EDB
Postgres for the AI Generation

# Bringing Vectors to Postgres with pgvector

Gülçin Yıldırım Jelinek
Staff Database Engineer
27 June 2024, Swiss PGDay

EDB
Postgres for the AI Generation

# Gülçin Yıldırım Jelinek
Staff Engineer, EDB

- Host @ The Builders: A Postgres Podcast
- Co-founder @ Prague PostgreSQL Meetup
- Previously on Board of Directors @ PostgreSQL Europe

X: @apatheticmagpie @postgrespodcast, @PrahaPostgreSQL

# AGENDA

- What is pgvector?
- What is vector search and why is it used?
- Generating and querying embeddings
- New index types: IVFFlat and HNSW
- Future of vectors, AI and Postgres


pgvector

# Language Support

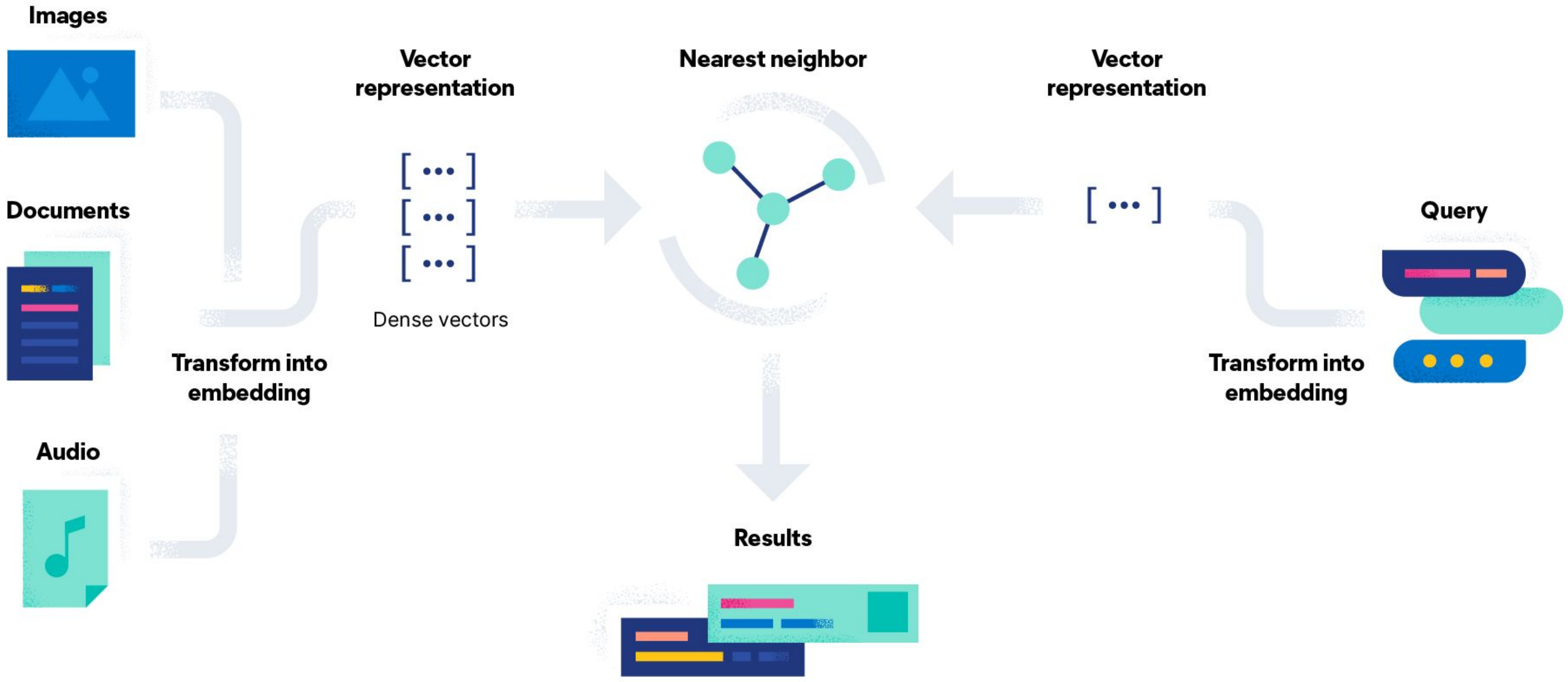
- **Go:** pgvector-go
- **Python:** pgvector-python
- **Rust:** pgvector-rust
- **C:** pgvector-c
- **JavaScript, TypeScript:** pgvector-node
- **PHP:** pgvector-php

# What is vector (similarity) search?

**Vector similarity search** is a technique used to find **the most similar vectors** to a **given vector** (usually a **query vector**).

This query is typically performed by calculating distances in vector space, and various **metrics** (such as **Euclidean distance, cosine similarity**) can be used to measure the similarity between the query vector and other vectors.

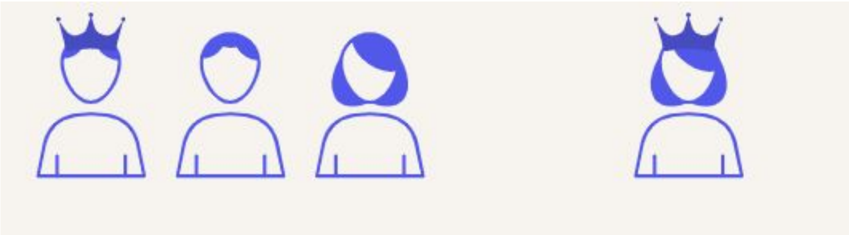
# What is vector (similarity) search?



# What is vector (similarity) search?

queen

king - man + woman



warsaw

paris - france + poland

# What is vector search useful for?

## AI applications: working with high-dimensional data

- Recommendation engines
- Image search
- Natural language processing (NLP)
- Content-based filtering
- Similarity-based AI tasks
- Prediction solutions

# What is vector?

$X = [1, 3, 5]$

# What is vector?

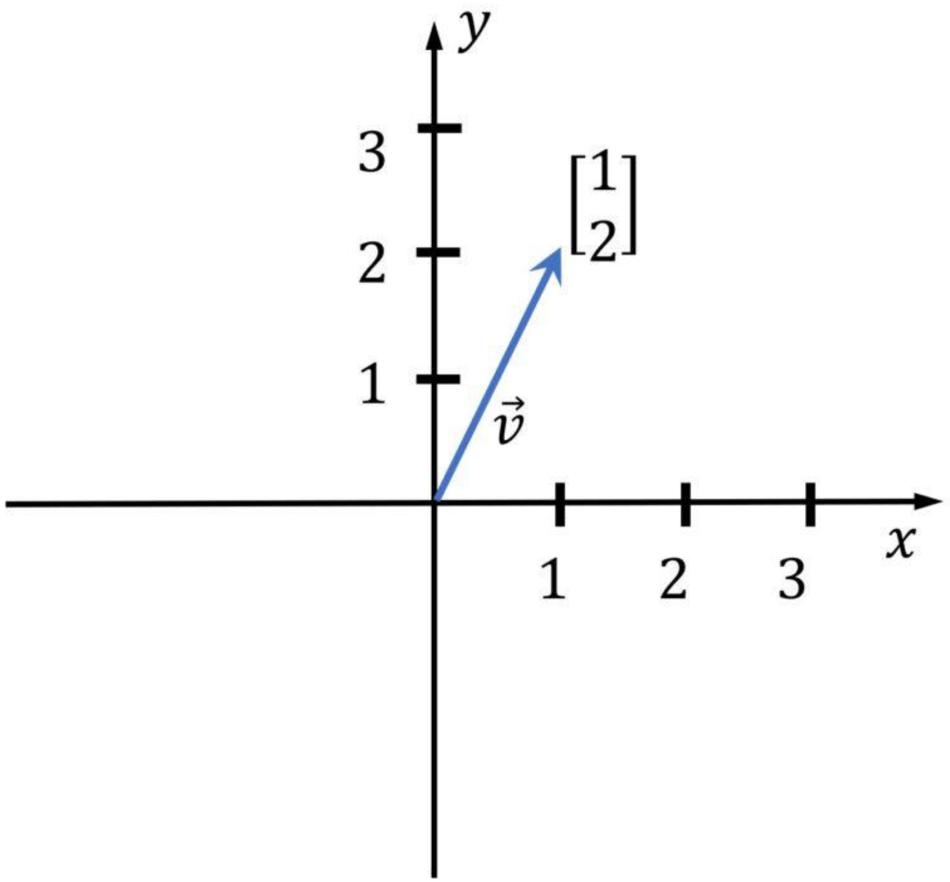



*Image source: https://media5.datahacker.rs/2020/03/Picture36-1-768x712.jpg*

# Vector Data Type

- Each vector takes **4 * dimensions + 8 bytes** of storage
- Vectors can have up to **16,000 dimensions.**

- Vector operators:
  - **<->** Euclidean distance
  - **<#>** negative inner product
  - **<=>** cosine distance
  - **+** element-wise addition
  - **-** element-wise subtraction
  - ***** element-wise multiplication

- Vector functions:
  - cosine_distance
  - inner_product
  - l2_distance (Euclidean distance)
  - l1_distance
  - vector_dims (number of dimensions)

# Sample app code

https://github.com/gulcin/pgvector_blog

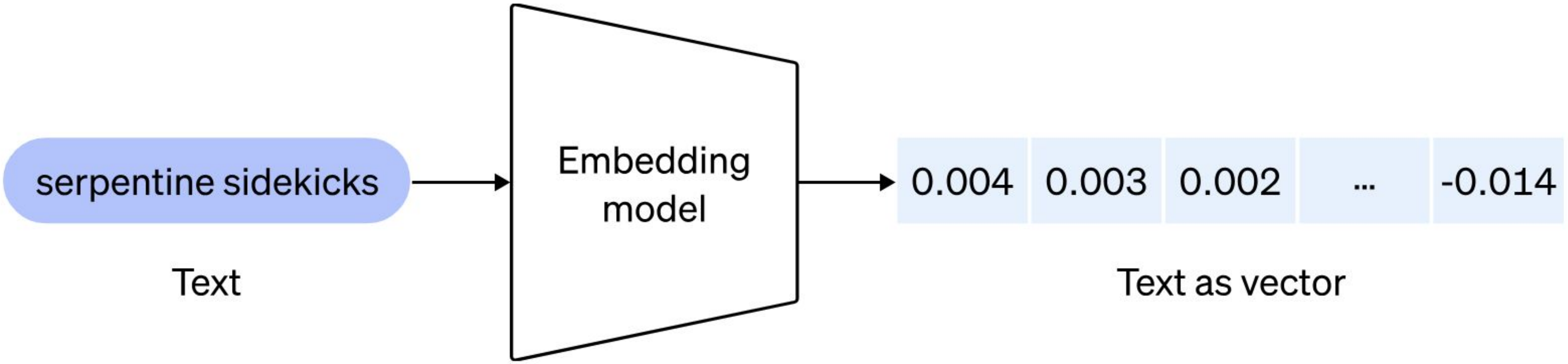
```
postgres=# Create extension vector;
CREATE EXTENSION

CREATE TABLE documents (
    id int PRIMARY KEY,
    title text NOT NULL,
    content TEXT NOT NULL
);
```

```sql
-- Create document_embeddings table
CREATE TABLE document_embeddings (
    id int PRIMARY KEY,
    embedding vector(1536) NOT NULL
);
```

```sql
CREATE INDEX document_embeddings_embedding_idx ON document_embeddings USING hnsw (embedding
vector_l2_ops);
```

```sql
-- Insert documents into documents table
INSERT INTO documents VALUES ('1', 'pgvector', 'pgvector is a PostgreSQL extension that provides support for vector similarity search and nearest neighbor search in SQL.');
INSERT INTO documents VALUES ('2', 'pg_similarity', 'pg_similarity is a PostgreSQL extension that provides similarity and distance operators for vector columns.');
INSERT INTO documents VALUES ('3', 'pg_trgm', 'pg_trgm is a PostgreSQL extension that provides functions and operators for determining the similarity of alphanumeric text based on trigram matching.');
INSERT INTO documents VALUES ('4', 'pg_prewarm', 'pg_prewarm is a PostgreSQL extension that provides functions for prewarming relation data into the PostgreSQL buffer cache.');
```

# What are embeddings and how do we generate them?


serpentine sidekicks
Text
Embedding model
0.004
0.003
0.002
...
-0.014
Text as vector

```python
# Python code to preprocess and embed documents
import openai
import psycopg2

# Load OpenAI API key
openai.api_key = "sk-..." #YOUR OWN API KEY

# Pick the embedding model
model_id = "text-embedding-ada-002"

# Connect to PostgreSQL database
conn = psycopg2.connect(database="postgres", user="gulcin.jelinek", host="localhost", port="5432")

# Fetch documents from the database
cur = conn.cursor()
cur.execute("SELECT id, content FROM documents")
documents = cur.fetchall()

# Process and store embeddings in the database
for doc_id, doc_content in documents:
    embedding = openai.Embedding.create(input=doc_content, model=model_id)['data'][0]['embedding']
    cur.execute("INSERT INTO document_embeddings (id, embedding) VALUES (%s, %s);", (doc_id, 
embedding))
    conn.commit()

# Commit and close the database connection
conn.commit()
```

# Querying embeddings

```
# Python code to preprocess and embed documents
import psycopg2

# Connect to PostgreSQL database
conn = psycopg2.connect(database="postgres", user="gulcin.jelinek", host="localhost", port="5432")

cur = conn.cursor()
# Fetch extensions that are similar to pgvector based on their descriptions
query = """
WITH pgv AS (
    SELECT embedding
      FROM document_embeddings JOIN documents USING (id)
     WHERE title = 'pgvector'
)
SELECT title, content
  FROM document_embeddings
  JOIN documents USING (id)
 WHERE embedding <-> (SELECT embedding FROM pgv) < 0.5;"""
cur.execute(query)

# Fetch results
results = cur.fetchall()

# Print results in a nice format
for doc_title, doc_content in results:
    print(f"Document title: {doc_title}")
    print(f"Document text: {doc_content}")
    print()
```

```
❯ python3 query.py
Document title: pgvector
Document text: pgvector is a PostgreSQL extension that provides support for vector similarity search
and nearest neighbor search in SQL.

Document title: pg_similarity
Document text: pg_similarity is a PostgreSQL extension that provides similarity and distance
operators for vector columns.
```

# Trade-off analysis

- Performance
- Cost

- Accuracy
- Precision
- Recall

# Indexing vectors

- pgvector performs **“exact nearest neighbor search”** by default
- Add index to use **“approximate nearest neighbor search”**
- Supported index types: IVFFlat, HNSW (0.5.0)

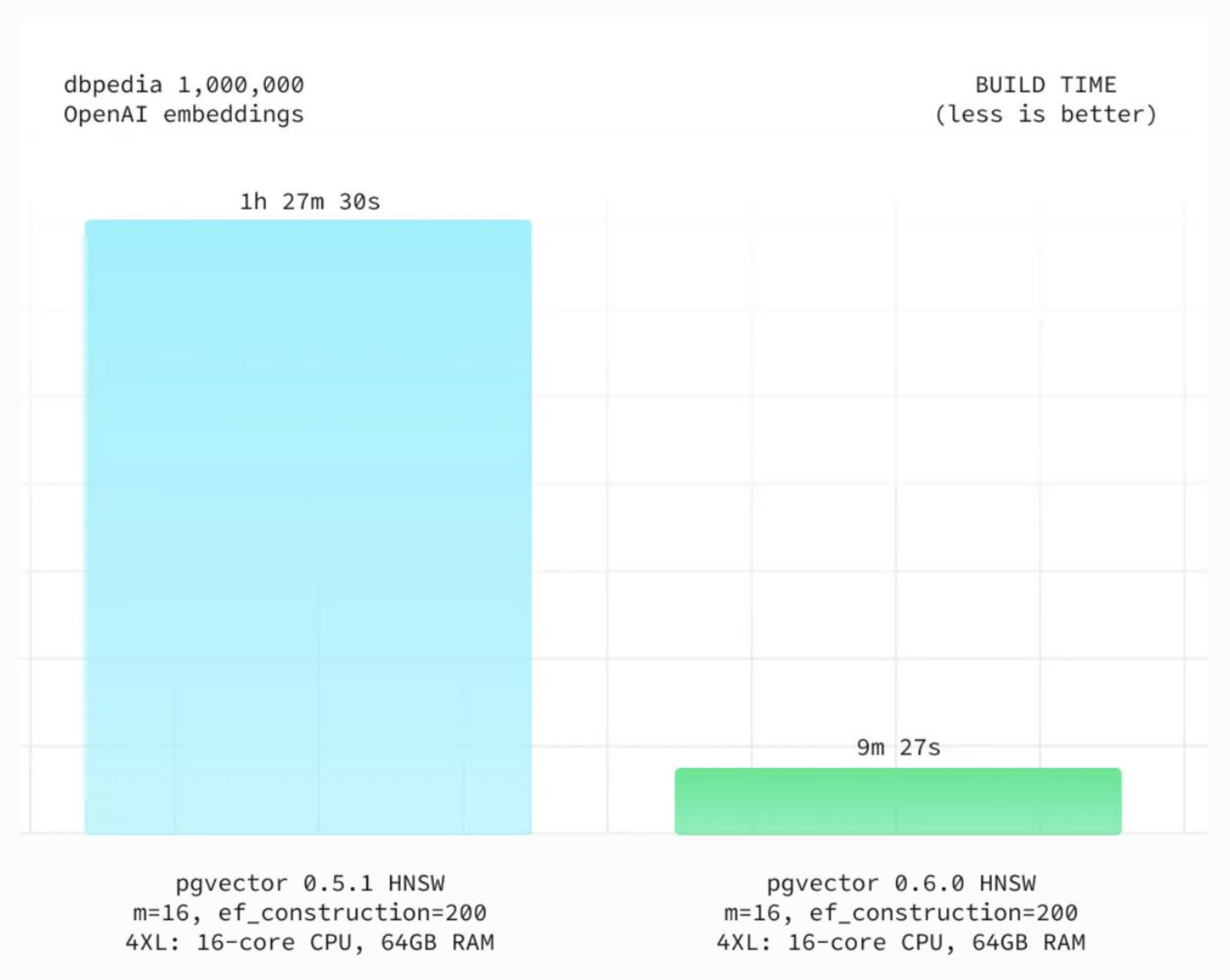
# Index types

## IVFFlat

- Divides vectors into lists
- Faster build times
- Uses less memory
- Lower query performance (speed-recall tradeoff)
- Create index after the table has some data

## HNSW

- Creates a multilayer graph
- Slower build times
- Uses more memory
- Better query performance
- Index can be created without any data in the table (no training step)

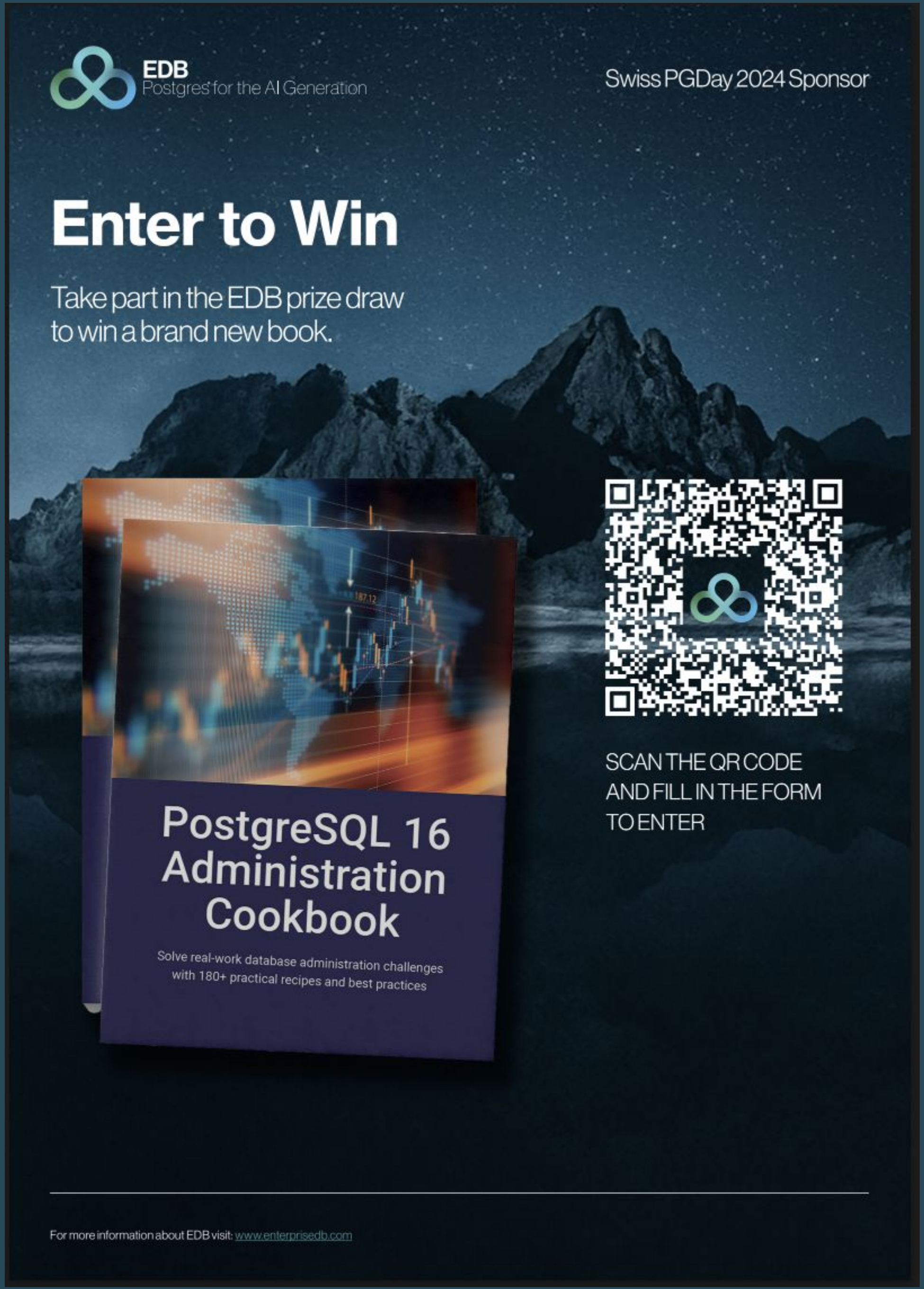

dbpedia 1,000,000
OpenAI embeddings
BUILD TIME
(less is better)
1h 27m 30s
9m 27s
pgvector 0.5.1 HNSW
m=16, ef_construction=200
4XL: 16-core CPU, 64GB RAM
pgvector 0.6.0 HNSW
m=16, ef_construction=200
4XL: 16-core CPU, 64GB RAM

# Future of vectors and Postgres

- pgvector 0.7.0 (29 April 2024)
  - Add halfvec and sparsevec type
  - Support for bit vectors to HNSW
  - Add hamming_distance function and jaccard_distance function
  - Add quantize_binary function and subvector function
  - Updated comparison operators to support vectors with different dimensions
- pgvector 0.6.0 (29 Jan 2024)
  - Support for parallel index builds for HNSW
  - Improved performance of HNSW
  - Reduced memory usage and reduced WAL generation for HNSW index builds

# Problems to-be-solved

- Hybrid search, hybrid ranking
  - Efficient filtering with vector search
  - Hybrid combinations of text scoring functions, like BM25, with vector distance
  - Pre-filtering, post-filtering?
- Multi-column indexing
- Multi-vector indexing
  - Multi-modal product search: Useful for product and e-commerce applications
  - Each data item represented by one vector is not very realistic for large documents
  - Index multiple vectors per document (for large text documents)
  - Retrieve documents by the closest vector in each
- Cost of environment (hardware)
  - Dependency to GPU, GPU-optimized instances are $$$
  - How to tune for lowest possible resource usage
- Scaling vector data for production


EDB
Postgres for the AI Generation
Swiss PGDay 2024 Sponsor
Enter to Win
Take part in the EDB prize draw to win a brand new book.
PostgreSQL 16 Administration Cookbook
Solve real-work database administration challenges with 180+ practical recipes and best practices
SCAN THE QR CODE AND FILL IN THE FORM TO ENTER
For more information about EDB visit: www.enterprisedb.com


Danke! Merci vilmal!