Project 6

DisasterContextGraph-RAG

Knowledge-Aware Context Graph Architecture for Disaster Intelligence

UrbanResilience Lab | Texas A&M University

1. Executive Summary

This project develops a **Context Graph architecture** for disaster intelligence systems, creating a unified knowledge layer that bridges structured disaster data (TDDL Silver) with unstructured knowledge resources (RAG corpus). The Context Graph serves as a "shared memory" enabling semantic query resolution, schema-aware reasoning, and consistent world modeling across AI agents.

The architecture addresses critical challenges: (1) disambiguating natural language queries against complex geospatial schemas, (2) binding regulatory and planning documents to operational data features, and (3) enforcing correct aggregation semantics for H3-indexed disaster metrics.

1.1 Research Questions

1. **RQ1:** How can a context graph architecture enable semantic disambiguation of natural language queries against disaster-specific data schemas?

2. **RQ2:** What graph topology and edge semantics optimally bind structured geospatial features to unstructured policy/planning documents?

3. **RQ3:** How can aggregation rules and temporal semantics be encoded in a graph structure to ensure query correctness?

4. **RQ4:** What are the performance and governance trade-offs between lakehouse-native and property graph implementations?

1.2 Problem Statement

The system operates across two fundamentally different knowledge planes:

- **TDDL Silver (Structured):** Feature-level disaster metrics indexed by H3 Level 10 hexagons with strict temporal validity, dynamic vs. static classifications, and area-weighted administrative rollups

- **Knowledge Repository (Unstructured):** Narrative policies, emergency plans, After-Action Reports (AARs), and scholarly content requiring retrieval-augmented generation

Without a unifying knowledge structure, the system risks: schema hallucination, incorrect feature family selection, unit and temporal confusion, and failure to apply required area-weighting rules.

2. Technical Architecture

The Context Graph implements a **two-layer design**: Schema Graph (what exists and how to use it) + Evidence Graph (what is true/available now).

2.1 Schema Graph (Layer 1)

Describes what exists in the system and how to use it correctly. This layer is mostly static and provides the semantic foundation for query construction.

Node Types

Node Type	Description & Properties
Feature	From feature_catalog_silver: feature_id, feature_name, category, hazard_group, is_dynamic, units, description, update_frequency
HazardGroup	Canonical hazard categories: Flood, Wind, Fire, etc.
Unit	Measurement units: index_0_100, fraction_0_1, mph, inches, km, boolean_gate
QuerySpec	Query library entries: id, title, tier, combination type
AggregationRule	Rollup recipes: area-weighted average, additive totals, weighted share, gate masking
Synonym/Concept	Canonical user phrases: "high flood risk", "floodplain", "last 24 hours"

Edge Types

- Feature → BELONGS_TO → HazardGroup: Maps features to hazard category
- Feature → MEASURED_IN → Unit: Specifies measurement unit
- Feature → HAS_TIME_SEMANTICS → Dynamic|Static: Indicates temporal behavior
- Feature → REQUIRES_AGG_RULE → AggregationRule: Mandates correct rollup method
- **QuerySpec → USES_FEATURE → Feature:** Links query specs to required features
- **Concept → MAPS_TO_FEATURE → Feature:** Resolves natural language to feature_ids

2.2 Evidence Graph (Layer 2)

Tracks what is true/available now and what evidence supports it. This layer is **dynamic** and frequently updated.

Node Types

Node Type	Description
Document	Policies, plans, AARs with metadata (hazard tags, jurisdictions, dates)
Chunk	RAG chunks with embedding references
AdminArea	Geoid nodes from dim_admin (county, tract, etc.)
TimeWindow	Normalized time ranges used in queries
Run/Artifact	Query runs with SQL hash, execution time, snapshot time

Edge Types

- **Document → ABOUT → HazardGroup:** Tags documents to relevant hazards
- Document → APPLIES_TO → AdminArea: Scopes documents to jurisdictions
- **Chunk → MENTIONS_FEATURE → Feature:** NER/ontology extraction linking text to features
- **Feature → AVAILABLE_AS_OF → TimeWindow:** Freshness/coverage marker from ingest_time
- Run → USED_FEATURE → Feature: Query lineage tracking

2.3 Integration Points

The Context Graph integrates with four primary system components:

- **NL-to-SQL Agent:** Feature selection, QuerySpec matching, aggregation enforcement
- **RAG Agent:** Structured filters based on AdminArea/HazardGroup, Chunk-Feature connections
- **Orchestrator:** Routing decisions (SQL vs RAG vs blended), plan generation, fallback logic
- **Visualization Layer:** Query spec output types, choropleth mappings, legend policies, LOD rules

3. Deliverables

Component	Specification
Schema Graph	Feature nodes from feature_catalog_silver, Concept/Synonym nodes (50-100 phrases), QuerySpec nodes, AggregationRule nodes.
Evidence Graph	Document and Chunk nodes, Document → HazardGroup and Document → AdminArea edges, feature freshness markers, query lineage nodes.
Feature Resolver	Service API that maps natural language phrases to candidate feature_ids with recommended semantics.
Query Router	Service that matches user intents to existing query specifications in the library.
Learned Edges	Feature-Feature similarity edges (co-usage patterns), Document-Feature alignment edges (embedding similarity).
Benchmark Dataset	500+ NL queries spanning ambiguous hazard references, temporal disambiguation, administrative scale variations.

4. Evaluation Framework

4.1 Performance Metrics

Metric	Definition	Target
Disambiguation Accuracy	Correct feature_id selection given ambiguous input	**>90%**
Routing Precision	Correct SQL/RAG/blended routing decision	**>95%**
Query Spec Match Rate	Existing spec selected when applicable	**>85%**
Aggregation Correctness	Correct rollup rule applied	**100%**
Latency Overhead	Added time for graph traversal	**<100ms p99**

4.2 Benchmark Dataset

500+ natural language queries spanning:

- Ambiguous hazard references (regulatory vs. predisposition vs. dynamic)
- Temporal disambiguation (historical vs. current vs. forecast)
- Administrative scale variations (hex-level vs. county vs. region)
- Multi-hazard and compound queries

5. Implementation Timeline

Total Duration: ~~~8 months (34 weeks)~~ By the end of the semester

Phase 1: MVP Development (Weeks 1-8)

High Leverage, Low Effort

- Schema Graph implementation with Feature nodes from feature_catalog_silver
- Concept/Synonym nodes for common user phrases (50-100 initial vocabulary)
- QuerySpec nodes for existing query library entries
- AggregationRule nodes (area-weight avg, additive total, weighted share, gate mask)
- Feature Resolver service API and Query Router service API

Phase 2: Evidence Binding (Weeks 9-18)

- Evidence Graph layer with Document and Chunk nodes
- Document → HazardGroup and Document → AdminArea edge population
- Feature freshness markers based on ingest_time/latest availability
- Query run lineage nodes for explainability
- NER pipeline for extracting feature mentions from document chunks

Phase 3: Learned Edges (Weeks 19-30)

- Feature-Feature similarity edges based on co-usage patterns and correlation analysis
- AdminArea-AdminArea similarity using Resilience Footprints (if available)
- Document-Feature alignment edges based on embedding similarity
- Hybrid implementation: Delta tables for canonical edges + Vector Search for semantic lookups

Phase 4: Evaluation & Documentation (Weeks 31-34)

- Benchmark construction and performance testing
- Documentation and governance patterns
- Manuscript preparation and submission

6. Target Venues & Risk Assessment

6.1 Publication Targets

1. **Tier 1 Venue:** Architecture paper targeting ACL, EMNLP, or AAAI
2. **Domain Venue:** Application paper targeting Natural Hazards and Earth System Sciences or IJDRR
3. **Benchmark Paper:** Context Graph benchmark dataset for disaster NL-to-SQL

6.2 Expected Contributions

- **Technical:** Novel two-layer context graph architecture for disaster intelligence systems
- **Algorithmic:** Disambiguation algorithms for geospatial disaster queries
- **Integration:** Patterns for hybrid structured-unstructured disaster knowledge systems
- **Operational:** Reduced query errors, improved RAG precision, consistent aggregation semantics

6.3 Risk Assessment

Risk	Impact	Mitigation Strategy
Concept vocabulary gaps	High	Iterative expansion based on production query analysis
Graph traversal latency	Medium	Caching, materialized views, query optimization
Schema drift in TDDL	Medium	Automated sync pipeline, schema change detection
NER extraction quality	Medium	Human-in-loop validation, confidence thresholds

6.4 Implementation Approach

Recommended: Lakehouse-Native Implementation (aligned with Databricks ecosystem)

- Unity Catalog governance and lineage
- Easy Orchestrator integration with SQL traversals
- Delta tables: context_nodes, context_edges, concept_embeddings
- Hybrid option: Delta for canonical edges + Vector Search for semantic lookups

— End of Document —