Lecture 21: Vector Database Introduction

Why Do We Need Vector Databases?

- We already know how embeddings help compare words or text.
- Searching with normal SQL is limited: it matches only exact words.
- Example: Searching “wearable” might miss results like “smartwatch” or “T-shirt.”
- What we want instead is **semantic search**—results based on meaning, not just keywords.

Example Scenario

- Dataset: product_details.json (with product title, description, price, category, and features).
- SQL search for "wearable" → returns nothing if the word is missing in text.
- But semantically, products like **T-shirts** or **smartwatches** are wearable.
  This gap is solved by embeddings and vector databases.

How Does It Work?

- **Convert user query into embeddings**
  - Input text like "wearable" → embedding vector.
- **Convert product data into embeddings**
  - Large documents are broken into **chunks**.
  - Each chunk is an embedding vector.
- **Store embeddings in a Vector Database**.
- **Perform similarity search**
  - Matches query embeddings with product embeddings.
  - Retrieves results even if words differ, but meaning is close.

VectorStore in Spring AI

- Spring AI provides an interface called **VectorStore**.
- Operations you can do:
  - Add documents (single or list).
  - Delete documents.
  - Run a similarity search.

Available Implementations

- Supports many backends: **Azure, Cassandra, Chroma, Elasticsearch, MariaDB, Neo4j, Oracle, PgVector, Redis, and SAP**.
- For learning/demo → use **SimpleVectorStore** (in-memory, not for production).
- For real projects → use **PgVector** (PostgreSQL extension).

Key Takeaway:

- Vector databases let us search based on **meaning**, not just keywords. Spring AI makes it simple with **VectorStore.** Start with **SimpleVectorStore** for practice, then move to PgVector or others for production.