Vector Embedding & Search in AlloyDB

Pushkar Khadilkar & Vaibhav Jain

February 2024

Agenda

01 Introduction

02 Vector Embeddings

03 AlloyDB AI

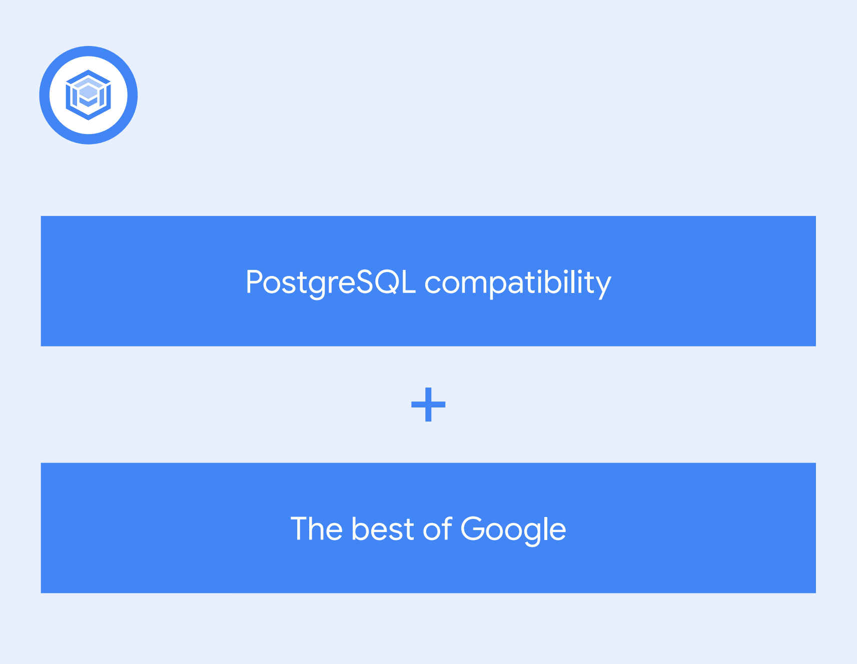
04 Resources

01

Introduction

Introducing AlloyDB

A new open-source compatible database engine ready for top-tier relational database workloads



PostgreSQL compatibility

+

The best of Google

AlloyDB is the highest performance database for gen AI apps

Industry leading multi-workload performance:

4x

faster transactional queries*

10x

faster vector queries**

100x

faster analytical queries*

*Compared to PostgreSQL | **Compared to PostgreSQL when using IVFFlat index

Commercial-grade, without the costs or vendor lock-in

Highly available

99.99% SLA, inclusive of maintenance

Automatic and fast failure recovery

Non-disruptive management operations

Highly scalable

Scale-out storage

Horizontal read scalability 1000+ vCPUs

Vertically scalable writes

Intelligent

Autopilot capabilities and embedded AI/ML make management easy

Integrated with Vertex AI

Performant

4x faster for transactional workloads

Up to 100x faster for analytical queries

Fully PostgreSQL-compatible

Predictable, transparent pricing

AlloyDB Omni

Run AlloyDB anywhere - in your datacenter, your laptop, and in any cloud

Runs anywhere

- Packaged in a downloadable container
- Runs on-premises and in most public clouds; developers can run it on their laptops

Highly scalable

- Scales to much larger number of CPUs than standard PostgreSQL
- Delivers more than 2x OLTP throughput compared with standard PostgreSQL

Intelligent

- Automatic vacuum management
- Automatic memory management
- Automatic columnarization
- Integration with Google Cloud Vertex AI Generative AI models

Performant

In-memory columnar delivers 100X faster analytics queries compared with standard PostgreSQL

Fully PostgreSQL-compatible

$ Predictable, transparent, pricing at a fraction of the cost of legacy databases

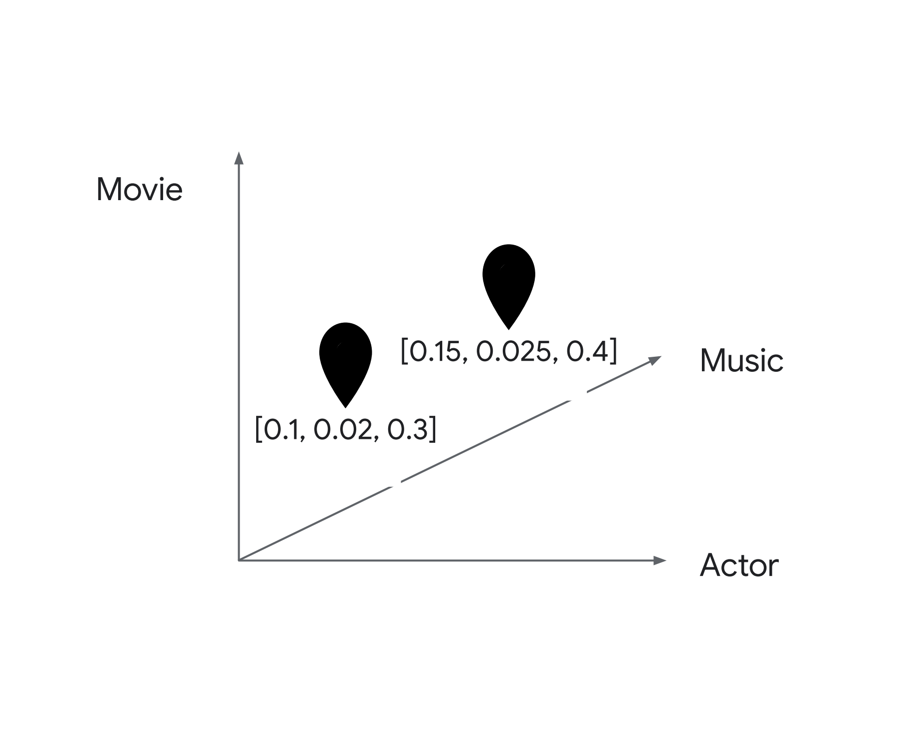
02

Vector Embeddings

Vector Embeddings

A vector is a mathematical object that has both **magnitude** and **direction**

A vector embedding is a specific type of vector that is used to **represent** any kind of data, such as numbers, text, or images



Getting value out of unstructured data with embeddings

Data

Images, videos, text, songs, time-series, etc.

Deep learning model

Embedding Model

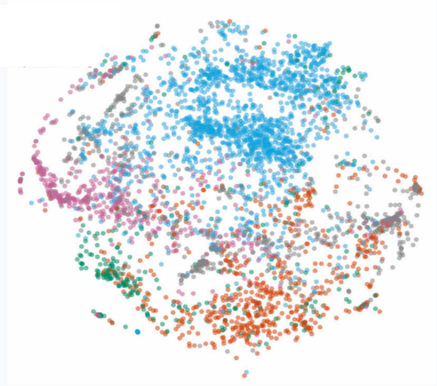
Pre-trained custom encoders

Embedding (vector) representations

[0.2, 0.5, 1.2, ..., 0.4, 0.05, 0.6]

Vectors of numbers representing the semantic structure of an entity

Serving index



Similar objects clustered together

Large Language Models (LLM)

- Trained on vast amounts of publically available data.
- Phenomenal for text generation, Q&A, reasoning.
- Rely on the information they were trained on, guided by the prompt.
- Problem: Don’t have access to the business proprietary data or real time information.
- Solution**: R**etrieval-**A**ugmented-**G**eneration
    - Augment the relevant context in real-time by an external knowledge source.

Databases bridge the gap between LLMs and enterprise Gen AI apps

Databases:

1. Provide the most **up-to-date** data
2. Can efficiently store and search **vector** embeddings
3. Are your **trusted** and familiar data store

03

AlloyDB AI

AlloyDB AI

AlloyDB AI

An integrated collection of capabilities for easily building generative AI enterprise applications with PostgreSQL

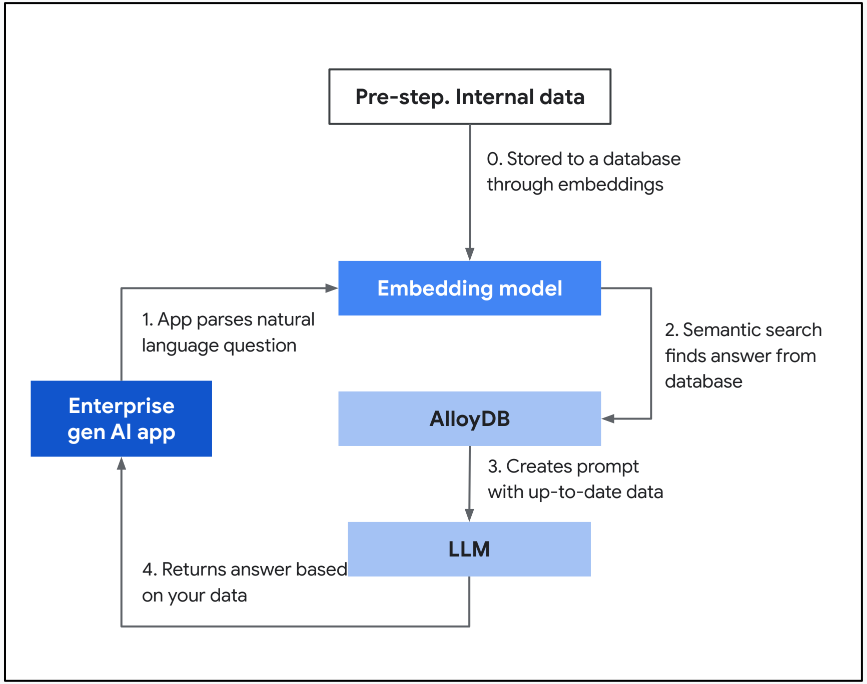
How it works

01 Automatically **generate embeddings** on your operational data **using SQL**, with easy access to Google's embeddings models

02 Store, index and query vector embeddings, turning your AlloyDB database into a **vector database** with upto **10x faster** vector similarity search

03 **Integrate AlloyDB into your GenAI applications** with Vertex AI and open source frameworks like LangChain

How Google databases and LLMs enable enterprise gen AI apps


Pre-step. Internal data
0. Stored to a database through embeddings
Embedding model
1. App parses natural language question
2. Semantic search finds answer from database
Enterprise gen AI app
AlloyDB
3. Creates prompt with up-to-date data
LLM
4. Returns answer based on your data

Pre-step: Your internal data is stored in a database through the embedding model.

1. Gen AI app uses the embedding model to convert natural language question ("What's your return policy?") to vectors.
2. Embedding model is used for semantic search on the database to retrieve the current return policy ("60 days").
3. Database returns the up-to-date policy to be used as part of the prompt for the LLM.
4. LLM constructs an accurate answers based on your data ("Our return policy is 60 days").

Vector Embedding Generation

Vertex AI Integration allows accessing predictions.

The functionality is available through google_ml_integration extension.

1. `embedding`: Text embedding for the given input.
2. `ml_predict_row`: Generic ML function inference with JSON input / output.

```
select
ml_predict_row('projects/PROJECT_ID/locations/us-central1/publishers/google/models/text-bison', '{"instances":[{"prompt": "What are three
advantages of using AlloyDB as the database server?"}],
"parameters":{"maxOutputTokens":1024, "topK": 40, "topP":0.8,
"temperature":0.2}}');
```

```
SELECT embedding(
    model_id => 'textembedding-gecko@001',
    content => consumer_complaint_narrative)
FROM consumer_details;
```

Vector Embedding Storage & Search

Supports pgvector extension for vector storage and search.

1. Use vector data type for columns, functions.
2. Generate embeddings using embedding function.
3. Index types hnsw, ivfflat & ivf (with SQ8 quantization) available in AlloyDB for ANN search.
4. Deeper integration with query engine allows upto 10x faster queries

```
CREATE EXTENSION IF NOT EXISTS vector;

ALTER TABLE furniture ADD COLUMN description_embeddings
vector(768) GENERATED ALWAYS AS
(embedding('textembedding-gecko@001',
description)::vector) STORED;

CREATE INDEX ON furniture
  USING ivfflat (description_embeddings vector_cosine_ops)
  WITH (lists = 20);
```

Ivf index

Usage of scalar quantization technique

1. Works with pgvector's vector data type
2. Uses scalar quantization technique
   a. Converts floating points into integers
   b. Optimizes storage
   c. Improves performance (with some recall loss)
      i. Original: [0.3411, 0.2113, 0.453322,...] - 4 bytes
      ii. Output: [12, 23, 15] - 1 byte
3. Supports indexing upto 8k dimension vector

```
CREATE INDEX ON furniture
  USING ivf (description_embeddings vector_cosine_ops)
  WITH (lists = 20, quantizer = 'SQ8');
```

04

Resources

Resources

- AlloyDB (https://cloud.google.com/alloydb)
- AlloyDB Omni (https://cloud.google.com/alloydb/omni)
- AlloyDB AI (https://cloud.google.com/alloydb/ai)
- Codelab: Getting Started with Vector Embeddings for AlloyDB AI (https://codelabs.developers.google.com/codelabs/alloydb-ai-embedding)
- Demo: Build AI-powered apps on Google Cloud with pgvector, LangChain & LLMs (https://www.youtube.com/watch?v=Jl1S4ZcSY8k)

Questions ?