Integrating Vector, Relational, and JSON Data

1. Overview of AI Workflows	2. Why Postgres as a vector store	3. Storing and managing vectors	4. Querying the vector store
• Look at high-level architecture - LLMs, vector stores and JSON • Look at key vocabulary and concepts (embeddings, vectors, hybrid queries, etc.)	• What is a vector store? Key concepts and use cases. • Why Postgres and how does it compare with other market tools • Setting up Postgres with vector capabilities (pgvector) • **Lab**: Install and configure Postgres using Docker	• Generating embeddings: Overview of tools and workflows • Storing and organizing embeddings in Postgres • Strategies for handling large datasets including chunking • Dense and sparse vectors • **Lab:** Generate embeddings for a dataset and store them	• Techniques for similarity search: k-NN, cosine similarity • Using indexes to optimize vector queries • Reranking results • **Lab:** Query stored vectors to retrieve similar items (document/image search)
5. Querying LLMs with retrieved data	**6. NoSQL with JSON in Postgres**	**7. Integrating Vector, Relational and JSON Data**	**8. Putting it all together**
• Recap on querying LLMs vis APIs • Best practices for combining vector retrieval with LLM prompts • Prompt configuration parameters (temperature, top-k, etc) • **Lab:** Build a pipeline where vector store results enhance LLM responses (context-aware Q&A, etc)	• Overview of JSON/JSONB support in Postgres • Querying JSONB data with SQL • Indexing JSONB data for performance • **Lab:** Design a schema mixing vector, relational and JSONB data for a sample project	• Building hybrid queries to power advanced workflows • **Case study:** Combining embeddings, metadata (relational) and configurations (JSON) • **Lab:** Implement a hybrid query to support a sample AI use case	• Full stack pipeline demo: Retrieve data, query the LLM and return results • Debugging and optimising the workflow • Spotlight on LLM frameworks • **Lab:** Build a working application combining all elements

Plan for the session

- Understand the Benefits of Hybrid Queries
- Write SQL Queries that Combine Vector, Relational, and JSONB Data

Leveraging Multiple Data Types for AI-Powered Workflows

📌 **Key Points:**

- AI applications require **multiple types of data**:
  - **Relational (structured metadata like categories, timestamps, user data).**
  - **Vector embeddings (semantic meaning and similarity).**
  - **JSONB (flexible, semi-structured configurations and additional attributes).**
- Hybrid queries allow **rich, context-aware data retrieval**.

📌 **Key Takeaway:**
Hybrid queries improve AI workflows by combining structured, unstructured, and semantic search capabilities.

Breaking Down the Query Stack

🔹 1️⃣ **Relational Data (PostgreSQL Table Columns)**

- Stores **structured fields** (e.g., id, name, created_at).
- Enables **fast lookups and filtering** using indexes.

🔹 2️⃣ **Vector Data (pgvector for Embeddings)**

- Stores **semantic representations** of text/images.
- Enables **nearest neighbor search** for similarity matching.

🔹 3️⃣ **JSONB Data (Flexible Attributes & Configurations)**

- Stores **semi-structured data** (e.g., {"price": 29.99, "stock": 100}).
- Allows **flexible querying and updates** without schema changes.

📌 **Key Takeaway:**
Each data type serves a different purpose, and combining them makes AI retrieval more powerful.

Query Flow: Retrieving Context-Rich Data

1 User Query → Generate Vector Embedding

- Convert text query into an embedding using bge-m3.

2 Vector Similarity Search → Retrieve Relevant Entries

- Use pgvector to find the most similar stored items.

3 Relational Filtering → Narrow Down Results

- Apply filters (e.g., WHERE category = 'programming').

4 JSONB Extraction → Enrich Data with Additional Fields

- Fetch relevant metadata (e.g., price, configuration, availability).

Demo

Real-World Scenarios That Benefit from Hybrid Data

✅ **Semantic Search + Metadata Filtering**

- Find **similar research papers** but filter only by **peer-reviewed publications**.

✅ **Personalized Recommendations**

- Retrieve **movies similar to "Inception"**, then filter by **user preferences (JSONB settings)**.

✅ **AI-Powered Knowledge Retrieval**

- Find **similar troubleshooting issues**, but **prioritize ones from recent cases**.

Optimizing Multi-Modal Queries in PostgreSQL

✅ **Use Indexing for Performance**

- **GIN indexes** for JSONB
- **IVFFLAT or HNSW indexes** for pgvector

✅ **Limit Query Scope for Efficiency**

- Use LIMIT, WHERE, and ORDER BY to **reduce search space**.

✅ **Normalize Data When Necessary**

- Store **high-frequency relational fields** separately to **avoid JSONB bloating**.

✅ **Cache Query Results for AI Pipelines**

- Precompute and store results for **frequent queries** to improve response times.

Lab

Apply Hybrid Querying in a Real-World AI Use Case *(Hands-on Lab)*
•Design a **real-world query** that fetches relevant **vector-based** results, filters them using **relational metadata**, and **enriches responses with JSONB fields**.
•Implement a **case study** demonstrating **multi-source AI retrieval** (e.g., **retrieving books, filtering by metadata, and adjusting configurations dynamically**).


Questions
?