🧠 NoSQL Ecosystem Overview

Generative AI Bootcamp – Week 2 (Databases & Data Pipelines)

Day 4 – Thursday, November 27, 2025

Why NoSQL for AI Systems?

- Traditional SQL databases = structured data, rigid schema
- **AI data** is often:
  - Unstructured (text, JSON, embeddings)
  - Variable schema (dynamic metadata)
  - High volume & real-time
- NoSQL = flexibility, scalability, and distributed design

The 4 Major NoSQL Categories

Type	Structure	Example Use	Common Tools
Document	JSON-like	Store chat logs, prompts, model outputs	MongoDB, Firebase
Key-Value	Dictionary-style	Cache embeddings, session data	Redis, DynamoDB
Columnar	Wide tables	Analytical pipelines, feature stores	Cassandra, HBase
Graph	Nodes + edges	Knowledge graphs, entity linking	Neo4j, ArangoDB

Document Databases

- Store JSON or BSON documents
- Flexible schemas → perfect for **dynamic AI data**
- Queries by document structure or content
- Often used for **prompt/response logs**, **user histories**, **metadata**

Example: MongoDB

```
{
  "user": "alex",
  "prompt": "Summarize this document",
  "response": "This paper explores...",
  "timestamp": "2025-11-27T09:35Z"
}
```

Key-Value Stores

- Simplest NoSQL form – a dictionary on steroids
- Perfect for **caching embeddings**, **tokens**, or **session states**
- Extremely fast reads/writes

Example: Redis

```
SET session:42 '{"context":"RAG retrieval","tokens":512}'
```

Columnar Databases

- Store data by **column** instead of by row
- Ideal for analytical workloads with high dimensionality
- Used for **feature storage**, **analytics**, and **batch queries**

Example: Cassandra

```
CREATE TABLE embeddings (
  doc_id UUID PRIMARY KEY,
  vector list<float>,
  created_at timestamp
);
```

Graph Databases

- Represent **entities and relationships**
- Great for **knowledge graphs**, **semantic reasoning**, and **RAG enrichment**
- Query language: **Cypher** (Neo4j) or **Gremlin**

Example (Cypher):

```
CREATE (p:Paper {title:'AI Ethics'})
CREATE (a:Author {name:'Ada'})
CREATE (a)-[:WROTE]->(p);
```

NoSQL vs SQL: Quick Comparison

Feature	SQL	NoSQL
Schema	Fixed	Flexible
Scaling	Vertical (harder to move beyond single server)	Horizontal (distributed by design)
Transactions	Strong ACID	Eventual consistency
Query Language	SQL	Various (JSON, key-based, graph)
Ideal For	Structured business data	AI, logs, embeddings

NoSQL in LLM Pipelines

- Store **unstructured datasets** (documents, prompts, embeddings)
- Enable **semantic search + retrieval** via metadata filters
- Power **RAG** and **memory components** for LLMs
- Integrate with vector stores (e.g. FAISS + MongoDB hybrid)

Choosing the Right Store

Scenario	Recommended Type
Caching embeddings	Key-Value (Redis)
Storing chat histories	Document (MongoDB)
Storing embeddings & metadata	Document or Vector
Knowledge graph for reasoning	Graph (Neo4j)
Large-scale analytical logs	Columnar (Cassandra)

Summary

- NoSQL offers flexibility for **AI-driven** data diversity
- Know your data type → choose the right model
- Forms foundation for **vector stores**, **graph reasoning**, and **multi-modal retrieval**

Next: MongoDB / Firebase Lab 🧪