relations among them. The argument and identity machinery exists but is currently lightly exercised—$2{,}433$ argument edges and 122 identity edges over 39.5M statements—a gap we attribute in Section IV to the difficulty of *detecting* contradictions across freely minted predicates and unresolved entities using a purely lexical key.

F. Contrast with collapse-based stores

Properties 2–4 place donto in deliberate opposition to the dominant storage idioms. A vector database resolves an ingested item against existing items by nearest-neighbour and typically *deduplicates or upserts*, discarding the loser and retaining no record that a conflict occurred. A classical KG enforces a closed or curated vocabulary and a single asserted value per functional property, so a contradicting assertion either fails validation or *overwrites* its predecessor; RDF and SPARQL provide no native account of why a triple was replaced, and Wikidata mitigates this only by social process and qualifier conventions [5], [6], [1]. KG embedding methods such as TransE and the broader literature [18], [19] likewise assume a fixed relation set and a single intended graph, which they compress into vectors—an operation that is by construction lossy with respect to provenance and contradiction. Table II states the contrast along the dimensions that matter for the abundance regime. The key inversion is that donto moves typing, alignment, identity resolution, and joining from write time (where they force premature collapse) to read time, so that the store can accept the unbounded, contradictory, evidence-anchored stream without discarding most of it.

III. Emit-Free Architecture and Query-Time Resolution

The central architectural commitment of donto is a strict separation between the *write path* and the *read path*. The write path is deliberately impoverished in semantics: it captures claims as faithfully and as cheaply as possible, imposing no schema, no vocabulary control, no deduplication, and no identity resolution. All of those operations—typing, alignment, entity resolution, and the joins that depend on them—are

TABLE I
LIVE CORPUS MEASUREMENTS (DONTO-PG, VERIFIED 2026-06-03).

Quantity	Value
Live statements (live)	39,570,770
Superseded statements (finite T upper)	281
Distinct live predicates	989,550
of which singletons	736,670 (74.4%)
Distinct live subjects (entities)	$\sim 4.1 \times 10^6$ †
Contexts	23,411
Documents	43,607
Evidence spans	326,021
Evidence links	1,886,580
statements with ≥ 1 link	877,648 (2.22%)
Argument edges	2,433
Identity edges	122
`donto_statement` on-disk size	34 GB

† The distinct-subject count is the one headline figure not re-verifiable in interactive time: an exact `COUNT(DISTINCT subject)` over the live slice exceeds a 180 s query budget and `pg_stats` sampling is unreliable for high-cardinality columns. We report the most recent completed exact count (4,105,944) as an approximation and flag it explicitly.

deferred to the read path, where they are performed non-destructively and reversibly at query time. We call this the *emit-free* discipline. This section makes the discipline precise: it states the write-time invariants, defines the claim lifecycle, argues from the live corpus why deferral is *necessary* (rather than merely convenient) under generative abundance, and formalizes query-time predicate alignment, its transitive closure, and the governance that bounds non-destructive expansion.

A. The Write Path: Maximal Faithful Capture

Under classical knowledge-base construction, the write path is the expensive, human-bottlenecked step, and the system invests heavily in constraining it: an ontology fixes the admissible predicates, a mapping rectifies surface forms to canonical terms, and a deduplication or fusion stage resolves conflicts before persistence. Each of these is a *collapsing* operation—it discards information at write time on the assumption that the discarded material is noise.

donto inverts this assumption. When a guided language model extracts claims from a source, it emits an essentially unbounded, multi-directional stream of (subject, predicate, object) triples and *invents the predicates as it goes*: `killedBy`, `countDiscrepancyWith`, `servedUnder`, `phoneticVariantOf`. The write path's sole obligation is to persist these claims with enough provenance to be re-examined later, and to refuse nothing on semantic grounds. Concretely, a write is governed by two invariants (Definition 1 and Properties 1 and 2): no statement is ever removed—retraction and supersession close the upper bound of the transaction time rather than issuing `DELETE`—and every statement is either anchored to a source span or explicitly marked hypothesis-only with confidence below 0.95. The evidence chain (Definition 4) is what licenses the emit-free posture: because each claim is tethered to a retrievable source span, the write path may admit contradictory and redundant claims indiscriminately without losing the ability, later, to adjudicate them on the merits. Faithfulness to the source—not agreement with prior state—is the only write-time acceptance criterion.

B. The Claim Lifecycle

A claim's progress through donto is monotone with respect to information: stages add structure and never destroy the raw claim.

1) **Extract.** A model reads a source revision and emits free, untyped claims with self-minted predicates, each carrying a candidate evidence span.
2) **Anchor.** Spans are persisted against a content-addressed blob and document revision; evidence links bind claims to spans (Prop. 1).
3) **Persist.** Claims are written as live statements (Prop. 2), context-scoped, with transaction time opened at insertion. Nothing is deduplicated or canonicalized.
4) **Relate.** Where claims bear on one another, typed *argument* edges (support, rebut, undercut, qualify) and *identity hypotheses* (over subjects) are recorded as first-class, reversible objects—never as merges.
5) **Align.** The scheduled alignment engine (Section IV) proposes predicate alignments and entity identity edges, governed by review status and expansion-safety flags.
6) **Resolve at query.** At read time, alignment closure and identity clusters expand a query non-destructively; bitemporal and evidential re-ranking orders the held claims by current best understanding.

Stages 1–4 are write-side and append-only; stages 5–6 are read-side and reversible. Crucially, no stage *collapses* the corpus: deduplication, winner-selection, and identity merge—the operations a vector store or a classical KG performs eagerly—are simply absent.

C. Why Deferral Is Necessary Under Abundance

Under generative abundance, deferral is more than an engineering convenience (amortized work, schema evolution): eager normalization is *infeasible* and, where feasible, *lossy in ways that defeat the substrate's purpose*. Three forces, all grounded in the live corpus (Table I), make this concrete. *(i) The predicate distribution forbids a fixed schema.* The 74.4% singleton tail (Sec. II) is the signature of a model naming relations at the granularity of each source. Write-time vocabulary control could only reject that tail—discarding the most specific, source-faithful relations—or coerce each hapax predicate into a canonical term, which is the alignment problem performed prematurely, with less context, and irreversibly; both options collapse the very distribution the substrate exists to hold. A predicate set that grows in all directions can be *organized* but never *fixed*, and organization is a read-time activity. *(ii) Identity must remain a hypothesis.* With $\sim 4.1 \times 10^6$ distinct subjects from heterogeneous, contradictory sources, eager entity resolution would commit to a single merge at write time, exactly when context is thinnest and the decision least reversible; donto instead records "s_1 *is*

TABLE II
DONTO VERSUS COLLAPSE-BASED STORES ALONG ABUNDANCE-RELEVANT DIMENSIONS.

Dimension	Vector DB	Classical KG (RDF/Wikidata)	donto
Vocabulary	none (opaque text)	fixed/curated	open, freely minted predicates
On conflict	dedup / upsert (drop loser)	validate-fail or overwrite	coexist; record argument edge
History of change	none	limited (revision logs)	full bitemporal (Def. 3, Prop. 2)
Provenance	metadata, optional	qualifiers / references	mandatory chain (Prop. 1, Def. 4)
Identity	merge by similarity	merge / `owl:sameAs`	reversible hypothesis (Prop. 3)
Typing & joining	none / at write	at write (schema)	deferred to query time
Inconsistency	not representable	trivializes / disallowed	paraconsistent (Prop. 4)

s_2" as a first-class hypothesis (stage 4) that can be asserted, doubted, rebutted, and withdrawn without fusing rows. *(iii) Cost asymmetry favors the read side.* The 34 GB statement relation scans in ≈ 34 s; eager normalization would pay this, and the far larger cost of pairwise comparison across $\sim 10^6$ predicates and ~ 4.1M subjects, redundantly per write epoch, since each canonicalization is invalidated by the next schema insight. Deferral keeps the write path $O(1)$ per claim and pays only for the keys a query touches, each query benefiting from the most recent alignment state.

D. The Read Path: Query-Time Predicate Alignment

If predicates are minted freely, then a query for one predicate must, to be useful, also reach its synonyms, inverses, and specializations. donto achieves this with *query-time predicate alignment*: a non-destructive expansion of a queried predicate to a set of aligned predicates, governed so that expansion never silently changes what a query means.

Definition 5 (Predicate alignment): A *predicate alignment* is a directed, bitemporal, governed edge

$$a = \langle p_{\mathrm{src}},\, p_{\mathrm{tgt}},\, r,\, \mathit{conf},\, rs,\, \mathit{flags},\, V,\, T \rangle,$$

where r is the relation type drawn from

$$r \in \{\, \textsf{exact\_equivalent},\ \textsf{inverse\_equivalent},\ \textsf{sub\_property\_of},\ \textsf{close\_match},\ \textsf{decomposition},\ \textsf{not\_equivalent} \,\},$$

$\mathit{conf} \in [0,1]$ is a confidence, $rs \in \{$ candidate, accepted, rejected, superseded$\}$ is the review status, and $\mathit{flags} = \langle$safe_for_query_expansion, safe_for_export, safe_for_logical_inference$\rangle$ are independent expansion-safety gates. Like statements, alignments are bitemporal and append-only: an alignment is retired by closing T or by setting $rs =$ superseded, never by deletion.

Two features of Definition 5 are essential. First, the relation type is *semantic*, not merely a similarity score: exact_equivalent (`killed` $\equiv$ `slew`), inverse_equivalent (`killedBy` $=$ `killed`$^{-1}$), sub_property_of (`assassinated` $\sqsubseteq$ `killed`), and not_equivalent (an explicit negative, recording that two lexically close predicates are *not* the same) each license a different expansion behavior. Second, not_equivalent edges show that the alignment graph holds disagreements about meaning with the same paraconsistent discipline the statement store holds disagreements about fact.

a) Transitive closure.: Expansion is computed over the transitive closure of the equivalence-bearing alignment relations. Restricting to edges that are review-eligible ($rs \neq$ rejected) and safe_for_query_expansion, and respecting relation algebra (equivalence is symmetric and transitive; sub_property_of contributes only in the specialization-to-generalization direction; not_equivalent *blocks* a closure path rather than extending it), the closure $C(p)$ of a queried predicate p is the set of predicates reachable from p under these rules. A query for p is rewritten to range over $C(p)$. The closure is materialized incrementally so that read-time expansion is a lookup rather than a graph traversal.

Read-time expansion is then a three-step lookup: gather the live alignments that are safe_for_query_expansion and not rejected, take the materialized closure $C(p)$ (which respects not_equivalent blocks), and return the statements σ with $\sigma.p \in C(p)$, each *tagged* by the alignment relation and confidence through which it was reached. The operation is *non-destructive* in the strict sense: it adds and annotates rows but neither rewrites nor hides any stored statement. A consumer that distrusts close_match expansion filters on the tag; one performing logical inference restricts to safe_for_logical_inference alignments, a strictly more conservative set than the query-expansion set. The same construction applies on the subject side: entity identity edges and clusters expand a queried subject to its hypothesized co-referents under the same governance, so joining over entities is likewise a reversible read-time operation.

b) Governance bounds expansion.: Because expansion changes what a query *returns*, it is gated rather than automatic. Each alignment carries an independent review status and three orthogonal safety flags (Definition 5). The intended posture is conservative by default: a freshly proposed alignment is born $rs =$ candidate, and the review state then governs whether it may be marked safe_for_query_expansion (the least dangerous use—it merely broadens recall) while remaining *unsafe* for export or for logical inference until further reviewed. This factoring lets the substrate be liberal where the cost of error is a few extra retrievable rows and strict where the cost of error is an exported falsehood or an unsound deduction.

c) Live state of the read path.: The alignment layer is real but young. The live corpus holds 18,500 predicate

alignments and a materialized closure of 874,811 rows. A batch promotion pass has run since the engine was first scheduled: of the 18,500 alignments, 18,475 now carry rs = accepted and only 25 remain candidate, with the accepted set dominated by close_match (18,080) and a smaller core of exact_equivalent (292), inverse_equivalent (53), sub_property_of (26), not_equivalent (12), and decomposition (12). We note this honestly: the "born candidate" default still holds at insertion, but the population has already been promoted in bulk rather than left awaiting review, so the current store reflects a post-promotion, not a pristine-candidate, state. The crucial point, developed next, is that these alignments were generated with a *lexical* key as the dominant signal, and a lexical key cannot connect synonymous self-minted predicates—a limitation we quantify canonically in Section IV (Table III). The deferred join is only as good as its key; making that key non-brittle is the subject of the next section.

IV. The Embedding Fabric and Scheduled Alignment Engine

The design of donto defers typing, identity resolution, and joining from write time to query time (Sections II and III). This deferral is attractive precisely because it lets the substrate absorb an unbounded, self-typed firehose without committing to a schema; but it relocates rather than removes the integration burden. A query that asks for everything *killed* by an agent is only correct if, at read time, the system can recover the synonymous and inverse predicates that an extractor minted independently (`murdered`, `slew`, `assassinatedBy`, `victimOf`, ...) without ever having materialized a canonical form. The quality of every deferred join therefore reduces to the quality of a single artifact: the *key* used to decide that two freely-minted symbols denote the same relation or the same entity. This section presents that key—a maintained *embedding fabric*—and the *scheduled alignment engine* that consults it. We argue, and show empirically, that a purely lexical key is inadequate for self-minted vocabulary, that a dense semantic key supplies the recall a lexical key cannot, and that an ensemble which adds an LLM adjudication signal is required to assign the relation *type* and to preserve donto's paraconsistent invariants. Throughout, one constraint is non-negotiable: the fabric may *cluster* and *rank*, but it must never *collapse*. Proximity is a hypothesis, not a merge.

A. The join-key argument: a deferred join is only as good as its key

Let P be the set of distinct predicate symbols in the live store ($|P| = 989{,}550$, of which 74.4% are singletons; Sec. II). A query-time expansion mechanism must, given a queried predicate p, return a set $\exp(p) \subseteq P$ of predicates that are semantically compatible with p under a stated relation (equivalent, inverse, sub-property, close-match), so that the planner can rewrite a single-predicate selection into a disjunction over $\exp(p)$; the recall of the whole substrate is upper-bounded by the recall of exp. The naive key is lexical—trigram (`pg_trgm`) or `tsvector` similarity—which is cheap, index-friendly, and genuinely useful for one failure mode: morphological and punctuation variants of the *same* lemma (`rdfType`/`rdf:type`/`rdf_type`), which share most of their n-grams. It is structurally blind to the dominant failure mode of self-minted vocabulary: distinct lemmas that denote the same relation. Two predicates can be perfect synonyms and share *no* meaningful substring.

TABLE III

Lexical (trigram) vs. semantic (embedding) similarity for the predicate `killed`. *Trigram values are measured* with `pg_trgm similarity()` on the live substrate (the usable recall threshold for trigram candidate generation is $\tau_{\text{lex}} = 0.30$). Cosine values are *illustrative* similarities of `bge-small-en-v1.5` (384-dim) embeddings on these hand-chosen pairs, shown to convey the order-of-magnitude gap; the *measured*, population-level version of this contrast is the silver-label study in Table IV. The lexical key recovers only the morphological family and *none* of the true synonyms.

Source	Candidate	Trigram (measured)	Cosine (illustr.)
`killed`	`killedAt`	0.60	—
`killed`	`killedBy`	0.60	—
`killed`	`killedIn`	0.60	—
`killed`	`killedOn`	0.60	—
`killed`	`murdered`	0.0667	≈ 0.95
`killed`	`slew`	0.0000	≈ 0.88
`killed`	`assassinated`	0.0556	≈ 0.91
`killedBy`	`murderedBy`	0.1765	≈ 0.95

Table III makes the point quantitatively, and we state it *once, canonically, here*; later sections refer back to it rather than repeating the numbers. The trigram neighbourhood of `killed` above the usable threshold consists only of predicates that share the substring `killed`—`killedAt`, `killedBy`, `killedIn`, `killedOn`, each at trigram similarity 0.60—i.e. the lexical key recovers *inflections of the same stem and nothing else*. The true synonyms fall far below threshold: `murdered` scores 0.0667 (one incidental shared trigram), `assassinated` 0.0556, and `slew` exactly 0.0000. Even the closely paired `killedBy` and `murderedBy` score only 0.1765, below τ_{lex}. In embedding space the order is reversed: the illustrative cosines place every synonym near ≈ 0.9, an order of magnitude above their lexical scores. We are explicit that these particular cosines are illustrative, hand-chosen pairs rather than a benchmark; the *measured* statement of the same claim, over thousands of pairs, is the silver-label recall study below (Table IV). The conclusion is structural, not anecdotal: because extractors mint predicates from the surface phrasing of each source, synonymy in donto's vocabulary is overwhelmingly *lexically invisible*, and a lexical-only key silently caps the recall of every deferred join. Embeddings are the non-brittle join key the defer-joining vision requires. This is also why, before this work, the contradiction machinery was under-exercised: a surface-string key cannot bring two clashing claims into the same neighbourhood, so most contradictions were never surfaced for the argument graph to record.

TABLE IV
CANDIDATE-GENERATION RECALL AGAINST 9,541 ACCEPTED PREDICATE ALIGNMENTS USED AS SILVER LABELS (BOTH ENDPOINTS EMBEDDED; MEASURED 2026-06-03). THRESHOLDS: TRIGRAM ≥ 0.30, COSINE ≥ 0.60. "SEM. ONLY" IS THE FRACTION RECOVERED BY THE SEMANTIC SIGNAL BUT MISSED BY THE LEXICAL SIGNAL. THE LABELLED SET IS LEXICALLY BIASED (SEE TEXT), SO THE LEXICAL FIGURE IS AN UPPER BOUND.

Candidate generator	Recall
Lexical only (trigram ≥ 0.30)	66.4%
Semantic only (cosine ≥ 0.60)	99.7%
Union (lexical $\cup$ semantic)	100.0%
Semantic-only, lexical-missed	33.6%

a) A measured recall study on silver labels.: The `killed`/`murdered` row is an anecdote; to make the "lexical key is insufficient" claim a quantity, we treat the 18,475 accepted predicate alignments (Section VI) as silver labels and ask, for each, whether a cheap candidate-generation stage would have *recovered* the pair. Restricting to the 9,541 accepted pairs whose two endpoints are both present in the embedding fabric, we score each pair by its raw-IRI trigram similarity and by its embedding cosine, and apply the engine's operating thresholds ($\tau_{\text{lex}} = 0.30$, $\tau_{\text{sem}} = 0.60$). Table IV reports the result. A lexical-only candidate generator recovers 66.4% of the silver pairs; semantic candidate generation recovers 99.7%; and their union—the policy donto actually uses—recovers 100.0%. Decisively, 33.6% of the accepted alignments are recovered by the *semantic* signal *and missed entirely by the lexical signal*: a third of the equivalences would be invisible to a lexical-only deferred join. We flag a selection bias candidly: the silver labels were themselves produced by an engine in which the lexical signal was dominant, so the labelled set over-samples lexically-findable pairs and the measured lexical recall (66.4%) is, if anything, an *upper bound* on lexical recall over all true equivalences. Even under that bias, the semantic key strictly dominates and contributes a third of the recall outright, which is the conservative direction for the claim.

B. The fabric: one maintained vector per join-relevant object

The embedding fabric is a dense vector representation, maintained over time, attached to every object type at which a join, match, or rank occurs: predicates, entities (subjects), statements, evidence spans, documents, and contexts. Each vector is produced by `bge-small-en-v1.5` [14], a 384-dimensional sentence encoder, served locally via `fastembed`/ONNX so that embedding is a substrate operation rather than an external API dependency. Vectors are stored in-database using `pgvector` 0.8.2 [15] as a `vector(384)` column and indexed with HNSW [16] under `vector_cosine_ops`, co-located with the relational data so that semantic candidate generation is a single SQL query rather than a round-trip to a separate vector service. For a predicate or entity, the embedded text is a humanized projection of the symbol (de-camel-cased, depunctuated IRI segment plus any attached label); for a statement or span, it is the surface realization of the claim or its evidence text.

The fabric is *tiered* for cost reasons. The two highest-value join surfaces—predicates ($\sim 10^6$) and entities ($\sim 4.1 \times 10^6$)—are embedded fully and refreshed on every scheduled pass, because almost every query touches them. Statements (~ 39.6M), whose vectors would approach 60 GB and whose HNSW index would rival the substrate's 34 GB base-table footprint, are an opt-in tier embedded on demand for contexts under active analysis. At the measurement date the predicate tier held 262,896 embeddings and was backfilling toward full coverage of the live predicate set. Tiering is the fabric's concession to a single-node deployment (4 vCPU/16 GB): it keeps the always-on cost proportional to the join surfaces that dominate query traffic, while leaving the firehose-sized statement layer addressable but not eagerly materialized.

C. A three-signal ensemble

No single similarity signal is sufficient. Lexical similarity catches morphological variants but misses synonyms (Table III). Semantic similarity catches synonyms and paraphrase but cannot, on its own, distinguish *equivalence* from *opposition* or fix *direction*: `killed` and `killedBy` are near-neighbours in embedding space yet stand in an inverse, not an equivalence, relation, and antonymic false-friends (`ally` vs. `enemy`) can be embedding-adjacent. The alignment engine therefore fuses three signals, each contributing what the others cannot:

1) **Lexical (trigram).** Catches morphological and punctuation variants of a shared lemma (`rdfType`↔`rdf:type`). High precision, low recall on synonyms.
2) **Semantic (embedding).** Catches synonyms and paraphrase across disjoint surface forms (`killed`↔`murdered`). High recall, but agnostic to relation type and direction.
3) **LLM adjudication.** The only signal that reads predicate *usage* (subject/object types, exemplar statements). It assigns the relation *type*—`exact_equivalent`, `inverse_equivalent`, `sub_property_of`, `close_match`, `decomposition`, or `not_equivalent`—and resolves false-friends and direction. It is the adjudicator, not the recall mechanism.

The pipeline is organized as cheap recall followed by expensive precision. *Candidate generation* takes the union of the lexical neighbourhood ($\geq \tau_{\text{lex}}$, an exact `pg_trgm` index probe) and the semantic neighbourhood (the HNSW top-k above a cosine floor τ_{sem}). The union is deliberate: lexical recall and semantic recall are complementary, so their union dominates either alone—a claim we verify directly in Table IV, where the union recalls 100% of the silver-label alignments against 66.4% for lexical alone. *LLM adjudication* then runs only on the *borderline* pairs—those whose combined signal is ambiguous—to assign a relation type and a confidence; pairs adjudicated `not_equivalent` are recorded

as negative alignments so the engine does not re-propose them. Surviving alignments pass through *governance* and into *closure*, and are finally consulted by *query-time expansion*. The same fabric powers the query-side dual: hybrid retrieval that fuses a lexical `tsvector` ranking with a vector ranking by reciprocal rank fusion [17], $\mathrm{RRF}(d) = \sum_{r \in R} \frac{1}{k + \mathrm{rank}_r(d)}$, so that a search benefits from exact-term matches and semantic proximity simultaneously.

Each surviving alignment is a bitemporal, governed, evidence-anchored row in `donto_predicate_alignment`, carrying `relation`, `confidence`, a transaction-time validity interval, and—crucially— three independent governance flags: `safe_for_query_expansion` (may this alignment widen read-time results?), `safe_for_export`, and `safe_for_logical_inference`. The default posture is permissive for *retrieval* and conservative for *inference*: an alignment may be trusted to surface candidate rows long before it is trusted to license a logical deduction or an exported assertion. Alignments enter with `review_status` = `candidate` and are subsequently promoted. At the measurement date the engine held 18,500 predicate alignments, of which a batch promotion pass has moved 18,475 to `accepted` and left 25 at `candidate`; the accepted set is dominated by `close_match` (18,080), with smaller cores of `exact_equivalent` (292), `inverse_equivalent` (53), `sub_property_of` (26), `not_equivalent` (12), and `decomposition` (12). This is a marked change from the engine's prior state, which was dormant, unscheduled, and lexical-only; the embedding signal is what lifts recall past the surface-string ceiling.

D. Closure and query-time expansion

Pairwise alignments are not directly usable by the planner: equivalence is transitive and inverse-composition flips direction, so the engine materializes a *closure* from the alignment graph into `donto_predicate_closure`. Each closure row records, for a predicate, an `equivalent_iri`, the `relation` by which it is reached, a propagated `confidence`, and a `swap_direction` flag that records whether an odd number of inverse edges lies on the path—so that a query for `killed` expanded through `killedBy`→`murderedBy` rewrites the subject/object roles correctly. At read time, expansion is a governed lookup: given p, the planner selects closure rows for p whose relation is admitted by the query's mode and whose underlying alignments are `safe_for_query_expansion`, yielding $\exp(p)$ and the per-member direction flags, and rewrites the selection accordingly. Crucially the base statements are never rewritten; expansion is a non-destructive *view* computed at query time. At the measurement date the closure held 874,811 rows. The same construction applies on the entity side: identity edges and clusters define an analogous expansion of a queried subject to its co-referent subjects—identity as a reversible query-time hypothesis, not a write-time merge.

Algorithm 1 One iteration of the scheduled alignment cycle

Require: cursor c over recently-changed predicates/entities; thresholds $\tau_{\mathrm{lex}}, \tau_{\mathrm{sem}}$; top-$k$

$N \leftarrow \textsc{ChangedSince}(c)$ ▷ new/altered symbols
for $x \in N$ without a current vector **do**
  $v_x \leftarrow \textsc{Embed}(\textsc{Humanize}(x))$
  $\textsc{Upsert}(\text{fabric}, x, v_x)$; refresh HNSW
end for
for $x \in N$ **do**
  $L \leftarrow \{y : \mathrm{trgm}(x, y) \geq \tau_{\mathrm{lex}}\}$ ▷ lexical recall
  $S \leftarrow \textsc{HnswTopK}(v_x, k) \cap \{y : \cos(v_x, v_y) \geq \tau_{\mathrm{sem}}\}$ ▷ semantic recall
  $C \leftarrow (L \cup S) \setminus \textsc{KnownNegatives}(x)$
  for $y \in \textsc{Borderline}(C)$ **do**
    $(r, \kappa, a) \leftarrow \textsc{LLMAdjudicate}(x, y)$ ▷ relation, confidence, anchors
    if $r \neq$ `not_equivalent` **then**
      $\textsc{RegisterAlignment}(x, y, r, \kappa, \text{flags}, a)$
    else
      $\textsc{RecordNegative}(x, y)$
    end if
  end for
end for
$\textsc{RebuildClosure}()$ ▷ transitive + inverse-aware
$\textsc{RebuildIdentity}()$ ▷ entity edges/clusters
advance cursor c

E. The scheduled cycle

The fabric and the alignments it feeds are not built once; they are maintained by a single scheduled loop that processes newly-arrived and newly-changed objects incrementally on each pass, so that alignment quality tracks the firehose rather than lagging behind an occasional one-shot batch. The loop is *scheduled* rather than streaming—it runs as a recurring maintenance job, not on every write—and we are deliberate about the distinction: the instrumentation records each pass in a cycle log, of which the deployment holds 95 alignment runs spanning 2026-04-30 to 2026-06-03, with multiple completed passes on the measurement date itself, so the loop demonstrably *runs* rather than merely existing in principle. Algorithm 1 states one iteration.

The loop is idempotent and resumable: KnownNegatives prevents re-adjudication of rejected pairs, the cursor makes restart cheap, and closure and identity are rebuilt from the alignment graph rather than mutated in place, so a crash mid-cycle never corrupts the read-time view. Because adjudication runs only on borderline candidates, the dominant per-iteration cost is embedding and ANN probing, both sublinear in the store thanks to HNSW and the `pg_trgm` GIN index—a deliberate design response to the fact that a full sequential scan of the 34 GB statement table takes on the order of tens of seconds and cannot be on the critical path of a join.

F. Cluster, never collapse: preserving paraconsistency

The fabric is powerful enough to be dangerous: a system that can tell that `killed`, `murdered`, and `slew` are synonyms is one keystroke away from *merging* them, and a system that can tell two entity mentions are near-duplicates is one keystroke away from deduplicating them. Either move would silently destroy the property that distinguishes donto from a vector database or a conventional knowledge graph. We therefore adopt an inviolable invariant:

> **Invariant (Cluster, never collapse).** Embedding proximity may induce a *hypothesis* (an alignment edge, an identity edge) and may *rank* results; it may never delete, overwrite, or canonicalize a statement, a predicate, or an entity. Every consequence of proximity is a reversible, query-time expansion governed by `review_status` and the `safe_for_*` flags.

This invariant is what lets the fabric *strengthen* rather than dilute donto's paraconsistency. When the fabric draws two clashing claims into the same neighbourhood—an event simultaneously asserted to have 11 and "over 100" casualties from different sources, for example—it does not pick a winner; it makes the conflict *visible* to the argument graph, which records the rebut/qualify relation as a first-class edge while both claims remain legal state. The fabric is thus the missing sensor that lets the contradiction machinery finally fire: it supplies the recall (synonymous, inverse, paraphrastic predicates and co-referent entities) that the surface-string key could never reach, while the cluster-not-collapse invariant guarantees that surfacing a contradiction never resolves it by destruction. In donto, proximity is evidence for a hypothesis; alignment is a governed, reversible, query-time view; and contradictions, once visible, are held—not deleted.

V. Implementation

The system described in the preceding sections is realized as a running deployment, not a paper design. We report the implementation here in enough detail to make the engineering claims falsifiable, and we are deliberate about the realities — a 34 GB statement table, the cost of indexing 39.5 M vectors, HNSW recall/latency tuning — that the abundance-native model forces on a commodity host. All figures in this section were measured against the live instance.

A. Deployment topology

donto runs on a single commodity virtual machine: 4 vCPUs, 16 GB of RAM, with the substrate's data directory on a separate block volume. The choice of one modest box is intentional and load-bearing for the thesis of the paper: if holding an unbounded, contradictory, evidence-anchored firehose required a cluster, the economics of generative abundance would not close. The deployment comprises four cooperating components, all co-located:

1) **The substrate store** — a PostgreSQL 16 instance (containerized) holding the bitemporal, paraconsistent quad store, the evidence model, the argument graph, the alignment and identity layers, and the embedding fabric.
2) **`dontosrv`** — a Rust HTTP service that exposes write (claim ingestion), read (query-time expansion), and search endpoints over the store.
3) **The extraction engine** — an LLM-driven pipeline that turns source documents into evidence-anchored claims, made durable through Temporal so that no work is lost across restarts or provider failures.
4) **The scheduled alignment loop** — a recurring maintenance job that refreshes the embedding fabric and recomputes predicate/entity alignments and their transitive closure.

PostgreSQL was chosen over a purpose-built triple store or a dedicated vector database precisely because the model requires *bitemporality, paraconsistency, evidence, lexical similarity, and dense-vector similarity in one transactional context*. A federation of specialized engines would have to reconstruct cross-store consistency and would forfeit the single most useful property here: that an alignment decision, an evidence link, and a vector neighbourhood can be read in one consistent snapshot. We use `pg_trgm` for the lexical signal, `pgvector` (v0.8.2) for the semantic signal, `btree_gist` for the bitemporal range exclusion constraints, and `pgcrypto` for content-addressed blob hashing.

B. The bitemporal, paraconsistent schema

The central relation `donto_statement` realizes the formal quad of Section II directly in SQL. Its salient columns are:

Column	Type	Role
`statement_id`	`uuid`	claim identity
`subject`	`text`	freely-minted IRI
`predicate`	`text`	freely-minted IRI
`object_iri`	`text`	object (IRI branch)
`object_lit`	`jsonb`	object (literal branch)
`context`	`text`	scope / named graph
`tx_time`	`tstzrange`	transaction time
`valid_time`	`daterange`	valid (world) time
`flags`	`smallint`	polarity, maturity, hypothesis
`content_hash`	`bytea`	idempotent ingest key

Two design decisions are worth stating precisely. First, *object polymorphism* is encoded as a mutually exclusive pair `(object_iri, object_lit)`: exactly one is non-null, enforced by a `CHECK` constraint, so a literal object retains its typed JSON payload ({`"v": ..., "dt": ...`}) while an IRI object remains joinable. Second, *bitemporality is native to the column types*. Transaction time is a `tstzrange` and valid time a `daterange`; the "live" set at the current instant is exactly $\{s : \text{upper}(s.\texttt{tx\_time}) \text{ IS NULL}\}$, i.e. rows whose transaction interval is still open. This single predicate is the spine of the system: it appears verbatim in the live-row filter of every materialized view and partial index, and matching it exactly is what keeps a query off the full 34 GB heap.

Invariant I3 (no destructive overwrite) is realized procedurally rather than declaratively: retraction and supersession set the *upper bound* of `tx_time` on the affected rows and insert successors, but never issue `DELETE FROM`

donto_statement. History is therefore reconstructable for any past transaction instant. Paraconsistency is structural: nothing in the schema prevents two rows with opposing polarity or incompatible objects from being simultaneously live in the same context. As of measurement the store holds $\approx$ 39.56 M live statements across $\approx$ 23,411 contexts, over $\approx$ 4.1 M distinct subjects and 989,550 distinct predicates, of which the long singleton tail (74.4%) is the expected signature of free predicate minting rather than a defect to be normalized away.

The evidence model is realized by further relations — donto_evidence_link, donto_span, and the document/revision/blob chain — realizing fact → link → span → revision → blob. Blobs are content-addressed (SHA-256 via pgcrypto, then offloaded to object storage), giving automatic deduplication of identical sources. The current instance holds 326,021 evidence spans across 43,607 documents and 1,886,580 evidence links; those links number $\approx$ 4.77% of the live-statement count, but because some statements carry several links they anchor a smaller *set* of 877,648 distinct live statements (2.22% coverage), which we report honestly as a low-anchoring fraction (Section VIII) driven by historical bulk imports that predate strict enforcement of invariant I1. The argument graph (support/rebut/undercut/qualify) and the identity layer (identity edges, proposals, clusters) are separate tables; both are under-exercised relative to the volume of claims — on the order of dozens to thousands of edges — precisely because, before the embedding fabric, the surface-string key could not surface most of the contradictions and co-references that those layers exist to record.

C. The embedding fabric

The fabric (motivated in Section IV) supplies the non-brittle join key in SQL. Embeddings are computed with BAAI/bge-small-en-v1.5 [14], a 384-dimensional sentence encoder served locally through fastembed over ONNX Runtime—CPU-only, no GPU, no per-call API cost, a prerequisite for embedding a corpus of this size on a single box. Vectors are stored in pgvector [15] columns (e.g. donto_predicate_embedding(iri, embedding vector(384), model, updated_at)) and indexed for approximate nearest-neighbour search with HNSW [16] under the cosine operator class:

```
CREATE INDEX donto_predicate_embedding_hnsw
  ON donto_predicate_embedding
  USING hnsw (embedding vector_cosine_ops);
```

The model column tags every vector with its producing encoder, so the fabric can be re-embedded incrementally when the encoder changes without a global stop-the-world rebuild.

a) Cost-tiering and HNSW tuning.: The join-relevant classes differ by orders of magnitude—989,550 predicates, $\approx 4.1 \times 10^6$ entities, $\approx 3.96 \times 10^7$ statements—so embedding is tiered (quantified in Section VI-D, Table VIII): predicates and entities fully, statements opt-in, because a full statement-vector HNSW index ($\approx$ 60 GB before overhead) would approach the host's storage budget. This is the system's single most important engineering concession. At measurement the predicate backfill is in progress (262,896 vectors materialized, converging toward full coverage). HNSW's recall/latency parameters (ef_construction, ef_search, M) are tuned per query class—a smaller ef_search for alignment-candidate generation (an LLM adjudicator recovers precision downstream), a larger one for interactive search. Critically, a vector neighbourhood is always a set of *candidate* alignments surfaced to the three-signal ensemble, never an automatic merge: the fabric clusters and ranks but never collapses.

D. Query-time aggregates and the 34 GB problem

The donto_statement relation is, at measurement, 34 GB total (heap plus indices and TOAST). A full sequential scan therefore costs tens of seconds, which is acceptable for analytics but fatal on an interactive path. Two mechanisms keep reads off the heap. First, every hot aggregate is precomputed as a *materialized view* refreshed by the maintenance loop rather than recomputed per query: the deployment maintains, among others, per-context statistics, per-predicate and per-subject counts, a predicate-proliferation summary, and contradiction-pressure / paraconsistency-density rollups. Second, the live-row predicate upper(tx_time) IS NULL is pushed into *partial* indices so that interactive queries touch only the open-transaction slice; the same predicate must appear, byte-for-byte, in any matching FTS expression or the planner falls back to a full scan. Hybrid search composes a bounded lexical-candidate CTE with the vector path and fuses the two rankings by reciprocal rank fusion [17], with a server-side statement timeout that degrades to a clearly-flagged partial result rather than blocking. These are unglamorous decisions, but they are what lets a 40-million-claim, contradiction-preserving store answer interactive queries in sub-second time on four cores (an order-of-magnitude operating characteristic on the single co-resident deployment, not a controlled benchmark).

E. The extraction engine and Temporal durability

Claims enter through an LLM extraction engine that reads source documents and emits free, multi-directional, evidence-anchored (subject, predicate, object) claims, inventing predicates as needed. Because each extraction is a multi-step, externally-dependent, and occasionally-failing process (model calls, document fetches, span alignment, idempotent ingest), the engine is made durable with Temporal: each document's extraction is a workflow whose activities are retried with back-off and whose state survives worker restarts, so a crash mid-firehose loses no committed work and re-runs no completed work. Ingestion is idempotent at the store: content_hash keys deduplicate re-submitted claims, so safe retries cannot inflate counts. The model/provider is a swappable abstraction behind the engine, decoupling the substrate from any single LLM vendor.

F. The scheduled alignment loop

Query-time alignment depends on alignments kept fresh as the predicate and entity populations grow. This is realized as the scheduled loop of Section IV (Algorithm 1)—previously dormant, unscheduled, and lexical-only—which on each pass runs cheap lexical∪semantic recall, LLM adjudication on borderline pairs, governance flagging, and closure recomputation, and also advances the embedding backfill and materialized aggregates, recording each pass in a cycle log. The deployment has logged 95 such runs between 2026-04-30 and 2026-06-03, producing 18,500 alignments (18,475 `accepted`, 25 `candidate`) and 874,811 closure rows; all expansion is read at query time and revocable by flipping a flag, never by mutating claims. The concessions this deployment makes—tiered statement embeddings, 2.22% evidence coverage from pre-I1 bulk imports, sparse identity/argument layers, and single-box throughput limits—are discussed in full in Section VIII.

VI. Evaluation

We evaluate donto along four axes that track the claims of the preceding sections: the *abundance* regime and its predicate distribution (Section VI-A); the inadequacy of a lexical join key and the recall the embedding fabric adds (Section VI-B); query coverage with simultaneous paraconsistent retention (Section VI-C); and the resource budget under the tiered embedding policy (Section VI-D).

All figures are measured against the live production instance: a single Postgres 16 substrate on one 4-vCPU / 16 GB virtual machine, with the Rust service (DONTOSRV) and the extraction engine co-resident. Unless stated otherwise, counts were obtained by direct aggregation over `donto_statement` and its companion tables on 2026-06-03. We are explicit throughout about what is *measured* versus what is *not yet rigorously benchmarked*; in particular we do *not* claim alignment precision/recall against a curated gold standard, which we discuss as future work in Section VI-E.

A. Scale and the abundance signature

Table V reports the size of the live store. The substrate holds $\approx 3.96 \times 10^7$ live statements (those with an open transaction-time upper bound) spanning 2.34×10^4 contexts. The property that matters for our thesis is not the row count alone but the shape of the predicate distribution. The corpus contains 989,550 *distinct* predicates—freely minted by the extraction engine rather than drawn from a fixed ontology—of which 736,670 (74.4%) occur exactly once. This long, heavy singleton tail is the empirical signature of generative abundance: predicates are invented as needed and the typical predicate is used a single time. A schema-first store would have rejected the firehose or forced each of these 989,550 surface strings onto a small controlled vocabulary at write time, discarding precisely the multi-directional detail that motivates the design. We note that the predicate population is *growing* as extraction continues; an earlier snapshot recorded 865,834 distinct predicates, so the absolute figures here are a point-in-time measurement of a live, accreting system rather than a fixed benchmark.

TABLE V
SCALE OF THE LIVE SUBSTRATE (2026-06-03).

Quantity	Value
Live statements (open tx_time)	39,570,770
Contexts	23,411
Documents	43,607
Evidence spans	326,021
Evidence links	1,886,580
statements with ≥ 1 link	877,648 (2.22%)
Distinct predicates	989,550
singleton predicates	736,670
singleton fraction	74.4%
Distinct subjects (entities)	$\sim 4.1 \times 10^6$ †
`donto_statement` total size	34 GB
Full sequential scan	$\sim$34 s

† Approximate: exact `COUNT(DISTINCT subject)` over the live slice exceeds an interactive query budget; we report the most recent completed exact count (4,105,944).

Two consequences of Table V drive the rest of the system design. First, evidence anchoring is sparse: while the 1,886,580 evidence links number 4.77% of the live-statement count, they anchor only 877,648 *distinct* live statements (2.22% coverage), because some statements carry several links. Although the write-time invariant requires every claim to be either source-anchored or explicitly hypothesis-only, only a minority of *live* statements currently carry a materialized `evidence_link` to a span. This is an honest limitation of the historical extraction pipeline rather than a property of the model, and raising this fraction is an active engineering target. Second, a full sequential scan of the 34 GB statement table takes on the order of 34 s. Interactive query latency therefore *cannot* depend on scanning the relation; it must be served from indices and from query-time-maintained aggregates, which is what motivates the embedding fabric and the materialized alignment closure evaluated below.

B. Lexical versus semantic alignment keys

The deferred join is only as good as the key that connects independently minted predicates. We established this contribution canonically in Section IV: the measured trigram values (Table III) show that the lexical key recovers only the morphological family of `killed` and *none* of its true synonyms, while the silver-label recall study (Table IV) quantifies the gap at the population level—semantic candidate generation recalls 99.7% of 9,541 accepted alignments against 66.4% for a lexical-only key, with a third (33.6%) of the equivalences recoverable only semantically. We do not repeat those numbers here; instead we report the two operating characteristics that the evaluation adds: index latency and the current state of the engine.

a) Index latency.: With the HNSW predicate index warm, k-nearest-neighbour predicate lookup completes in ap-

TABLE VI
ALIGNMENT, IDENTITY, AND EMBEDDING-FABRIC STATE (2026-06-03).

Component	Count
Predicate alignments	18,500
`accepted` / `candidate`	18,475 / 25
Predicate alignment closure rows	874,811
Scheduled alignment runs logged	95
Argument edges (support/rebut/...)	2,433
Identity edges	122
Predicate embeddings backfilled	262,896
of distinct predicates	989,550
Embedding dimension	384
HNSW warm k-NN latency*	$\sim$100 ms

* $n = 1$, single co-resident deployment under live load; an operating characteristic, not a controlled benchmark.

proximately 100 ms, which is the cost incurred once per queried predicate to generate alignment candidates. This is the relevant figure for query-time expansion: the candidate-recall step adds a fixed sub-second overhead rather than scaling with the 39.5 M-row statement table. We report this and all other latency figures as order-of-magnitude operating characteristics taken with $n = 1$ on the single co-resident deployment under live load, not as controlled benchmarks (Section VI-E).

b) State of the alignment engine.: Table VI summarizes the alignment and identity machinery. At the time of measurement the alignment pass had produced 18,500 predicate alignments and a transitive closure of 874,811 rows, and the predicate embedding backfill had reached 262,896 of the 989,550 distinct predicates and is ongoing. The identity layer remains lightly exercised—122 identity edges and 2,433 argument edges—which is consistent with our diagnosis that a surface-string key cannot *see* most contradictions, so the contradiction machinery was under-fed before the semantic key was introduced. We report these as the current operational state, not as a tuned result.

C. Query coverage and a paraconsistency demonstration

a) Heterogeneous query workload.: To probe whether the substrate is queryable across diverse access patterns—and not merely on the predicates its alignment engine has already touched—we ran an empirical study of 105 queries organized into 21 thematic *lenses* (e.g. events, places, kinship, casualties, dates), five queries per lens. Each query was counted as returning if it produced at least one real row from the live store. 103 of 105 queries (98.1%) returned rows. We characterize this as a *coverage* measurement, and we are candid that it carries limited discriminating power: a non-empty 39.5M-row store returns rows for almost any well-formed query, so this number establishes that the store is broadly reachable across lenses, not that the returned rows are correct or complete. The discriminating, quantitative evidence for the paper's central claim is instead the silver-label recall study of Section VI-B (Table IV); relevance and ranking quality remain partially qualitative (Section VI-E).

TABLE VII
A HELD CONTRADICTION. TWO SOURCE-ANCHORED CASUALTY CLAIMS ABOUT ONE EVENT COEXIST AS LIVE STATEMENTS, JOINED BY AN EXPLICIT DISCREPANCY CLAIM. SUBJECT/SOURCE IDENTIFIERS ARE ABBREVIATED.

Claim (subject e)	Object	Status
`casualtyCount`	11	live, anch. (src. A)
`casualtyCount`	"over 100"	live, anch. (src. B)
`countDiscrepancyWith`	(the two above)	live

TABLE VIII
TIERED EMBEDDING POLICY. "FULL" TIERS ARE EMBEDDED EAGERLY AND REFRESHED ON EVERY SCHEDULED PASS; THE STATEMENT TIER IS OPT-IN. VECTOR PAYLOAD ESTIMATES ASSUME 384×4 BYTES PER VECTOR AND EXCLUDE HNSW INDEX OVERHEAD.

Object	Cardinality	Tier	Raw vectors
Predicates	9.87×10^5	full / cont.	$\sim$1.5 GB
Entities	4.1×10^6	full / cont.	$\sim$6 GB
Statements	3.96×10^7	opt-in	$\sim$60 GB (if full)

b) Holding a contradiction as legal state.: The query study surfaced a concrete instance of donto's defining behaviour: a single event whose two source-anchored claims assert incompatible casualty figures—11 and "over 100"—are both retained as live statements under the same subject with distinct provenance, joined by an explicit `countDiscrepancyWith` claim (Table VII). A collapse-based store would have invalidated, replaced, or deduplicated the second; in donto both remain queryable, each anchored to its own source, and the discrepancy is itself a first-class fact—invariant I3 and paraconsistent semantics operating on real data rather than a constructed example.

This demonstrates that the mechanism *functions* on production data, but 122 identity and 2,433 argument edges (Table VI) show the contradiction layer remains sparse relative to the 39.5 M-statement corpus; quantifying how many *latent* contradictions the semantic key newly exposes is ongoing work.

D. Cost and the tiered embedding policy

The fabric's cost is dominated by which object types are embedded (a 384-dim `float4` vector is $\sim$ 1.5 KB before index overhead). Embedding the $\sim 10^6$ predicates and $\sim$4.1 M entities is well within budget, but embedding all $\sim$ 39.6 M statements would produce tens of gigabytes of vectors plus a comparably large HNSW index, approaching the single box's disk budget. donto therefore tiers (Table VIII): predicates and entities full, statements/spans/documents opt-in. This keeps the always-on fabric proportional to the $\sim$5 M distinct join keys rather than the $\sim$40 M statements—the difference between a few gigabytes and a figure that would dominate the disk.

E. Threats to validity and what is not yet benchmarked

We close with an explicit account of the limits of this evaluation. First, the 98.1% query result is a *coverage* metric

(rows returned), not an accuracy metric; we do not report query precision or recall against ground truth, and the genealogical domain that dominates the corpus is one in which there is, by design, no single ground truth. Second, our recall measurement (Table IV) uses *silver* labels—the engine's own accepted alignments—rather than an independent gold standard, and those labels are biased toward lexically-findable pairs (Section VI-B); the individual embedding cosines in Table III are likewise illustrative hand-chosen pairs, not a measurement. We therefore report candidate- generation *recall* but not alignment *precision* against a curated gold set of predicate equivalences, nor a calibrated false-merge rate. Building such a gold set, stratified by predicate frequency, and reporting precision/recall—including the LLM adjudication stage's contribution over lexical∪semantic recall—is the principal remaining item of future work. Third, all latency figures are from a single co-resident deployment under live load and are reported as order-of-magnitude operating characteristics rather than controlled benchmarks. Fourth, the system is live and accreting; every count here is a point-in-time measurement, and the predicate population in particular is monotonically growing. We report these caveats not to discount the results but because the substrate's value proposition— holding an unbounded, contradictory firehose without discarding it—is only credible if its evaluation is equally candid about what it has and has not yet established.

VII. Related Work

donto draws on, and departs from, six lines of prior work: LLM-based knowledge-graph (KG) construction, the RDF/SPARQL/Wikidata tradition, KG and text embeddings, approximate nearest-neighbour (ANN) vector search, schema/ontology matching and entity resolution, and the logical foundations of paraconsistency, bitemporality, and hybrid retrieval. The recurring contrast is one of *disposition*: prior systems reduce an incoming stream of facts to a canonical, consistent state at write time—by deduplication, schema conformance, winner-selection, or invalidation-on-conflict—whereas donto holds the stream intact and defers reduction to query time. We summarise the contrast in Table IX and develop it below.

A. LLM-based KG construction

A recent line of work treats a language model itself as a source of structured knowledge. Petroni *et al.* [2] showed that pretrained LMs answer cloze-style relational queries competitively with supervised extractors, framing the LM "as a knowledge base." At the construction end of the spectrum, GPTKB [3] materialises a very large general-purpose KB—on the order of 10^8 triples for fractions of a cent each—by recursively prompting an LLM, and AutoSchemaKG [4] performs autonomous, schema-free KG construction at the 10^8–10^9-node scale, inducing the schema from the extracted graph rather than imposing one. These efforts establish empirically the premise on which donto rests: typed-fact generation, historically the human-bottlenecked and therefore scarce step, is now effectively unbounded and cheap. They differ from donto in what happens *after* generation. GPTKB and AutoSchemaKG produce a graph that is subsequently canonicalised, entity-linked, and (in the AutoSchemaKG case) schema-induced; contradiction between independently extracted assertions is resolved or simply absent because each pass yields a single graph. donto instead treats the multi-directional, predicate-inventing LLM stream as the *steady-state input* to a store designed to hold its contradictions, anchor each assertion to its source span, and align the freely minted vocabulary only when queried. The $989{,}550$ distinct predicates in our live store—of which 74.4% are singletons—are the expected signature of this regime, not a defect to be normalised away before storage.

B. RDF, SPARQL, and Wikidata

The Semantic Web stack provides donto's closest data-model ancestors. RDF 1.1 [5] models knowledge as $(\mathit{subject}, \mathit{predicate}, \mathit{object})$ triples and SPARQL 1.1 [6] queries them; Wikidata [1] is the largest collaboratively edited instance, and its qualifier-and-reference machinery and statement *ranks* are a partial, manually curated analogue of provenance and re-ranking. Two assumptions of this tradition are precisely what donto relaxes. First, RDF is monotonic and consistency-oriented at the schema level: a triple is asserted as fact, and conflicting triples about a functional property are treated as an error to be repaired, not as legal coexisting state. Second, vocabulary alignment in the Semantic Web is a *write-time, ontology-author* activity—one mints or reuses a stable IRI and declares `owl:sameAs` / `rdfs:subPropertyOf` relations up front. donto retains the quad shape but extends it to a bitemporal, context-scoped, polarity- and maturity-bearing tuple with mandatory source anchoring, permits contradictory quads to coexist, and pushes alignment from authoring time to query time. Where Wikidata curators rank competing values by hand, donto re-ranks by accumulated evidence and argument structure over time.

C. KG and text embeddings

Embedding methods supply donto's non-lexical join key. Translational and bilinear KG embeddings, beginning with TransE [18] and surveyed by Wang *et al.* [19], learn vector representations of entities and relations to support link prediction. Their objective, however, is to *complete* a fixed, comparatively small relational schema, which is ill-suited to a store whose relations are open-ended and largely singleton. donto therefore embeds the *surface form* of each join-relevant object—predicate, entity, statement, span, document, context—using a sentence-embedding model (BGE-small-en-v1.5 [14], in the lineage of Sentence-BERT [20]) so that synonymous self-minted predicates land near one another in vector space without any learned schema. This choice is what makes the deferred join tractable: a purely lexical key cannot connect `killed` to `murdered` (trigram similarity ≈ 0.07, below any usable threshold), whereas their embeddings are cosine-near (≈ 0.95). Embeddings here *cluster and rank*; unlike completion-oriented KG embeddings they are never used to assert or merge facts.

D. ANN and vector search

Serving embedding lookups at scale rests on approximate nearest-neighbour indexing. HNSW [16] provides the graph-based index donto uses (via `vector_cosine_ops`), FAISS [21] is the canonical high-throughput library, and pgvector [15] embeds ANN search directly in PostgreSQL, which lets donto co-locate vectors with the bitemporal store and fuse them with relational and full-text predicates in a single engine. Our contribution is not a new index but the *role* the index plays: rather than backing a standalone semantic-search product, the ANN layer is consulted wherever a join, match, or rank occurs, and is tiered—predicates and entities embedded fully and refreshed on a recurring schedule, the $\sim 4 \times 10^7$ statements left as an opt-in layer because a full HNSW index over them would approach the disk budget.

E. Ontology matching and entity resolution

Reconciling heterogeneous vocabularies and identifying coreferent entities are mature fields. Euzenat and Shvaiko [9] survey ontology matching; Christen [7] surveys data matching and entity resolution; and Dong and Srivastava [8] treat large-scale data integration including conflict resolution by truth discovery. donto's scheduled alignment engine is a recurring-batch instance of these techniques, but with two deliberate departures. First, matching is *non-destructive*: a predicate alignment or an identity edge is a reversible, governance-gated hypothesis (carrying a relation type among exact/inverse/sub-property/close/not-equivalent, a `review_status`, and a `safe_for_query_expansion` flag) and a transitive closure expands a queried key to its aligned set at read time, rather than rewriting or merging records. Second, donto refuses the truth-discovery move of selecting a single correct value per conflict that systems such as [8] adopt; competing values are retained and re-ranked. The engine combines three complementary signals—lexical (trigram, for morphological variants such as `rdfType`↔`rdf:type`), semantic (embeddings, for synonyms and paraphrase), and LLM adjudication (which reads usage to assign the relation type and to reject false friends and reversed direction)—in a recall-then-precision pipeline, a design that follows the matcher-combination literature but applies it as a recurring scheduled pass over an LLM-fed firehose.

F. Paraconsistency, bitemporality, and hybrid retrieval

donto's invariants have logical and database antecedents. Paraconsistent logics [10], defended philosophically by Priest [11], tolerate contradictions without explosion ($\{p, \neg p\} \not\vdash q$); donto operationalises this stance by admitting both polarities of a claim as legal state and recording their relation in an explicit argument graph (support/rebut/undercut/qualify) instead of letting a single inconsistency invalidate a context. Bitemporal data management [12], [13] supplies the valid-time×transaction-time structure under which retraction and supersession set time bounds rather than deleting rows, realising the no-destructive-overwrite invariant. For retrieval, donto fuses full-text and vector rankings with reciprocal rank fusion [17], a parameter-light combiner well suited to merging heterogeneous scores. Finally, the system's discovery ambition—surfacing latent relations across otherwise disconnected claims—descends from Swanson's literature-based discovery [22]; donto differs in that the connecting bridges are query-time embedding-mediated alignments over a contradiction-preserving store rather than curated term co-occurrences.

TABLE IX
DISPOSITION TOWARD THE INCOMING FACT STREAM. DONTO IS THE ONLY DESIGN THAT HOLDS CONTRADICTIONS AS LEGAL STATE AND DEFERS VOCABULARY/IDENTITY RECONCILIATION TO QUERY TIME.

Approach	Holds contradict.	Reconciles alignment	Provenance first-class
Classical KG / RDF [5]	no	write-time	partial
Wikidata [1]	ranked	manual	yes
LLM-built KGs [3], [4]	no	post-hoc	varies
Truth-discov. ER [8]	no	write-time	yes
Vector/hybrid [15], [17]	n/a	n/a	no
donto	**yes**	**query-time**	**yes**

Summary. The differentiator is crisp. Prior systems achieve consistency by *collapse*—deduplicating, conforming to a schema, choosing a winner, or invalidating on conflict—and they fix vocabulary and identity at write time. donto holds the unbounded, contradictory, evidence-anchored stream paraconsistently and bitemporally, and reconciles vocabulary and identity *non-destructively at query time* through an embedding fabric whose proximity is treated as a reversible hypothesis rather than a merge. No prior system combines abundance-native ingestion, paraconsistent retention, and an embedding-keyed deferred join in one substrate.

VIII. Limitations, Future Work, and Conclusion

donto is a working substrate carrying roughly 3.96×10^7 live statements, yet it remains a research prototype, and several of its central claims are at present better described as design commitments validated by construction than as empirically settled results. We discuss the principal limitations honestly, because they delimit exactly what the embedding fabric and the deferred-join model have and have not yet been shown to do.

A. Alignment quality: recall measured, precision not yet

The three-signal alignment engine (lexical, semantic, LLM adjudication) rests on two pieces of evidence established earlier: the measured trigram values showing that `killed` has *no* usable lexical neighbour among its true synonyms (Table III), and the silver-label study showing that semantic candidate generation recalls 99.7% of accepted alignments against 66.4% for a lexical-only key (Table IV). Together these demonstrate that a lexical key is *insufficient* and that the semantic key substantially closes the recall gap; they do *not* establish that the embedding-plus-LLM key is *sufficient* or *precise*. We have measured candidate-generation *recall*

against silver labels, but no independent gold set of alignment and identity judgements, so we report neither alignment *precision*/F_1 nor a calibrated false-merge rate for query-time expansion. A blind, adjudicated benchmark stratified by predicate frequency—so the 74.4% singleton tail is not under-represented—is the single most important piece of missing evidence and our immediate priority. The caveat applies with greater force to the identity layer ($\sim 10^2$ identity edges, $2{,}433$ argument edges): the contradiction machinery is genuinely under-exercised and its accuracy correspondingly unmeasured.

B. LLM adjudication: cost and reproducibility

The LLM is the only signal that reads usage—and hence the only one that can assign a relation *type* and resolve false-friends and direction (e.g. `killedBy` vs. `murderedBy`)—but it is also the slowest, most expensive, and only non-deterministic signal. Reserving it for borderline candidates bounds the cost, yet the candidate set grows with an unbounded vocabulary; we have not characterised adjudication throughput, marginal cost per resolved pair at full scale, or the closure's sensitivity to model, prompt, or temperature. Because alignments are governed, reversible state, an erroneous adjudication is recoverable rather than catastrophic, but reproducibility and drift across model versions remain real concerns for a component on the read path of every aligned query.

C. Engineering boundaries: scale, ANN recall, single box

Several limits are engineering boundaries rather than model boundaries. *(i) Statement-grain semantics is partial.* Predicates and entities are embedded fully, but the $\sim 3.96 \times 10^7$ statement vectors (≈ 60 GB before an HNSW index) would approach the disk budget on the single 4-vCPU/16 GB box, so they remain a tiered, opt-in layer; the deferred join is fully semantic for predicates and entities but only partially so for individual claims. *(ii) ANN recall is approximate and untuned.* HNSW can miss true neighbours, and since the neighbour set *is* the alignment candidate generator, any recall loss propagates into silent under-expansion—a failure mode harder to detect than over-expansion—which we have not yet measured against an exact-cosine baseline over `ef_construction`/`ef_search`/M. *(iii) One box, one domain.* The whole stack runs on a single machine: provenance and fabric stay co-resident with the statements, at the cost of throughput, horizontal scaling, and high availability (the ~ 34 s full scan is the symptom that motivates maintained aggregates). The corpus is also heavily skewed toward family-history evidence—a domain unusually rich in irreducible contradiction, which suits paraconsistency but limits external validity, since predicate distributions and contradiction density are domain-dependent and may not transfer to, e.g., biomedical or legal corpora.

D. Future work

Four directions follow directly. *(i) A hot-set statement/span semantic layer*—embedding statements in actively queried, contested, or argument-linked contexts rather than the whole 39.5M, with explicit coverage and staleness metrics. *(ii) Identity-cluster evaluation*—a gold set of co-reference judgements, cluster purity/completeness, and confidence calibration for identity-driven expansion, which is also the path to exercising the argument graph at load-bearing scale. *(iii) Retrospective time-slicing*—using bitemporality (invariant I3) to replay the claim stream as of time t and measure whether later evidence would have corrected the then-current ranking, turning "re-ranking by reality over time" into a quantity. *(iv) A discovery engine*—the aligned, paraconsistent, embedding-indexed store is a natural substrate for literature-based-discovery-style hypothesis generation [22], surfacing latent connections where embeddings place claims close but no argument edge yet links them; whether deferred-join expansion yields *verifiable* rather than merely plausible discoveries is the most interesting open question the substrate raises.

E. Conclusion

The economics of knowledge generation have inverted: the scarce step is no longer *producing* typed facts but *holding* the unbounded, contradictory, source-anchored firehose that guided LLMs now emit for $\sim 10^{-4}$ per claim. Vector databases and classical knowledge graphs answer by collapsing—deduplicating, picking a winner, invalidating on conflict—and so discard precisely the contested, evidence-bearing material abundance makes cheap. donto takes the opposite stance: a bitemporal, paraconsistent, evidence-first substrate of context-scoped quads with polarity, maturity, and dual valid/transaction time, anchoring every claim to a source span or an explicit hypothesis (I1), never destructively overwriting (I3), and holding incompatible claims as legal state related by a typed argument graph with identity as a revisable hypothesis. Its governing principle—emit free and untyped at write time, defer typing, alignment, identity, and joining to query time—is what lets it absorb $989{,}550$ freely minted predicates (74.4% singletons) as the signature of abundance rather than a defect.

The central technical claim is that the deferred join is only as good as its key, and that a purely lexical key is fatally brittle—made quantitative in Section IV, where a lexical-only candidate generator recovers 66.4% of the engine's accepted alignments against 99.7% for the semantic key, a third of them lexically invisible (Table IV). The scheduled embedding fabric and three-signal ensemble (lexical for morphology, semantic for synonymy, LLM for relation type and direction) supply the non-brittle key while *clustering and ranking but never collapsing*: proximity is a hypothesis, alignment a reversible query-time expansion, contradictions held. We are candid that alignment precision is unmeasured against gold standards, that LLM adjudication is the costly and least reproducible link, that statement-grain semantic search is disk-bounded on one box, and that our corpus is domain-skewed. None undercuts the core contribution: a concrete, deployed demonstration that an evidence-first, paraconsistent substrate with a scheduled embedding fabric can make deferred query-time joining work

over tens of millions of contradictory, self-typed claims without collapsing them into a single sanitised truth.

REFERENCES

[1] D. Vrandečić and M. Krötzsch, "Wikidata: A free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[2] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?" in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019, pp. 2463–2473.

[3] Y. Hu, S. Ghosh, T.-P. Nguyen, and S. Razniewski, "GPTKB: Comprehensively materializing factual LLM knowledge," *arXiv preprint arXiv:2411.04920*, 2024.

[4] J. Bai, W. Fan, Q. Hu, Q. Zong, C. Li, H. T. Tsang, H. Luo, Y. Yim, H. Huang, X. Zhou, F. Qin, T. Zheng, X. Peng, X. Yao, H. Yang, L. Wu, Y. Ji, G. Zhang, R. Chen, and Y. Song, "AutoSchemaKG: Autonomous knowledge graph construction through dynamic schema induction from web-scale corpora," *arXiv preprint arXiv:2505.23628*, 2025.

[5] World Wide Web Consortium, "RDF 1.1 concepts and abstract syntax," W3C Recommendation, 2014, https://www.w3.org/TR/rdf11-concepts/.

[6] ——, "SPARQL 1.1 query language," W3C Recommendation, 2013, https://www.w3.org/TR/sparql11-query/.

[7] P. Christen, *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer, 2012.

[8] X. L. Dong and D. Srivastava, *Big Data Integration*, ser. Synthesis Lectures on Data Management. Morgan & Claypool, 2015.

[9] J. Euzenat and P. Shvaiko, *Ontology Matching*, 2nd ed. Springer, 2013.

[10] N. C. A. da Costa, "On the theory of inconsistent formal systems," *Notre Dame Journal of Formal Logic*, vol. 15, no. 4, pp. 497–510, 1974.

[11] G. Priest, *In Contradiction: A Study of the Transconsistent*. Dordrecht: Martinus Nijhoff, 1987.

[12] R. T. Snodgrass, *Developing Time-Oriented Database Applications in SQL*. San Francisco, CA, USA: Morgan Kaufmann, 1999.

[13] C. S. Jensen and R. T. Snodgrass, "Temporal data management," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 36–44, 1999.

[14] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J.-Y. Nie, "C-Pack: Packed resources for general chinese embeddings," *arXiv preprint arXiv:2309.07597*, 2023.

[15] A. Kane, "pgvector: Open-source vector similarity search for PostgreSQL," https://github.com/pgvector/pgvector, 2024, software, version 0.8.2.

[16] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2020.

[17] G. V. Cormack, C. L. A. Clarke, and S. Büttcher, "Reciprocal rank fusion outperforms Condorcet and individual rank learning methods," in *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2009, pp. 758–759.

[18] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems 26 (NeurIPS)*, 2013, pp. 2787–2795.

[19] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[20] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019, pp. 3982–3992.

[21] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021.

[22] D. R. Swanson, "Fish oil, Raynaud's syndrome, and undiscovered public knowledge," *Perspectives in Biology and Medicine*, vol. 30, no. 1, pp. 7–18, 1986.