Journal of Advances in Information Technology, Vol. 17, No. 1, 2026

Toward an Embedded Semantic Reasoning Database: From Similarity Search to Semantic Discovery

Gerry Wolfe*, Ashraf Elnashar, and William Schreiber
Intificia, LLC, Bismarck, ND, USA
Email: gwolfe@intificia.com (G.W.); aelnashar@intificia.com (A.E.); wschreiber@intificia.com (W.S.)
*Corresponding author

***Abstract*—While Large Language Models (LLM) excel at semantic reasoning across concepts and domains, existing database systems, including those with vector capabilities and knowledge graphs, only support similarity search and graph traversal. They cannot perform inter-domain discovery, analogical reasoning, or causal chain discovery. This paper proposes a semantic reasoning database (ReasonDB), a novel database paradigm with a functional prototype that validates a couple of core capabilities. ReasonDB is the first system that treats embeddings as primary data and implements multiple reasoning modes as first-class operations. Through three core innovations: vector-native storage with probabilistic semantics, semantic reasoning primitives, and machine learning-driven adaptive indexing, ReasonDB transforms databases from passive storage systems into active discovery partners. Experimental validation demonstrates two breakthrough results: 7× performance improvement in inter-domain analogical reasoning over similarity search, and successful causal chain discovery where vector similarity fundamentally fails (0.708 causal chain strength despite 0.074 cosine similarity). This paper investigates four key areas. First, how vector-native storage transforms query capabilities to enable discovery-based reasoning that uncovers novel relationships across domains. Second, what new query classes and reasoning capabilities become possible. Third, how adaptive indexing improves performance over fixed strategies. Fourth, how these innovations integrate into a cohesive architecture. ReasonDB demonstrates that semantic reasoning requires fundamentally different database primitives. This enables entirely new classes of discovery operations impossible with current database architecture.**

***Keywords*—semantic reasoning databases, analogical reasoning, causal chain discovery, inter-domain discovery, vector-native storage, reasoning primitives, adaptive indexing, prob-abilistic semantics, database architecture**

I. Introduction

The Semantic Reasoning Database (ReasonDB) introduces a novel architecture designed for semantic reasoning, treating embeddings as primary data to enable inter-domain discovery. Large Language Models excel at cross-domain reasoning, yet current databases cannot natively support analogical capabilities at the data layer [1]. Unlike traditional relational databases, vector databases, or knowledge graphs, ReasonDB is embedding-native: it treats dense vector representations as primary data and implements semantic reasoning operations as first-class d atabase primitives [2, 3]. Current systems, while effective for similarity search and relationship traversal, cannot perform analogical reasoning due to their reliance on predefined semantic neighborhoods or explicit relationships [4, 5, 6].

Scientific breakthroughs often stem from analogical reasoning, recognizing structural patterns across domains [7]. For example, CRISPR-Cas9's repurposing from bacterial immunity to gene editing [8] and biomimetics' adaptation of gecko feet for adhesives [9] exemplify cross-domain analogical reasoning. Current databases cannot support such discovery at scale; implementing analogical reasoning as database primitives enables declarative, optimized queries over large-scale semantic data rather than requiring custom application-layer implementations. This architectural gap prevents discovering functional similarities across therapeutic domains or detecting fraud patterns adapted across financial markets, contributing to billions in losses [10, 11]. ReasonDB's database-layer reasoning primitives could help minimize these costs by making cross-domain discovery just a query away instead of requiring dedicated application services.

ReasonDB addresses these challenges through three innovations: vector-native storage, reasoning primitives, and adaptive indexing. These transform databases into active discovery partners, enabling queries impossible with current systems. This paper contrasts existing approaches with semantic reasoning requirements, presents ReasonDB's architectural innovations including comparisons with recent AI-enhanced database systems, provides mathematical foundations, validates the approach through prototype experimentation, and discusses implications for future database architectures.

This paper makes four key contributions: (1) experimental validation showing 7× performance improvement in inter-domain analogical reasoning over similarity search; (2) a novel vector-native architecture treating embeddings as primary data rather than auxiliary indices; (3) five semantic reasoning primitives (analogical reasoning, temporal evolution, domain alignment mapping, causal chain discovery, concept synthesis) implemented as first-class database operations; and (4) adaptive indexing using machine learning to optimize for diverse reasoning patterns. Together, these contributions demonstrate that semantic reasoning databases require fundamentally different architectural primitives than current systems.

The remainder of this paper is organized as follows: Section II contrasts current approaches with semantic reasoning requirements; Section III presents our research questions; Section IV details ReasonDB's architectural

Manuscript received September 6, 2025; revised October 4, 2025; accepted October 23, 2025; published January 15, 2026.

innovations including contrasts with AI-enhanced systems; Section V provides mathematical foundations; Section VI analyzes implementation feasibility; Section VII validates the approach through prototype experimentation, demonstrating 7× performance improvement; Section VIII positions our work within the broader research landscape; and Section IX concludes with future directions.

II. Current Databases: The Cross-Domain Causal and Analogical Reasoning Gap

Vector databases excel at similarity search but cannot distinguish between semantic similarity and structural analogy, typically returning results from the same conceptual domain [12]. Knowledge graphs enable relationship traversal yet remain constrained to pre-encoded connections [5]. Neither supports analogical reasoning across disparate domains, which is critical for intelligent discovery.

The semantic reasoning gap persists across current database architectures due to three fundamental constraints. First, reliance on discrete records, exact results, and fixed schemas prevents discovery-oriented queries [13, 14]. Second, vector systems measure correlation in embedding space, not causation through mechanistic pathways. Third, graph systems traverse explicit relationships, not implicit structural correspondences. No current system natively supports queries like ”what mechanisms from domain A solve problems in domain B?” or ”what causal pathways connect semantically dissimilar states?”

Code Block 1 shows a similarity search retrieving drugs similar to remdesivir (e.g., antivirals), confined to the same therapeutic domain due to a high similarity threshold.

```
query = "SELECT similar_drugs FROM drugs WHERE
    embedding_similarity(drug, 'remdesivir') >
    0.9"
similarity_search(query) -> [ivermectin, paxlovid
    , ...]
```

Code Block 1. Similarity search.

Code Block 2 demonstrates knowledge graph traversal, returning drugs explicitly linked to COVID-19, limited to predetermined relationships.

```
query = "MATCH (covid:Disease)-[:CAUSED_BY]->(
    virus:Virus)-[:INHIBITED_BY]->(drug:Drug)
    RETURN drug.name"
knowledge_graph_traversal(query) -> [remdesivir,
    paxlovid, molnupiravir, ...]
```

Code Block 2. Knowledge graph traversal.

In contrast, Code Block 3 illustrates ReasonDB’s semantic reasoning, transferring viral RNA interference mechanisms to materials science and agriculture, generating novel inter-domain applications like smart materials and antiviral crop coatings.

```
concept_a = "viral_RNA_interference"
domain_1 = "material_science"
domain_2 = "agriculture"
reasoning_type = "mechanism_transfer"
semantic_reasoning(concept_a, domain_1, domain_2,
     reasoning_type) -> ["
    smart_material_gene_silencing", "
    crop_antiviral_coatings", "
    agricultural_RNA_delivery_systems", ...]
```

Code Block 3. Semantic reasoning discovery.

These limitations motivate ReasonDB’s vector-native architecture with semantic reasoning primitives, designed to enable the discovery operations that current systems fundamentally cannot support.

III. Research Questions

The fundamental limitations identified in current database architectures drive ReasonDB’s architectural innovations. Rather than pursuing incremental improvements to existing approaches, ReasonDB addresses four critical requirements: (1) vector-native storage that eliminates dual-storage architecture by treating embeddings as primary data; (2) reasoning primitives that implement semantic reasoning as first-class database operations; (3) adaptive indexing that provides dynamic optimization for different semantic reasoning patterns; and (4) system integration that ensures the innovations work together as a cohesive, extensible, and production-ready architecture.

This research investigates how each innovation contributes to overcoming the core constraints that prevent current systems from supporting semantic reasoning and inter-domain discovery. The research questions are:

1) How does vector-native storage impact storage overhead, computational efficiency, and the semantic integrity of data compared to traditional systems that maintain separate vector and data stores?
2) What new classes of analytical queries and inter-domain reasoning tasks does ReasonDB enable, and how accurately can it track the temporal evolution of concepts?
3) To what extent can a machine learning-driven adaptive indexing strategy improve query performance and resource utilization over static indexing methods when handling dynamic, evolving query patterns?
4) How effectively do the proposed innovations in storage, reasoning, and indexing integrate into a cohesive, extensible, and production-ready system?

These questions guide our investigation from architectural design (RQ1, RQ3) through capability validation (RQ2) to system integration (RQ4). The prototype in Section VII directly addresses RQ2 by demonstrating analogical reasoning performance, while Sections IV and V establish the theoretical foundations for RQ1 and RQ3.

IV. Methodology and Architectural Innovations

ReasonDB’s architecture comprises four components: Semantic Manifold Storage Layer, Reasoning Primitive Engine, Adaptive Query Optimization Layer, and Temporal Consistency Management System—designed to enable semantic reasoning while ensuring performance and reliability (Fig. 1). Unlike traditional databases that separate storage, computation, and optimization, ReasonDB integrates these to handle the continuous, probabilistic nature of semantic data.

A. Vector-Native Storage Layer

The Semantic Manifold Storage Layer implements the vector-native storage principle, treating embeddings as primary data rather than auxiliary indices. This layer represents entities as probabilistic distributions (μ, Σ, τ, $\mathcal{C}$, Γ, Ψ) on semantic manifolds, eliminating dual-storage overhead and enabling direct reasoning operations (Fig. 2).

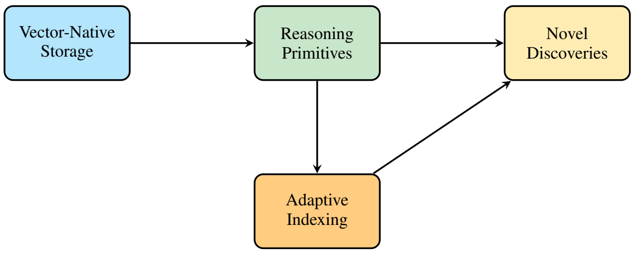

Fig. 1. Vector-native storage enables reasoning primitives, which inform adaptive indexing for optimized inter-domain discovery.

Core embeddings (μ) capture semantic meaning, uncertainty matrices (Σ) quantify ambiguity, temporal signatures (τ) track concept evolution, contextual factors ($\mathcal{C}$) adjust meaning by domain, causal connectivity tensors (Γ) encode mechanistic relationships, and synthesis potentials (Ψ) represent conceptual blending capacity. Traditional fields are computed on-demand, reducing storage from $O(n(k+d))$ to $O(nd+|\Gamma|+d)$.

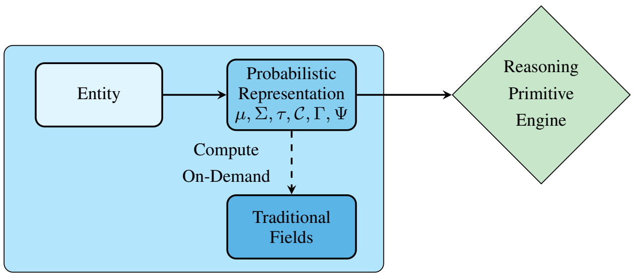
Entity
Probabilistic Representation $\mu, \Sigma, \tau, \mathcal{C}, \Gamma, \Psi$
Compute On-Demand
Traditional Fields
Reasoning Primitive Engine

Fig. 2. Entities as probabilistic distributions with traditional fields computed on-demand, enabling semantic reasoning.

B. Contrasting Architectural Philosophies: AI-Enhanced vs. AI-Native

Recent work has explored AI integration with fundamentally different architectural philosophies than ReasonDB. Understanding these contrasts clarifies ReasonDB's design choices and their implications for semantic reasoning capabilities.

Stanford's LOTUS introduces semantic operators into relational query processing, extending SQL with semantic filters, joins, and aggregations [15, 16]. AnDB bridges structured and unstructured data through LLM-powered query interfaces that map between data representations [17]. Both systems represent the AI-enhanced paradigm: traditional database architectures augmented with AI capabilities.

ReasonDB adopts a fundamentally different approach: AI-native architecture where reasoning is a first-class operation rather than an enhancement layer. The key distinction lies in the computational model. LOTUS and AnDB augment existing database operations with LLM calls for semantic retrieval; ReasonDB performs multi-step analogical reasoning natively through specialized primitives. LOTUS extends relational models while AnDB enhances SQL capabilities, both computing semantics when needed. ReasonDB inverts this model: embeddings are primary data, with traditional fields computed on demand.

This architectural inversion enables qualitatively different query capabilities. AI-enhanced systems excel at making traditional database operations semantically aware—filtering records by semantic similarity, joining tables through conceptual relationships, or translating natural language to SQL. ReasonDB enables discovery queries that bridge conceptual gaps between domains through structural pattern mapping rather than semantic similarity alone. The system can answer "what mechanisms from domain A solve problems in domain B?" through native analogical reasoning primitives, or "what causal pathways connect semantically dissimilar states?" through causal chain discovery—queries impossible in AI-enhanced architectures without extensive application-layer programming.

Table I summarizes these architectural distinctions.

TABLE I
ARCHITECTURAL PHILOSOPHY COMPARISON

Characteristic	**AI-Enhanced (LOTUS, AnDB)**	**AI-Native (ReasonDB)**
Primary Data Model	Relational records	Vector embeddings
AI Integration	Enhancement layer	First-class operations
Semantic Processing	On-demand LLM calls	Native primitives
Query Capability	Enhanced SQL	Discovery operations
Example Query	"Find similar records"	"Transfer mechanisms across domains"

The choice of AI-native architecture imposes different trade-offs. AI-enhanced systems maintain compatibility with existing database ecosystems and leverage mature optimization techniques from decades of relational database research. ReasonDB sacrifices this compatibility to enable reasoning operations that would require complex application-layer orchestration in traditional architectures. This trade-off proves worthwhile for discovery-oriented workloads where cross-domain reasoning and causal pathway exploration constitute core requirements rather than occasional enhancements.

In summary, LOTUS and AnDB make database operations AI-aware; ReasonDB makes AI reasoning operations database-native. This distinction shapes the entire system architecture and determines which classes of semantic queries can be expressed declaratively versus requiring procedural implementation.

C. Reasoning Primitive Engine

The reasoning primitive engine processes discovery queries through five operations: analogical reasoning (structural pattern mapping), temporal evolution (concept change tracking), causal chain discovery (cause-effect pathways), domain alignment mapping (shared semantic space projection), and concept synthesis (novel idea generation). These operate on vector-native storage, with an execution pipeline decomposing queries into optimized primitive sequences (Fig. 3).

Adaptive indexing uses contextual bandit algorithms, which balance exploration and exploitation to dynamically select index structures (e.g., hierarchical clustering, inter-

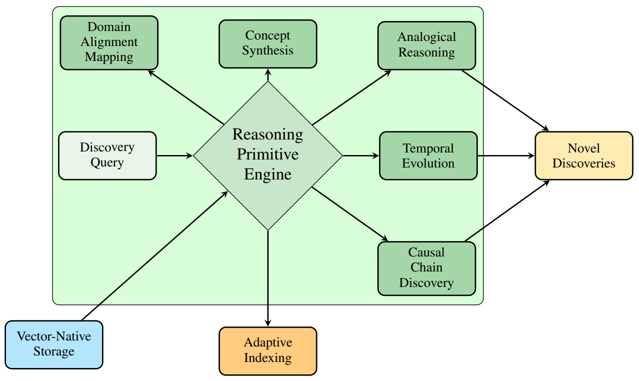

Fig. 3. Five operations process discovery queries on vector-native storage, coordinated for performance optimization.

domain bridges) based on query patterns, optimizing performance for diverse reasoning tasks (Fig. 4).

Temporal consistency management ensures semantic coherence in distributed deployments through probabilistic alignment, where replicas are consistent if $\int_{\mathcal{M}} |P_1(x) - P_2(x)|dx < \epsilon$.

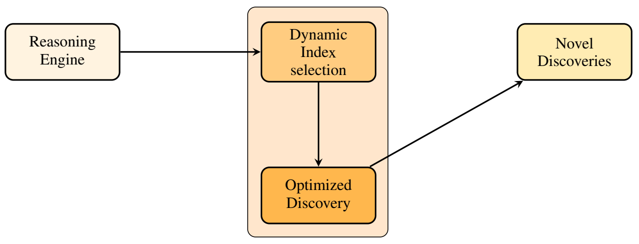

Fig. 4. Dynamic index selection optimizes semantic reasoning performance based on query characteristics.

V. Theoretical Framework

Having established why semantic reasoning requires a new database architecture, we now formalize how ReasonDB works. This section presents the mathematical foundations underlying the storage layer, reasoning primitives, and adaptive indexing mechanisms.

A. Semantic Manifold Storage Layer

As outlined in Section IV, ReasonDB represents entities through an extended probabilistic framework $e = (\mu_e, \Sigma_e, \tau_e, \mathcal{C}_e, \Gamma_e, \Psi_e)$ on a semantic manifold $\mathcal{M} \subseteq \mathbb{R}^d$, addressing Research Question 1.

1) Storage vs. Computation Architecture: The extended storage layer provides the foundational representations upon which reasoning primitives operate (Fig. 2). Each component $(\mu_e, \Sigma_e, \tau_e, \mathcal{C}_e, \Gamma_e, \Psi_e)$ stores entity properties as data; the reasoning primitives (analogical reasoning, temporal evolution, domain alignment mapping, causal chain discovery, concept synthesis) are computational operations that leverage these stored representations to perform discovery queries. This separation enables efficient query processing. Entities maintain rich semantic representations while primitives compute relationships dynamically.

2) Component Definitions: The core embedding $\mu_e \in \mathbb{R}^d$ captures semantic meaning. The uncertainty covariance matrix Σ_e quantifies semantic ambiguity. The temporal signature τ_e encodes evolution patterns. Contextual factors $\mathcal{C}_e$ modulate meaning based on domain. The causal connectivity tensor $\Gamma_e : \mathcal{E} \to [0,1] \times \mathcal{M}$ maps entities to causal relationships with strength scores and mechanism embeddings. The synthesis potential $\Psi_e \in \mathbb{R}^d$ represents conceptual blending capacity and constraints. This vector-native approach eliminates dual-storage overhead, reducing complexity from $O(n(k+d))$ to $O(nd + |\Gamma_e| + d)$ by computing traditional fields on demand, enabling direct reasoning operations across all five primitives.

The causal connectivity tensor Γ_e enables queries like ”what sequence of mechanisms transforms problem state X to solution state Y?” by storing not just that entities are causally related, but how they are related through specific mechanisms. For example, Γ_{pathogen} might map to (0.85, $m_{\text{immune\_response}}$), indicating strong causal influence through an immune response mechanism. This representation enables causal chain discovery to trace mechanistic pathways that would be invisible to similarity-based retrieval, as demonstrated in Section VII where high causal connectivity (0.708) exists between semantically dissimilar entities (0.074 cosine similarity).

The semantic distance between entities combines embedding, context, temporal, causal, and synthesis components:

$$\begin{aligned} d_s(e_1, e_2) = \alpha \cdot d_{embed}(\mu_1, \mu_2) + \beta \cdot d_{context}(\mathcal{C}_1, \mathcal{C}_2) \\ + \gamma \cdot d_{temp}(\tau_1, \tau_2) \\ + \delta \cdot d_{causal}(\Gamma_1, \Gamma_2) \\ + \epsilon \cdot d_{synth}(\Psi_1, \Psi_2) \end{aligned} \tag{1}$$

with dynamically adjusted weights $(\alpha, \beta, \gamma, \delta, \epsilon) \in \Delta^5$, ensuring flexible similarity measures for comprehensive cross-domain reasoning.

Note that the uncertainty matrix Σ_e is not included in distance calculation but rather influences confidence scoring and probabilistic consistency checks.

In practice, a high $d_{context}$ value indicates entities whose meanings shift significantly across domains; for example, ”bank” as a financial institution versus riverbank would exhibit high contextual distance despite similar embeddings. The dynamic weight adjustment allows queries to emphasize the most relevant similarity component. Temporal reasoning queries increase γ to prioritize evolution patterns, inter-domain discovery increases β to emphasize contextual alignment, and causal queries increase δ to weight mechanistic relationships. This flexibility enables the same storage representation to serve multiple reasoning modes efficiently.

B. Reasoning Primitives

Table II summarizes the five reasoning primitives, addressing Research Question 2. These operate on the semantic manifold to enable complex queries, orchestrated by an execution pipeline (Algorithm 1).

- Analogical Reasoning: Identifies structural patterns across source domain s and target domain t by maximizing the sum of similarities between mapped relations, where ϕ is the mapping function transforming source relations R_s to target relations R_t; complexity $O(|s| \cdot |t|)$ [18]. The mapping function

TABLE II
REASONDB REASONING PRIMITIVES

Primitive	Formulation
Analogical Reasoning	$A(s,t) = \max_\phi \sum_{r \in R_s} \text{similarity}(\phi(r), R_t)$
Temporal Evolution	$E_t(c) = P_c(t) - P_c(t_0) + \sum_i \lambda_i \cdot I_i(t)$
Domain Alignment Mapping	$B(e_1, e_2) = \exp\left(-\frac{d(\text{proj}(e_1),\text{proj}(e_2))^2}{2\sigma^2}\right)$
Causal Chain Discovery	$\text{Path}(X \to Y) = \prod_{\text{edge }(u,v)} w_{u,v}$
Concept Synthesis	$c_1 \oplus c_2 = \arg\min_c [d(c, c_1) + d(c, c_2) - \lambda \cdot \text{novelty}(c)]$

Algorithm 1 Semantic Reasoning Primitive Engine

```
1: procedure EXECUTESEMANTICQUERY(query $q$, semantic manifold $\mathcal{M}$)
2:     $plan \leftarrow$ ParseQueryIntent($q$)
3:     $operators \leftarrow$ SelectOperators($plan$)
4:     $pipeline \leftarrow$ OptimizePipeline($operators$)
5:     for each operator $op \in pipeline$ do
6:         $\mathcal{M} \leftarrow$ ApplyOperator($op$, $\mathcal{M}$)
7:     end for
8:     return MaterializeResults($\mathcal{M}$, $q$)
9: end procedure
```

ϕ acts as a translation dictionary: in the biology-to-engineering example from Section VII, ϕ(recognition) = detection and ϕ(modification) = correction, enabling inter-domain knowledge transfer by finding structural correspondence rather than surface similarity.

- Temporal Evolution: Tracks changes in concept c over time by measuring shifts in probability distribution $P_c(t)$ from baseline t_0, adjusted by weighted external influences $I_i(t)$ (e.g., $\lambda_1 = 0.8$ for major events, $\lambda_2 = 0.2$ for minor trends).
- Domain Alignment Mapping: Projects entities e_1 and e_2 from different domains into a shared semantic space and computes a Gaussian similarity score, where proj is the projection function and σ^2 controls matching tolerance for meaningful cross-domain comparisons.
- Causal Chain Discovery: Traces cause-effect relationships by multiplying edge weights $w_{u,v} \in [0,1]$ along paths from X to Y, representing causal influence strength (e.g., a three-edge path with weights 0.9, 0.8, 0.7 yields 0.504). This product-based formulation naturally enforces parsimony. Longer causal chains have exponentially decreasing strength, favoring direct mechanistic connections over tenuous multi-hop relationships. In practice, this enables discovering transformative pathways where start and end states are semantically dissimilar (problem vs. solution) but causally connected through intermediate mechanisms, as validated in Section VII.
- Concept Synthesis: Combines concepts c_1 and c_2 to generate novel ones by minimizing distance while maximizing novelty (measured as distance to existing concepts), with λ balancing preservation and innovation.

Algorithm 1 outlines the execution framework, transforming natural language queries into executable operations by parsing intent, selecting primitives, optimizing their sequence, and materializing results.

C. Adaptive Query Optimization Layer

Adaptive indexing optimizes query performance using contextual bandit algorithms, which balance exploration of new index strategies with exploitation of known optimal ones, addressing Research Question 3. The system observes queries, measures execution times, and learns effective indices for specific patterns (e.g., hash-based for temporal queries, graph-based for analogical). The query cost is:

$$C(q) = \sum_{op \in q} C_{comp}(op) + C_{mem}(op) + C_{io}(op) \cdot P_{selectivity}(op) \tag{2}$$

Optimal index selection is formulated as:

$$I^*(q) = \arg\max_I \mathbb{E}[\text{Performance}(q, I)] - \lambda \cdot C(I) \tag{3}$$

The contextual bandit formulation provides theoretical guarantees on learning efficiency. Regret, defined as the cumulative difference between chosen and optimal index performance, is bounded by:

$$R_T \leq 2\sqrt{dT \log\left(\frac{1 + T/\lambda d}{\delta}\right)} + \sqrt{\lambda}||\theta^*||_2 \tag{4}$$

where R_T is total regret after T queries, d is context dimensionality, δ is confidence, λ regularizes exploration, and $||\theta^*||_2$ reflects strategy complexity. This ensures performance converges to optimal as the system learns from query patterns.

D. Temporal Consistency Management

Semantic data require probabilistic consistency rather than ACID guarantees because embeddings represent continuous probability distributions over semantic space, where exact replication is neither achievable nor meaningful. Instead, replicas are consistent if their probability distributions align:

$$\text{Consistent}(R_1, R_2) \iff \int_{\mathcal{M}} |P_1(x) - P_2(x)| dx < \epsilon \tag{5}$$

This measures total variation distance across the semantic manifold $\mathcal{M}$, with ϵ as the tolerance threshold, ensuring some basis for coherence in distributed systems for scalable semantic operations.

Fig. 5 illustrates how the mathematical formulations translate into executable reasoning operations. Each entity's extended probabilistic representation enables all five reasoning primitives to compute semantic distances, temporal evolution, domain alignment mapping, causal chain discovery, and concept synthesis that would be impossible with discrete data representations.

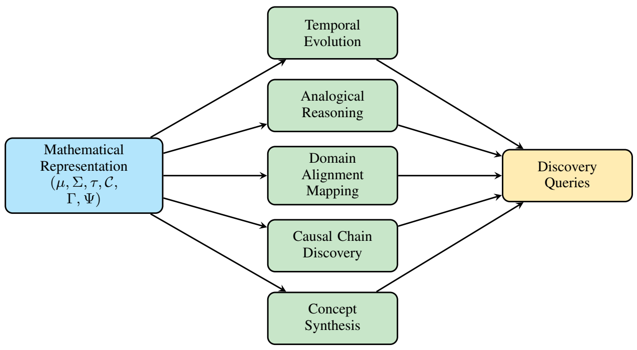

Fig. 5. From mathematical formulations to discovery operations. The extended probabilistic representation enables all five reasoning primitives to operate on semantic data.

VI. Implementation Feasibility and Scalability Analysis

ReasonDB's reasoning primitives exhibit computational complexity characteristics that require careful consideration for practical deployment. While the system demonstrates novel semantic reasoning capabilities, scalability constraints must be addressed through targeted optimization strategies.

A. Computational Complexity Constraints

Analogical reasoning operations scale as $O(|D_s|{\cdot}|D_t|{\cdot}k)$ where $|D_s|$ and $|D_t|$ are source and target domain sizes. For domains with 10,000 entities each, this translates to approximately 100 million comparison operations per query. However, adaptive indexing reduces this to $O(\log|D_s| + \log|D_t| + k)$ through hierarchical pruning, achieving sub-second response times for most analogical queries.

Domain Alignment Mapping exhibits quadratic worst-case complexity $O(n^2)$ but benefits from dimensionality reduction and GPU acceleration. Causal chain discovery scales as $O(b^h)$, bounded by branching factor b and search depth h, requiring approximate algorithms for deeper searches.

B. Memory and Storage Requirements

Vector-native storage requires d-dimensional embeddings per entity, typically 768-1536 dimensions for current embedding models. For 1 million entities with 1024-dimensional embeddings, ReasonDB requires approximately 4 GB of core storage compared to 8-12 GB for equivalent dual-storage architectures. Uncertainty matrices add $O(d^2)$ overhead per entity, but diagonal approximations reduce this to $O(d)$ with minimal quality degradation.

C. Scalability Bottlenecks and Mitigation

The primary bottleneck emerges from quadratic complexity in inter-domain operations. Key mitigation strategies include:

- GPU acceleration for batch similarity computations
- Quantized embeddings (8-bit) reducing memory by 4×
- Hierarchical domain partitioning through semantic clustering
- Approximate reasoning algorithms trading accuracy for speed

While scalability challenges exist, they represent engineering problems rather than fundamental architectural limitations. Production deployment requires continued optimization of reasoning algorithms and indexing strategies, but the core semantic reasoning approach remains viable for enterprise-scale applications.

D. Distributed Architecture and Scale Thresholds

To address scalability for production deployments, ReasonDB employs a distributed architecture with domain-partitioned indexing and parallel primitive execution. The architecture comprises five key components: a query coordinator that decomposes discovery queries into primitive sequences, domain partition servers managing entity subsets with local vector indices, distributed HNSW or IVF index shards for approximate nearest neighbor search, reasoning primitive worker pools executing analogical and causal operations in parallel, and a result aggregator merging partial discoveries across partitions. This design enables horizontal scaling while maintaining semantic coherence through consistent entity embedding spaces.

Table III specifies data volume thresholds where different mitigation strategies become necessary, based on extrapolation from prototype results (10 entities, sub-second latency, 8 GB RAM) and established vector database performance characteristics. These estimates assume 768-dimensional embeddings, typical HNSW index performance (1 M vectors queried in 10–100 ms at 95% recall), and consumer hardware (modern multi-core CPUs, consumer GPUs). Actual performance depends on query complexity, domain characteristics, hardware specifications, and quality-performance trade-offs.

TABLE III
Scalability Thresholds and Mitigation Strategies

Scale Category	Entity Count	Strategy Required	Expected Latency
Small	10^2–10^4	Single node, CPU	¡ 1 s
Medium	10^4–10^5	Quantization (8-bit)	1–5 s
Large	10^5–10^6	GPU acceleration	5–30 s
Very Large	10^6–10^7	Distributed index	30 s–5 min
Massive	10^7+	Full distributed	5–30 min

These thresholds represent order-of-magnitude estimates rather than guaranteed performance bounds. Actual latency depends on several factors: reasoning primitive complexity, embedding models producing 768 to 1536 dimensions, search parameters (beam width, quality thresholds), hardware capabilities, and domain characteristics such as entity distribution and relationship density. These scale categories correspond to common deployment scenarios: ”Medium” (10K–100K entities) for departmental knowledge bases, ”Large” (100K–1M) for enterprise document collections, and ”Very Large” (1M–10M) for scientific literature or product catalogs. Performance validation at these scales remains future work; the current prototype validates architectural feasibility at small scale with 10 entities.

VII. Analogical Reasoning and Causal Chain Discovery Prototype

This section presents a dual-component prototype that validates two complementary reasoning primitives: analogical pattern matching for structural correspondence discovery and causal chain discovery for mechanistic pathway exploration. While both operate on semantic relationships, they serve distinct but synergistic purposes. Analogical reasoning identifies structural correspondences between entities based on relational patterns (what relates to what in similar ways), enabling inter-domain knowledge transfer. Causal chain discovery traces mechanistic pathways through directed cause-effect relationships (what leads to what), enabling transformative process understanding.

The key insight from our validation is that high causal connectivity can exist between semantically dissimilar entities. This enables discovery of problem-solution pathways invisible to vector similarity methods, which assume semantic similarity implies relevance. Together, these demonstrate how semantic reasoning operations can identify meaningful connections across domains that traditional similarity-based methods fundamentally cannot detect.

A. Analogical Reasoning Validation

Analogical reasoning operates by identifying structural correspondences between relational patterns across domains. While vector databases compute relevance through embedding similarity, analogical reasoning discovers mappings between how entities relate to each other (relational structure) rather than what they are (entity similarity). This enables knowledge transfer across semantically distant domains by recognizing parallel problem-solving patterns.

1) Experimental design and methodology: The prototype uses six entities: three from biology (CRISPR immunity, DNA repair, RNA interference) and three from engineering (error correction codes, adaptive filtering, pattern recognition). The prototype (Fig. 6) follows a two-stage process: first, identify relational patterns within each domain; second, map biological patterns onto engineering patterns using a translation dictionary.

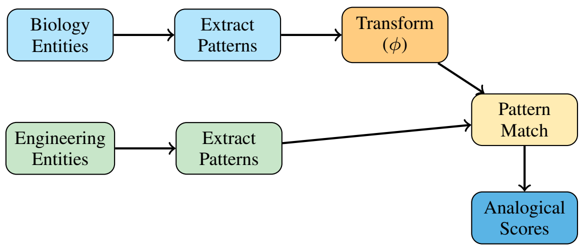

Fig. 6. Analogical reasoning: pattern extraction, transformation, and matching.

2) Domain literature validation: The discovered analogies between biological and engineering systems are not merely computationally valid but align with established inter-domain research. Goldman and colleagues demonstrated practical implementation of information storage in synthesized DNA, explicitly leveraging error correction codes analogous to those used in digital communication systems [19]. This design choice reflects the deep structural correspondence between biological error correction mechanisms (DNA repair pathways) and computational error correction algorithms (Reed-Solomon codes, checksums). The fact that ReasonDB's analogical reasoning independently rediscovered these expert-validated connections provides external validation that high structural alignment corresponds to scientifically meaningful and actionable discoveries rather than mere computational artifacts.

B. Causal Chain Discovery: Mechanistic Pathway Exploration

Causal chain discovery operates on a fundamentally different principle than similarity-based retrieval. While vector databases compute relevance through cosine similarity in embedding space, causal chain discovery traces directed mechanistic pathways through cause and effect relationships. Chain strength is computed as the product of individual causal link strengths along each path. This product-based calculation naturally favors direct connections. Longer paths accumulate exponentially lower strength.

1) Experimental design and methodology: The causal chain discovery prototype (Fig. 7) tests whether causal connectivity can exist independent of semantic similarity. We constructed two structurally parallel 4-hop causal chains from deliberately different domains (Table IV).

TABLE IV
PARALLEL CAUSAL CHAINS ACROSS DOMAINS

Domain	**Causal Chain**
Biology	bacterial_infection → pathogen_detection → CRISPR_activation → DNA_cleavage → immunity_acquired
Engineering	transmission_error → error_detection → correction_activation → data_repair → reliability_achieved

Despite low pairwise cosine similarities between corresponding concepts (mean: 0.074, range: 0.031–0.128), both chains share identical causal structure. Detection triggers activation, activation enables correction, correction produces the desired outcome. The experimental design deliberately maximizes this challenge by constructing chains that begin and end at semantically dissimilar states (error/failure vs. reliability/success), forcing the discovery algorithm to rely purely on causal structure rather than embedding similarity. This tests whether causal chain discovery can identify mechanistic parallels that similarity search cannot detect.

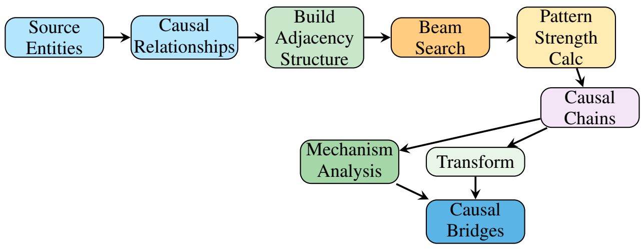

Fig. 7. Causal chain discovery architecture showing temporal ordering, pathway search, and inter-domain knowledge transfer.

2) The divergence from vector similarity: The engineering domain reveals the inherent restrictions of similarity-based methods. A vector database assigns relevance score 0.074 to a connection with causal chain strength 0.708. This 0.634 divergence (nearly an order of magnitude) occurs because vector similarity measures correlation in semantic space, not causation.

Vector databases fail in this regard. In transformative processes, start states (error, failure, corruption) and end states (reliability, success, achievement) occupy different or opposite regions of semantic space. Their embeddings have low cosine similarity despite a strong causal pathway connecting them through intermediate states (detection → activation → repair).

3) Inter-domain causal bridging: The most powerful application combines analogical and causal reasoning. Our Biology → Engineering bridge achieves 0.794 mechanism transfer strength with 5/11 shared mechanisms after transformation (recognition → detection, modification → correction, immunity → protection). This enables knowledge transfer. Understanding CRISPR immunity mechanisms informs error correction system design.

C. Combined Prototype Results

The combined prototype demonstrates two critical capabilities working synergistically.

1) Analogical reasoning performance: Success is measured using a structural alignment score based on Gentner's Structure Mapping Theory [7], which evaluates analogical quality through three components: systematicity (whether higher-order relations are preserved), one-to-one mapping (whether each source element maps to exactly one target element), and pragmatic relevance (whether the mapping serves the discovery goal). The final score ranges from 0 to 1, where scores above 0.3 indicate meaningful analogies according to validation studies on scientific analogies [20].

Analogical reasoning achieves an average structural alignment score of 0.207 compared to vector similarity's 0.030, representing 7× better performance for finding meaningful inter-domain connections (Fig. 8). Fig. 9 shows pattern transformation quality scores range from 0.28 to 0.38, with 100% of mappings achieving "good quality" status (above the 0.30 threshold, where scores above 0.30 indicate meaningful analogies according to validation studies on scientific analogies [20, 21]), demonstrating that pattern translation across domains successfully bridges conceptual gaps between domains.

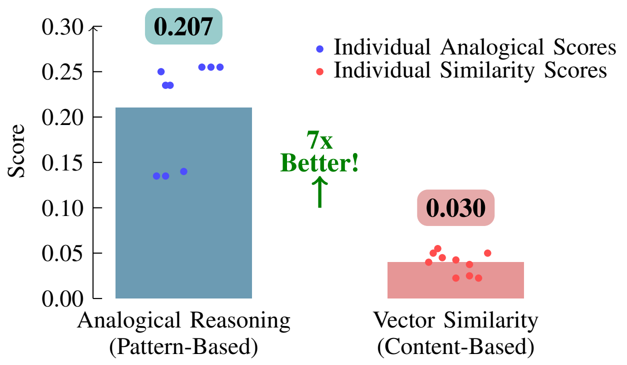

Fig. 8. Cross-domain comparison: Analogical reasoning vs. vector similarity

2) Causal chain discovery performance: Table V presents the critical finding. Causal chain discovery succeeds precisely where vector similarity fails.

Causal chain discovery traces mechanistic pathways with 0.708 chain strength in the engineering domain despite only 0.074 semantic similarity between endpoints, demonstrating discovery where vector similarity fundamentally fails. The Biology → Engineering bridge achieves 0.794

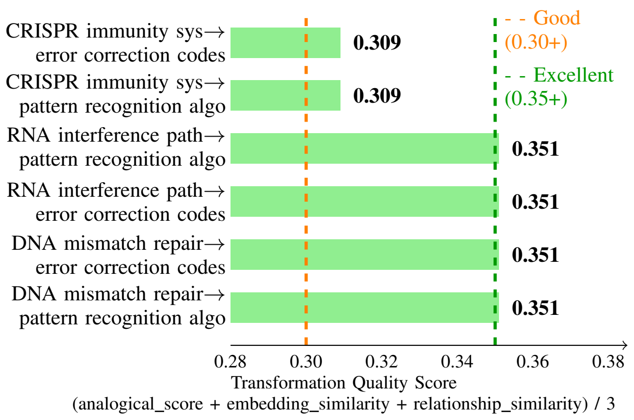

Fig. 9. Pattern transformation, biology to engineering mappings.

TABLE V
CAUSAL CHAIN DISCOVERY VS. VECTOR SIMILARITY

Domain	Chain Strength	Cosine Similarity	Δ (Divergence)
Biology	0.692	0.719	0.027
Engineering	0.708	0.074	0.634
Biology→Engineering	0.794	-0.050	0.844

mechanism transfer strength across domains through these discovered causal pathways.

3) Execution timing and efficiency: Empirical execution times from the prototype validate its efficiency. For the analogical pattern extraction process on a dataset of 6 entities (3 each from biology and engineering), total execution time was 2.21 ms, comprising 0.35 ms for biology pattern extraction, 0.23 ms for engineering pattern extraction, 1.24 ms for pattern embedding computation, and 0.39 ms for cross-domain analogical scoring. This yields a throughput of 5651 operations per second, with pattern extraction rates at 9077 patterns/second.

For the causal chain discovery process on 10 entities with 8 causal relationships across two domains, total execution time was 0.40 ms, including 0.06 ms for biology chain discovery, 0.16 ms for engineering chain discovery, and 0.18 ms for cross-domain bridge creation, achieving a throughput of 9077 chains/second and 5651 bridges/second. Figure 10 illustrates the execution time breakdown and comparative throughput across operations. These results confirm sub-second response times for small-scale operations, consistent with the theoretical complexity analysis in Section VI.

4) Synthesis: Discovery beyond similarity: Together, these primitives enable a new query class: "What causal mechanisms from analogous systems in domain A could solve problems in domain B?" Analogical reasoning discovers structural correspondences for knowledge transfer ("biological system X solves problems similar to engineering challenge Y"), measured by structural alignment score (0.207 vs. 0.030 for similarity search). Causal chain discovery traces mechanistic pathways for intervention design ("this sequence of mechanisms transforms problem state A to solution state B"), measured by chain strength independent of endpoint similarity (0.708 strength despite 0.074 similarity). This combined capability is impossible

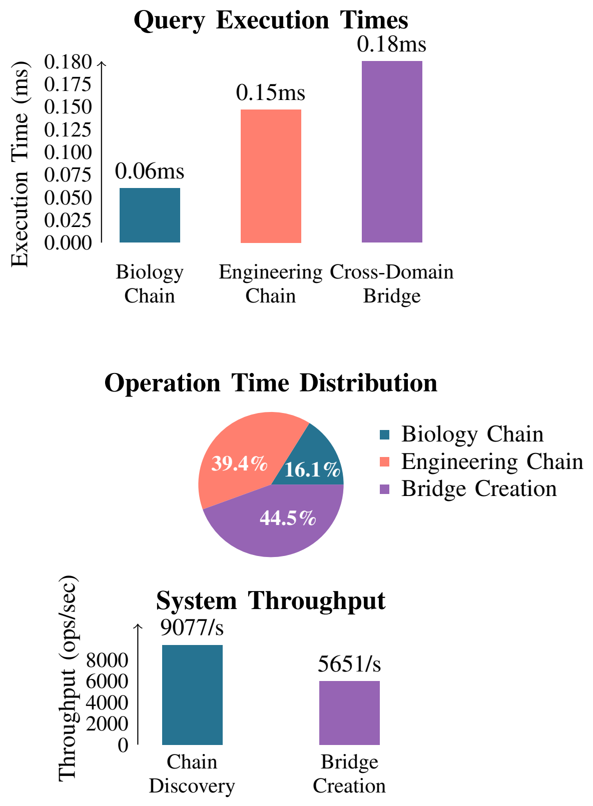

Fig. 10. Causal chain discovery: Efficiency metrics on 10-entity dataset.

with either similarity search or knowledge graph traversal alone.

D. Application Domains

Beyond the biology-engineering validation, ReasonDB's approach suggests potential applications in domains requiring cross-domain analogical discovery. While these applications remain to be validated, they illustrate the breadth of problems addressable through semantic reasoning primitives.

In pharmaceutical research, ReasonDB would identify functional similarities between compounds that work in completely different therapeutic areas. Instead of just comparing molecular structures, the system analyzes how mechanisms actually work; for instance, it could discover that antifungal pathways and cancer resistance mechanisms solve problems in structurally similar ways. This adds a new dimension to traditional structure-based screening.

Financial institutions could apply ReasonDB to detect emerging fraud patterns by finding structural similarities across different markets. For example, the system could recognize that certain cryptocurrency manipulation schemes follow the same playbook as historical commodity market frauds, enabling earlier intervention. Causal chain discovery can trace how instabilities in one region ripple through seemingly unrelated financial instruments.

E. Constraints, Implementation Details and Reproducibility

While ReasonDB demonstrates promising semantic reasoning capabilities, practical deployment faces several constraints including computational complexity, memory requirements, and accuracy trade-offs. These limitations define operational boundaries that require further experimentation and optimization before production systems can be fully characterized. Comprehensive performance evaluation and scalability testing remain essential for understanding the system's practical deployment.

The prototype implementation uses a modern Python-based stack optimized for semantic reasoning operations. Core computational operations leverage NumPy, SciPy, and scikit-learn for vector mathematics and probabilistic computations. Entity embeddings are generated using Ollama with the nomic-embed-text model, providing 768-dimensional semantic representations. Vector-native storage uses LMDB (Lightning Memory-Mapped Database) for high-performance key-value operations with minimal overhead, while comparison baselines employ ChromaDB for vector similarity search and NetworkX for graph-based traversal. Data processing and visualization use Pandas for tabular operations and Matplotlib with Seaborn for results plotting.

The prototype requires 8 GB RAM minimum for optimal performance, enabling sub-second query response times for the experimental workload (10 entities, 4-hop causal chains). Dependency management uses uv for reproducible Python environments, ensuring consistent results across deployments. All timing measurements were conducted on consumer laptops (MacBook Pro M1 Max 64Gb), demonstrating that semantic reasoning primitives achieve practical performance without specialized infrastructure.

Complete implementation, experimental data, and reproduction instructions are available at https://github.com/crywolfe/ReasonDB-prototype.

VIII. ReasonDB in Context: AI-Enhanced Vector Systems vs. Native Semantic Reasoning Primitives

The foundational limitations of vector databases and knowledge graphs for semantic reasoning were discussed in Section II, which established why current systems cannot support analogical reasoning or causal chain discovery. Section IV contrasted ReasonDB's AI-native architecture with recent AI-enhanced approaches like LOTUS [15, 16] and AnDB [17], clarifying how treating reasoning as a first-class operation rather than an enhancement layer enables qualitatively different discovery capabilities.

This architectural foundation positions ReasonDB within the broader evolution of database systems toward semantic reasoning. While vector databases optimize for similarity search and knowledge graphs enable relationship traversal, ReasonDB introduces reasoning primitives as native database operations. This enables discovery queries—such as cross-domain mechanism transfer and causal pathway exploration—that current systems cannot express declaratively.

The experimental validation in Section VII demonstrates these capabilities concretely: 7× improvement in analogical reasoning over similarity search, and successful causal chain discovery where vector similarity fundamentally fails (0.708 chain strength despite 0.074 cosine similarity). These results validate that semantic reasoning requires fundamentally different database primitives than those provided by current architectures, whether enhanced with AI capabilities or not.

IX. Concluding Remarks

A. Core Architecture

ReasonDB demonstrates that semantic reasoning requires fundamentally different database architectures. By treating embeddings as primary data and implementing reasoning as first-class operations, the system enables discovery queries impossible with current architectures. Three architectural innovations work synergistically: vector-native storage eliminates dual-storage overhead while preserving semantic relationships, reasoning primitives implement analogical and causal operations as native database functions, and adaptive indexing dynamically optimizes for diverse reasoning patterns.

B. Validation and Breakthrough

Experimental validation demonstrates two breakthrough capabilities. Analogical reasoning achieves 7× performance improvement over similarity search (structural alignment 0.207 vs. 0.030), enabling inter-domain knowledge transfer through structural pattern mapping. Causal chain discovery succeeds where similarity-based methods fundamentally fail, tracing mechanistic pathways between semantically dissimilar states (0.708 chain strength despite 0.074 cosine similarity). Together, these results validate that current database architectures cannot support semantic discovery operations.

C. Practical Implementation

ReasonDB complements rather than replaces traditional databases. Production deployments use hybrid architectures where query routers direct transactional queries requiring ACID properties to traditional systems while routing discovery queries requiring semantic reasoning to ReasonDB. Data synchronization layers maintain consistency across systems. This division enables each system to optimize for its computational strengths: deterministic transactions versus probabilistic discovery.

D. Future Directions

Three critical areas require development. First, full production implementation integrating vector-native storage with real-time updates, optimized reasoning engines, and the remaining primitives (temporal evolution, domain alignment mapping, concept synthesis). Second, systematic benchmarks measuring discovery quality, inter-domain transfer effectiveness, and causal reasoning accuracy across diverse scales. Third, domain validation in pharmaceutical drug repurposing, financial fraud detection, and scientific literature analysis to characterize performance envelopes and optimal application scenarios. These advances will transform databases from passive repositories into active discovery partners.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

All authors collaborated extensively throughout the conceptual development, writing, and review processes, and approved the final manuscript. Gerry Wolfe conceived the original idea, led the writing, and developed the prototypes. Gerry Wolfe and Ashraf Elnashar provided critical insights during the research and writing phases, shaping the manuscript collaboratively. Ashraf Elnashar contributed deep expertise to refine research concepts and sharpen the manuscript's direction. William Schreiber provided valuable feedback, sourced and managed the bibliography and references, and ensured their accuracy and completeness. All authors had approved the final version.

References

[1] Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Tracing the thoughts of a large language model. *Anthropic*, March 2025.

[2] Vleer Doing and Ryan Wisnesky. Towards a more reasonable semantic web. *arXiv preprint arXiv:2407.19095*, 2024.

[3] Oluwafemi Oloruntoba. Ai-driven autonomous database management: Self-tuning, predictive query optimization, and intelligent indexing in enterprise it environments. *World Journal of Advanced Research and Reviews*, 25:1558–1580, 02 2025.

[4] Toni Taipalus. Vector database management systems: Fundamental concepts, use-cases, and current challenges. *Cognitive Systems Research*, 85:101216, 2024.

[5] Andreas Eibeck, Arkadiusz Chadzynski, Mei Qi Lim, Kevin Aditya, Laura Ong, Aravind Devanand, Gourab Karmakar, Sebastian Mosbach, Raymond Lau, Iftekhar A Karimi, et al. A parallel world framework for scenario analysis in knowledge graphs. *Data-Centric Engineering*, 1:e6, 2020.

[6] Nicolas Bschor. *Enabling Semantic-Aware Query Evaluation in a Traditional Database Framework.* PhD thesis, Technische Universität Wien, 2025.

[7] Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 1983.

[8] Munshi Azad Hossain. Crispr-cas9: A fascinating journey from bacterial immune system to human gene editing. *Progress in Molecular Biology and Translational Science*, 178:63–83, 2021.

[9] Robert Bogue. Biomimetic adhesives: a review of recent developments. *Assembly Automation*, 28(4):282–288, 2008.

[10] Ted T Ashburn and Karl B Thor. Drug repositioning: identifying and developing new uses for existing drugs. *Nature Reviews Drug Discovery*, 3(8):673–683, 2004.

[11] Naoufal Rtayli and Nourddine Enneya. Enhanced credit card fraud detection based on svm-recursive feature elimination and hyper-parameters optimization. *Journal of Information Security and Applications*, 55:102596, 2020.

[12] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[13] Lipyeow Lim, Haixun Wang, and Min Wang. Semantic queries in databases: problems and challenges. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1505–1508, 2009.

[14] Dan Suciu. *Probabilistic Databases*, pages 1–7. Springer New York, New York, NY, 2016.

[15] Liana Patel, Siddharth Jha, Carlos Guestrin, and Matei Zaharia. Lotus: Enabling semantic queries with llms over tables of unstructured and structured data. *arXiv preprint arXiv:2407.11418*, 2024.

[16] Adnan Masood. Lotus: Semantic operators for ai-powered data processing — a technical review. *Medium*, 2024.

[17] Tianqing Wang, Xun Xue, Guoliang Li, and Yong Wang. Andb: Breaking boundaries with an ai-native database for universal semantic analysis. *arXiv preprint arXiv:2502.13805*, 2025.

[18] Kenneth D Forbus, Dedre Gentner, and Keith Law. Mac/fac: A model of similarity-based retrieval. *Cognitive science*, 19(2):141–205, 1995.

[19] Nick Goldman, Paul Bertone, Siyuan Chen, Christophe Dessimoz, Emily M LeProust, Botond Sipos, and Ewan Birney. Towards practical, high-capacity, low-maintenance information storage in synthesized dna. *Nature*, 494(7435):77–80, 2013.

[20] Kevin Dunbar. How scientists really reason: Scientific reasoning in real-world laboratories. In *The nature of insight*, pages 365–395. MIT Press, 1995.

[21] Mark T Keane, Tim Ledgeway, and Stuart Duff. Constraints on analogical mapping: A comparison of three models. *Cognitive Science*, 18(3):387–438, 1994.