🧠 Introduction to Databases in AI Systems

Generative AI Bootcamp – Week 2, Day 1, Session 1

November 24, 2025

🎯 Learning Objectives

- Understand the role of databases in AI systems
- Differentiate relational vs. non-relational databases
- Identify where data persistence fits in LLM workflows
- Recognize performance and scalability considerations

🧩 Databases in the AI Stack

- **Data layer** underpins all AI workflows
- Handles structured and unstructured data
- Supports:
    - Training data storage
    - Retrieval-Augmented Generation (RAG)
    - Logging & evaluation traces

🗂️ Data Modalities

Type	Examples	Usage in AI
Structured	SQL, BigQuery	Metadata, logs, analytics
Semi-structured	JSON, CSV	Prompts, configs
Unstructured	Text, images, embeddings	Retrieval and fine-tuning

🧱 Relational Databases (SQL)

- Tabular, schema-defined
- ACID (Atomic, Consistent, Isolated, Durable) transactions

> Consistency and reliability are of utmost importance

- Examples: PostgreSQL, MySQL, BigQuery
- Strengths: Consistency, analytics, joins
- Limitations: Schema rigidity, scaling writes

📜 SQL is Also a *Language*

- *Structured Query Language*
- Declarative DSL (essentially):
  - `SELECT`, `FROM`, `WHERE`, `LIMIT`, `JOIN`, `GROUP BY`, `ORDER BY`...
- Procedural elements:
  - `CREATE`, `INSERT`, `UPDATE`, `DELETE`...

⚙ Non-Relational Databases (NoSQL)

- Flexible schema, document or key-value models
- BASE (Basically Available, Soft State, Eventual Consistency) properties

> Immediate consistency is less critical than continuous service and horizontal scalability

- Examples: MongoDB, Firebase, DynamoDB
- Used in AI for storing documents, chat histories, and embeddings

🧮 Vector Databases (for LLMs)

- Based on **embeddings** (vectorial representations of data)
- Fast nearest-neighbor queries (cosine similarity / dot-product metrics)

> Efficiently navigating semantic spaces (highly unstructured) is what matters

- Examples: FAISS, Chroma, Weaviate, Pinecone
- Foundation for Retrieval-Augmented Generation (RAG)

🌐 Databases in Generative AI Workflows

1. **Input management:** store prompts and context
2. **Artifact storage:** save preprocessed context for AI workflows
3. **Retrieval layer:** vector search for relevant context
4. **Output logging:** capture LLM responses
5. **Feedback & evaluation:** store quality metrics

⚖️ Choosing the Right Database

Use Case	Recommended DB
Structured analytics	BigQuery, PostgreSQL
Logs, metadata	MongoDB, Firebase
Vector search	FAISS, Pinecone
Graph relationships	Neo4j

☁️ Cloud Databases Overview

- **BigQuery (Google Cloud)**: scalable analytics engine
- **Watson Query (IBM)**: federated data access
- Integration via Python SDKs (`google-cloud-bigquery`, `ibm-watsonx`)

🧰 Python Integration Patterns

- ORM (SQLAlchemy, Tortoise ORM)
- Direct SDK access (BigQuery client)
- REST/GraphQL APIs
- Embedding storage via FAISS/Chroma Python APIs

🚧 Common Challenges

- Latency and query optimization
- Schema evolution
- Access control and security
- Handling large-scale vector data

💡 Summary

- Databases form the foundation of every AI system
- SQL handles structured data; NoSQL handles flexibility
- Vector databases enable semantic retrieval
- Next: Hands-on SQL basics in the next session!