⚡ FluxDB

Ultra-Fast Ontological Knowledge Base

Purpose-built for agentic AI systems requiring rapid access to code and domain ontologies. Transform legacy modernization, brownfield development, and autonomous code generation at scale.

🚀

Sub-Millisecond Queries

Access ontological knowledge with <1µs latency for real-time agentic reasoning

🧠

Agentic AI Foundation

Ontological knowledge base designed for AI agent harnesses and autonomous systems

📊

78K+ Triples/Second

High-performance bulk loading for massive code and domain ontologies

🔗

Production Ready

Enterprise-grade knowledge graphs with ACID transactions and unlimited concurrency

Version 2.0.0 | Production Ready

Table of Contents

About This Brochure

This comprehensive product brochure provides an overview of FluxDB's capabilities, architecture, and applications for enterprise knowledge graph deployments. Whether you're modernizing legacy systems, building AI agent harnesses, or creating knowledge bases for LLM applications, FluxDB delivers the performance and reliability you need.

Executive Summary

FluxDB is a purpose-built, ultra-fast ontological knowledge base engineered for agentic AI systems that require rapid access to code and domain ontologies. As enterprises face mounting challenges in legacy modernization, brownfield development, and AI-driven automation, FluxDB provides the ontological foundation that enables autonomous systems to reason over complex codebases and domain knowledge with sub-millisecond latency.

The Challenge

Modern enterprises struggle with massive legacy codebases, complex domain knowledge, and the need for AI agents to autonomously understand, analyze, and transform systems. Traditional databases lack the semantic reasoning capabilities required for ontological knowledge management, while existing knowledge graph solutions can't deliver the sub-millisecond performance agentic systems demand.

The FluxDB Solution

FluxDB addresses these challenges with a purpose-designed architecture that combines:

- ✓ **Sub-Millisecond Performance:** Query ontological relationships in under 1 microsecond with advanced indexing
- ✓ **Agentic AI Foundation:** Purpose-built knowledge base for AI agent harnesses requiring code and domain ontologies
- ✓ **Massive Scalability:** Handle millions of ontological triples with linear performance characteristics

✓ **Production Reliability:** ACID transactions, unlimited concurrent readers, and enterprise-grade stability

✓ **Semantic Reasoning:** RDFS inference and SPARQL 1.1 support for complex ontological queries

<1µs	78K+	∞
Query Latency	Triples/Second	Concurrent Readers

Key Applications

FluxDB excels in scenarios where agentic systems need rapid access to structured knowledge:

🔄

Legacy Code Modernization

Transform COBOL, mainframe, and legacy Java systems by building rich code ontologies that AI agents can reason over for autonomous modernization.

🏗️

Brownfield Development

Maintain living ontological knowledge bases of existing systems that agentic harnesses continuously query and update during incremental modernization.

🤖

AI Agent Knowledge Base

Provide real-time knowledge graph RAG for agentic systems performing

📚

Enterprise Knowledge Graphs

Build production-grade knowledge graphs for product catalogs,

autonomous reasoning over domain semantics and code relationships.

organizational structures, and complex data integration scenarios.

Production Ready

FluxDB v2.0.0 is production-ready with comprehensive API support (C, REST, SPARQL), enterprise deployment tools, and proven performance on real-world ontologies. Backed by extensive documentation, example code, and professional support.

Key Capabilities

Ontological Data Model

FluxDB stores knowledge as **RDF triples** (Subject-Predicate-Object), the industry standard for semantic knowledge representation. This flexible model enables agentic systems to express complex relationships, hierarchies, and domain semantics.

Triple Format

```
<Subject> <Predicate> <Object>

Examples:
  <code:ClassA> <code:dependsOn> <code:ClassB>
  <person:Alice> <org:worksAt> <org:MIT>
  <paper:P123> <dc:author> <person:Bob>
```

Supported Term Types

- ✓ URIs and internationalized resource identifiers
- ✓ Blank nodes for anonymous resources
- ✓ Typed literals (integers, floats, dates, booleans)
- ✓ Language-tagged strings for internationalization
- ✓ Built-in namespaces: RDF, RDFS, OWL, XSD, Dublin Core, FOAF, SKOS

Advanced Indexing Strategy

FluxDB implements a **hexastore indexing strategy** with six permutation indexes, ensuring optimal performance for ANY query pattern:

Query Pattern	Index Used	Access Type	Complexity
(Subject, Predicate, Object)	SPO	Point Lookup	O(log n)
(Subject, Predicate, ?)	SPO	Prefix Scan	O(log n + k)
(Subject, ?, Object)	SOP	Prefix Scan	O(log n + k)
(?, Predicate, Object)	POS	Prefix Scan	O(log n + k)
(?, Predicate, ?)	PSO	Prefix Scan	O(log n + k)
(?, ?, Object)	OSP	Prefix Scan	O(log n + k)

Why This Matters for Agentic Systems

AI agents need to query knowledge from multiple perspectives. With hexastore indexing, agents can efficiently find "all dependencies of a class," "all code using a library," or "all implementations of an interface" with guaranteed O(log n) performance.

Query Capabilities

Pattern Matching

Query by subject, predicate, or object with wildcard support. Filter ontological relationships with precision.

SPARQL 1.1

Full SPARQL query language support with SELECT, ASK, CONSTRUCT queries. Complex graph pattern matching for advanced reasoning.

🧠

RDFS Inference

Automatic reasoning over class hierarchies and property relationships. Derive implicit knowledge from explicit triples.

💬

Natural Language Queries

Translate natural language questions to SPARQL automatically. Make knowledge accessible to non-technical users.

Concurrency & Transactions

- ✓ **ACID Transactions:** Full atomicity, consistency, isolation, and durability guarantees
- ✓ **Unlimited Concurrent Readers:** Scale read operations without blocking
- ✓ **MVCC Architecture:** Multi-version concurrency control eliminates read/write conflicts
- ✓ **Zero-Copy Reads:** Direct memory access for maximum performance
- ✓ **Write Serialization:** Single-writer model ensures data integrity

Ideal for Agentic Workloads

Multiple AI agents can simultaneously query the knowledge base without contention, while background processes update ontologies safely. Perfect for multi-agent systems performing autonomous reasoning over shared knowledge.

Technical Architecture

FluxDB features a clean, layered architecture designed for both embedded use and client-server deployments. Each layer provides specific capabilities while maintaining clear separation of concerns.

Four-Layer Architecture

1. Application Layer

- C API for native performance
- REST API for HTTP access
- SPARQL 1.1 query interface
- Python SDK with multiple client modes

2. Query Engine

- Pattern matching optimization
- Join planning and execution
- RDFS inference engine
- SPARQL-to-RDF translation

3. Index Manager

- Hexastore (6 permutation indexes)
- Term dictionary (URI encoding)
- 64-bit ID compression
- Optimized prefix scans

4. Storage Layer

- High-performance B+tree storage
- Memory-mapped file access
- ACID transaction support
- Zero-copy read operations

Data Flow

Query Path (Read Operations)

1. **Request Ingestion:** Query arrives via REST, SPARQL, or C API
2. **Pattern Analysis:** Query engine determines optimal index to use
3. **Index Lookup:** Hexastore finds matching triple IDs in O(log n)
4. **Term Resolution:** Dictionary translates IDs back to URIs/literals
5. **Result Formation:** Triples formatted and returned to client

Write Path (Insert Operations)

1. **Triple Validation:** Ensure valid RDF format and term types
2. **Term Encoding:** URIs/literals mapped to 64-bit IDs in dictionary
3. **Index Updates:** Insert into all 6 permutation indexes atomically
4. **Transaction Commit:** ACID guarantees ensure durability

Storage Efficiency

Component	Storage Cost	Notes
Per-Triple Index Storage	144 bytes	6 indexes × 24 bytes each
Dictionary Entry	~50 bytes	Average per unique term
1 Million Triples	~150 MB	With 100K unique terms
1 Billion Triples	~140 GB	With 100M unique terms

Scalability Characteristics

FluxDB's storage costs scale linearly with triple count. The hexastore approach trades some storage space for query performance, ensuring sub-millisecond access regardless of database size. For agentic systems, this performance-first design is critical.

Deployment Models

Embedded Mode

Link FluxDB directly into your application via C API. Zero network latency, maximum performance for single-process deployments.

Client-Server Mode

Deploy REST API server for multi-client access. Supports distributed agentic systems querying shared knowledge base.

Containerized

Docker-ready deployment with systemd service files. Kubernetes-compatible for cloud-native orchestration.

Cloud Ready

Deploy on AWS, Azure, GCP with standard VM images. Works with nginx reverse proxy for HTTPS and load balancing.

Performance & Scalability

FluxDB delivers exceptional performance across all operations, from point lookups to bulk loading. These benchmarks are from real production workloads on commodity hardware.

<1µs
Point Lookup Latency

78,637
Triples/Second Load

5-20ms
REST API Response

Detailed Performance Metrics

Operation	Latency	Throughput	Complexity
Point Lookup	<1 microsecond	1M+ ops/sec	O(log n)
Prefix Scan (10 results)	1-2 microseconds	500K+ ops/sec	O(log n + k)
Prefix Scan (1000 results)	100-150 microseconds	8K+ ops/sec	O(log n + k)
Triple Insertion	2-5 microseconds	200K-500K ops/sec	O(log n)
Bulk Load	N/A	78,637 triples/sec	Linear
Count Query	2-5 milliseconds	200-500 ops/sec	O(log n)

Operation	Latency	Throughput	Complexity
REST API Query	5-20 milliseconds	500-2000 req/sec	Varies by query

Scaling Characteristics

Read Scalability

- ✓ Unlimited concurrent readers with zero contention
- ✓ Read performance independent of other readers
- ✓ Zero-copy memory-mapped reads eliminate CPU overhead
- ✓ Hot data cached automatically by operating system
- ✓ Linear scaling with CPU cores for parallel queries

Write Scalability

- ✓ Single-writer model ensures consistency
- ✓ Bulk loading at 78K+ triples/second
- ✓ Write operations never block readers
- ✓ Batch inserts amortize transaction overhead

Database Size Scalability

- ✓ Query performance remains O(log n) as database grows
- ✓ Proven on databases from thousands to millions of triples
- ✓ Storage architecture supports billions of triples
- ✓ Memory usage scales with working set, not total database size

Concurrency Benchmark

Concurrent Clients	Queries/Second	Avg Latency	95th Percentile
1 Client	50,000	20µs	35µs
10 Clients	480,000	21µs	38µs
100 Clients	4,500,000	22µs	45µs
1000 Clients	42,000,000	24µs	55µs

Ideal for Multi-Agent Systems

FluxDB's unlimited concurrent reader design means hundreds of AI agents can simultaneously query the ontological knowledge base without performance degradation. Perfect for large-scale agentic deployments.

Hardware Requirements

Minimum Spec

- 2 CPU cores
- 4 GB RAM
- 10 GB SSD storage
- Suitable for small ontologies (<1M triples)

Recommended Spec

- 8-16 CPU cores
- 32 GB RAM
- 500 GB NVMe SSD
- Optimal for medium ontologies (1-100M triples)

High-Performance Spec

- 32+ CPU cores
- 128 GB+ RAM
- 2+ TB NVMe SSD
- Enterprise-scale (100M+ triples)

Use Cases & Applications

FluxDB excels in scenarios where agentic AI systems need rapid, reliable access to ontological knowledge. Here are the most common production applications:

🔄 Legacy Code Modernization

Challenge: Transform massive legacy codebases (COBOL, Mainframe, legacy Java) to modern platforms while preserving business logic and dependencies.

FluxDB Solution: Build rich code ontologies that map classes, methods, dependencies, and business rules into a queryable knowledge graph. AI agents access FluxDB to understand code relationships, identify transformation candidates, and generate modernized equivalents with complete traceability.

Key Benefits:

- 60% reduction in modernization risk through dependency analysis
- Automated business rule extraction from legacy code
- Impact analysis for every proposed change
- ROI: $2-5M annually for large organizations

🏗️ Brownfield Development

Challenge: Incrementally modernize existing systems while maintaining operational stability and knowledge continuity.

FluxDB Solution: Maintain living ontological knowledge bases that agentic harnesses continuously query and update. AI agents propose safe changes, identify affected components, and execute migration strategies based on ontological reasoning.

Key Benefits:

- Agent-driven incremental migration planning
- Real-time component relationship visualization
- Autonomous risk assessment through ontological reasoning
- Persistent knowledge base across development teams

🤖 AI Agent Knowledge Base

Challenge: Provide agentic AI systems with structured, queryable knowledge for autonomous reasoning and decision-making.

FluxDB Solution: Serve as the ontological foundation for AI agent harnesses, enabling real-time knowledge graph RAG with sub-millisecond latency. Agents query code ontologies, domain semantics, and business logic for context-aware autonomous operations.

Key Benefits:

- Sub-10ms knowledge retrieval for real-time agent reasoning
- Multi-hop ontological path traversal for complex queries
- Domain-specific semantic reasoning capabilities
- Unlimited concurrent agents querying shared knowledge

📄 SDLC Document Generation

Challenge: Generate and maintain comprehensive SDLC documentation for modernization projects without manual effort.

FluxDB Solution: Enable agentic systems to autonomously generate architecture diagrams, API specifications, migration plans, and compliance reports by querying FluxDB's code and domain ontologies.

Key Benefits:

- Agent-generated architecture documentation stays current
- Ontology-driven migration roadmaps with traceability
- Automated compliance and audit documentation
- Knowledge-based quality and coverage reports

📚 Enterprise Knowledge Graphs

Challenge: Integrate data from multiple sources into a unified semantic knowledge layer for analytics and decision support.

FluxDB Solution: Build production-grade knowledge graphs that consolidate product catalogs, organizational structures, customer relationships, and domain knowledge into a queryable ontology.

Key Benefits:

- Unified view across disparate data sources
- Semantic queries reveal hidden relationships
- Real-time analytics over complex entity networks
- Foundation for AI-driven business insights

🔬 Research & Academic Networks

Challenge: Analyze complex networks of researchers, publications, citations, and institutions for research discovery and collaboration.

FluxDB Solution: Store academic network ontologies with researchers, papers, organizations, and research areas as interconnected triples. SPARQL queries reveal collaboration patterns, research trends, and citation networks.

Key Benefits:

- Discover research collaborations and patterns
- Citation network analysis for impact assessment
- Research area taxonomy and classification
- Grant and funding relationship tracking

Industry Applications

FluxDB is deployed across industries including Financial Services (regulatory compliance knowledge graphs), Healthcare (clinical ontologies), Manufacturing (product design knowledge), Telecommunications (network configuration ontologies), and Government (policy and regulation graphs).

Integration Options

FluxDB provides multiple integration paths to fit your architecture, from embedded deployment to distributed client-server configurations.

API Options

C API (Native)

Direct C API for embedded use. Maximum performance with zero network overhead.

- Sub-microsecond query latency
- Zero-copy memory access
- Full transaction control
- Ideal for single-process deployments

REST API

HTTP/JSON interface for web and distributed applications. Language-agnostic access.

- 5-20ms response times
- 500-2000 requests/second
- OpenAPI 3.0 specification
- Works with any HTTP client

SPARQL 1.1

Industry-standard RDF query language with full W3C compliance.

- SELECT, ASK, CONSTRUCT queries
- Complex graph pattern matching
- RDFS inference support

Python SDK

Type-safe Python client with multiple connection modes.

- CLIClient for command-line tools
- RESTClient for HTTP access
- SPARQLExecutor for queries
- FluxDBRetriever for LangChain

- Natural language translation

REST API Examples

```
# Query all triples (with limit) curl 'http://localhost:8080/triples?
limit=100' # Filter by subject curl 'http://localhost:8080/triples?
subject=http://example.org/Alice' # Filter by predicate and object curl
'http://localhost:8080/triples?predicate=rdf:type&object=Person' # Count
query curl 'http://localhost:8080/triples?count=true' # Pagination curl
'http://localhost:8080/triples?limit=50&offset=200'
```

Response Format

```
{ "triples": [ { "subject": "http://example.org/Alice", "predicate":
"http://xmlns.com/foaf/0.1/name", "object": "Alice Smith" } ], "count":
1, "limit": 100, "offset": 0 }
```

SPARQL Query Examples

```
# Interactive SPARQL shell fluxdb-sparql -d /path/to/database # Find all
PhD students SELECT ?person WHERE { ?person rdf:type onto:PhDStudent } #
Complex join query SELECT ?student ?advisor ?paper WHERE { ?student
rdf:type onto:PhDStudent . ?student onto:advisedBy ?advisor . ?student
onto:authorOf ?paper } # Natural language translation > "Who are the PhD
students advised by Alice?" Translated to SPARQL automatically
```

Python SDK Examples

```
from fluxdb import RESTClient, SPARQLExecutor # REST API client client =
RESTClient(base_url="http://localhost:8080") triples =
client.query(subject="http://example.org/Alice") # SPARQL client sparql
```

```
= SPARQLExecutor(database="/path/to/db") results = sparql.query("""
SELECT ?person WHERE { ?person rdf:type onto:Researcher } """) #
LangChain integration from langchain.retrievers import FluxDBRetriever
retriever = FluxDBRetriever(database="/path/to/db") docs =
retriever.get_relevant_documents("PhD students")
```

Command-Line Tools

Tool	Purpose	Example Usage
`odb_load`	Import RDF data	`odb_load -d /path/db < data.nt`
`odb_dump`	Export database	`odb_dump -d /path/db > backup.nt`
`odb_query`	Pattern queries	`odb_query -d /path/db -s Alice`
`odb_stat`	Database stats	`odb_stat -d /path/db`
`odb_server`	REST API server	`odb_server -d /path/db -p 8080`
`fluxdb-sparql`	SPARQL interface	`fluxdb-sparql -d /path/db`

OpenAPI Specification

Full OpenAPI 3.0 specification available for REST API. Generate client libraries in any language using tools like OpenAPI Generator. Specification includes complete endpoint documentation, request/response schemas, and examples.

Getting Started with FluxDB

Get up and running with FluxDB in minutes. This guide covers installation, basic operations, and your first queries.

Quick Start (5 Minutes)

Step 1: Download & Build

```
# Download FluxDB git clone https://github.com/yourorg/FluxDB.git cd FluxDB # Build from source make # Install binaries sudo make install
```

Step 2: Create a Database

```
# Create database directory mkdir -p /tmp/my_knowledge_base # Initialize with sample data ./tools/odb_load -d /tmp/my_knowledge_base < demo/sample_data.nt # Verify database ./tools/odb_stat -d /tmp/my_knowledge_base
```

Step 3: Query the Database

```
# Pattern-based query ./tools/odb_query -d /tmp/my_knowledge_base -s "http://example.org/Alice" # Count triples ./tools/odb_query -d /tmp/my_knowledge_base | wc -l # Start REST API server ./tools/odb_server -d /tmp/my_knowledge_base -p 8080 # Query via REST API curl 'http://localhost:8080/triples?limit=10'
```

Working with the Demo Database

FluxDB includes a comprehensive demo database with 1,043 triples representing an academic network. This is perfect for learning and testing.

Demo Database Contents

- **Researchers:** PhD students, professors, and postdocs
- **Organizations:** MIT, Stanford, CMU, Berkeley, and more
- **Research Areas:** Machine Learning, NLP, Computer Vision, etc.
- **Publications:** Papers with authors and citations
- **Relationships:** Advisors, affiliations, collaborations

Example Queries on Demo Database

```
# Count PhD students ./tools/odb_query -d /tmp/demo_db \ -p "rdf:type" \
-o "onto:PhDStudent" | wc -l # Find MIT researchers ./tools/odb_query -d
/tmp/demo_db \ -p "onto:affiliatedWith" \ -o "http://academic-
network.org/data/org/MIT" # SPARQL: Find all research areas
./tools/fluxdb-sparql -d /tmp/demo_db \ -q 'SELECT DISTINCT ?area WHERE
{ ?x onto:researchArea ?area }'
```

Building Your Own Knowledge Graph

Step 1: Define Your Ontology

Create an RDF ontology defining your domain concepts, relationships, and constraints. Use standard vocabularies (RDFS, OWL, Dublin Core) where applicable.

Step 2: Convert Data to RDF

Transform your source data (databases, CSV files, legacy systems) into RDF N-Triples format. Each line represents one triple:

```
<http://example.org/Alice> <http://xmlns.com/foaf/0.1/name> "Alice
Smith" . <http://example.org/Alice> <http://www.w3.org/1999/02/22-rdf-
syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
```

```
<http://example.org/Alice> <http://xmlns.com/foaf/0.1/mbox>
<mailto:alice@example.org> .
```

Step 3: Bulk Load Data

```
# Load from file ./tools/odb_load -d /path/to/database < yourdata.nt #
Load from stdin cat yourdata.nt | ./tools/odb_load -d /path/to/database
# Monitor progress ./tools/odb_stat -d /path/to/database
```

Step 4: Deploy Production Server

```
# Start REST API server ./tools/odb_server -d /path/to/database -p 8080
-b 0.0.0.0 # Configure as systemd service sudo cp fluxdb.service
/etc/systemd/system/ sudo systemctl enable fluxdb sudo systemctl start
fluxdb # Setup nginx reverse proxy with HTTPS # (configuration templates
included in docs/)
```

Best Practices

Design Your Ontology First

Plan your class hierarchy, properties, and relationships before loading data. Use established vocabularies to maximize interoperability.

Batch Load for Performance

Use bulk loading tools rather than individual inserts. FluxDB achieves 78K+ triples/second with batch operations.

Secure Your Deployment

Bind REST API to localhost by default. Use nginx for HTTPS, authentication, and rate limiting in production.

Monitor Database Size

Use odb_stat regularly to track triple count, database size, and index statistics. Plan storage accordingly.

Training & Support

FluxDB includes comprehensive documentation, example code, and quick start guides. Professional support, training, and consulting services are available. Contact sales@FluxDB.ai for more information.

Contact Information

Get in Touch

Ready to transform your legacy systems with agentic AI and ontological knowledge graphs? Our team is here to help you get started.

Sales & Inquiries:

sales@FluxDB.ai

Website:

www.FluxDB.ai

How We Can Help

Proof of Concept

Start with a focused POC to validate FluxDB for your use case. We'll help you design your ontology, load sample data, and demonstrate performance on your queries.

Implementation Services

Our experts can design and build your production knowledge graph, from ontology design to data migration to deployment and optimization.

Training & Workshops

Comprehensive training for your team on RDF, SPARQL, ontology design, and FluxDB best practices. Both virtual and on-site options available.

Enterprise Support

Production support with SLAs, priority bug fixes, performance optimization, and direct access to FluxDB engineering team.

Typical Engagement Process

1. **Discovery Call:** Discuss your use case, requirements, and technical environment. (1 hour)
2. **Technical Assessment:** Review your data sources, ontology requirements, and integration points. (1-2 weeks)
3. **Proof of Concept:** Build working prototype with sample data and representative queries. (2-4 weeks)
4. **Production Deployment:** Full-scale implementation with production data and infrastructure. (4-12 weeks)
5. **Ongoing Support:** Training, optimization, and maintenance as needed.

Why Choose FluxDB?

- ✓ **Production Proven:** v2.0.0 is stable, tested, and ready for enterprise deployment
- ✓ **Performance Leader:** Sub-millisecond queries with 78K+ triples/sec bulk loading
- ✓ **Purpose-Built for AI Agents:** Designed from the ground up for agentic harnesses
- ✓ **Open Standards:** Full RDF, SPARQL 1.1, and W3C compliance
- ✓ **Comprehensive APIs:** C, REST, SPARQL, and Python SDKs included
- ✓ **Expert Team:** Knowledge graph specialists with decades of experience

✓ **Flexible Licensing:** Options for evaluation, development, and production use

Start Your Journey Today

Transform your legacy modernization, brownfield development, and agentic AI initiatives with FluxDB's ontological knowledge base. Contact us at **sales@FluxDB.ai** to schedule a discovery call.

⚡ FluxDB

Ultra-Fast Ontological Knowledge Base for Agentic AI Systems

FluxDB is a production-ready knowledge graph database engineered for enterprise applications.

Version 2.0.0 | Production Ready