Lecture 26: PG Vector Store Implementation

Why Changes Are Needed

- Until now, we used **SimpleVectorStore** (in-memory).
- To use a **real database** for embeddings, we switch to **PGVectorStore** (works with PostgreSQL).
- This requires **extra dependencies** and **application property changes**.

Dependencies to Add

```xml
<dependency>
   <groupId>org.springframework.ai</groupId>
   <artifactId>spring-ai-starter-vector-store-pgvector</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-docker-compose</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
```

Key Code Change

- Replace **SimpleVectorStore** with **PGVectorStore**.
- PGVector needs:
    - **EmbeddingModel** (e.g., OpenAI embeddings)
    - **JdbcTemplate** (bridge between Java & Postgres)

```java
@Configuration
public class AppConfig {

    @Bean
    public VectorStore vectorStore(JdbcTemplate jdbcTemplate, EmbeddingModel embeddingModel) {
        return PgVectorStore.builder(jdbcTemplate, embeddingModel)
                .dimensions(1536)                    // embedding dimensions
                .distanceType(COSINE_DISTANCE)       // similarity metric
                .indexType(HNSW)                     // index type
                .build();
    }
}
```

Application Properties Setup:

- allow-bean-definition-overriding → avoids bean conflicts
- schema-locations → points to schema file (schema.sql)
- init.mode=always → ensures schema loads on startup

```
spring.main.allow-bean-definition-overriding=true

spring.sql.init.schema-locations=classpath:init/schema.sql
spring.sql.init.mode=always
```

Key Takeaways:

- PGVectorStore lets you **persist vectors in PostgreSQL** (instead of memory).
- Needs:
    - Docker container with Postgres + PGVector
    - Correct schema setup
    - Proper dimensions matching embedding model
- Once configured, semantic search works the same as before—but data is stored **persistently** in Postgres.