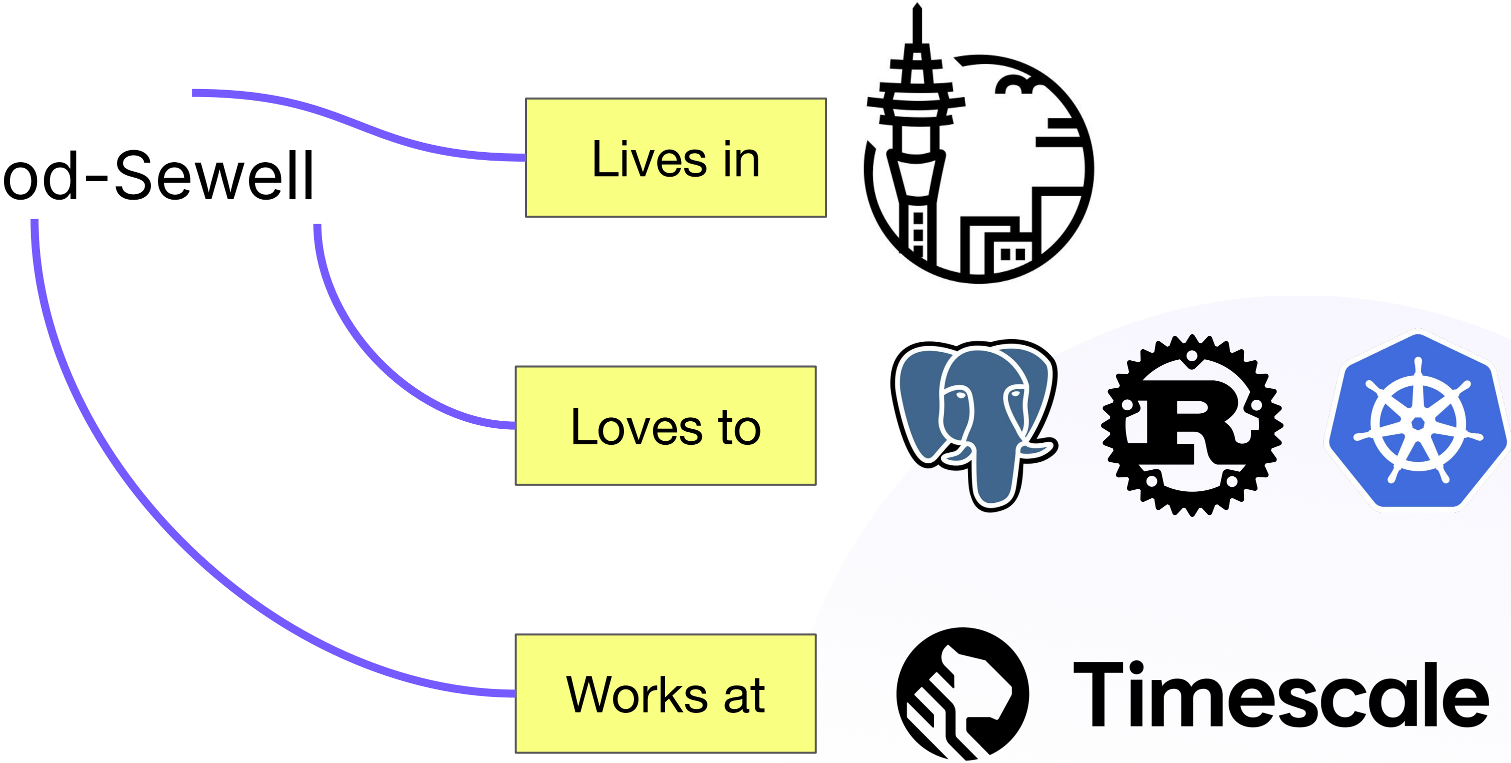
The Timescale PGAI extensions: pgai and pgvectorscale

James Blackwood-Sewell



Lives in

Loves to

Works at

Agenda

01 Who is Timescale?

02 What is PGAI?

03 The pgai extension

04 The pgvectorscale extension

Who is Timesale?

- Timescale is a company focused on empowering developers to build and run scalable, performant databases on PostgreSQL.
    - We are the creators of the TimescaleDB C extension, that combines the capabilities of time-series databases with the reliability and flexibility of Postgres.
    - We are the creators of the PGAI suite of extensions
    - We have a developer focused cloud DBaaS platform

You can build AI applications with PostgreSQL

Software engineers
DBAs
Data engineers
AI engineers

(Everyday application
developers with
no specialized
AI/ ML background
needed)

You can build AI applications with PostgreSQL

What AI systems is PostgreSQL good for?

Use case	**RAG**	**Search**	**Agents**	**Text to SQL**	**And more**
Description	“ChatGPT” but with your company/ customer data *(Retrieval Augmented Generation)*	Search and find relevant information by meaning (not keyword/ tag)	ChatGPT that can use tools, plan, and act autonomously Tools = web search, database query, code, APIs etc.	ChatGPT but for numerical, structured, tabular data Asking questions in English vs writing formulas / code	Recommendation Systems (RecSys) Anomaly Detection
Example apps	- Customer support chatbot - Research Copilot - Docs chatbot	- Semantic Search - Image/ video search	- AI Software engineer (e.g Devin, Replit Agents)	- “Chat with your data” - Data / Financial Analysis Agent	- Time-series anomaly detection - E-commerce purchase recommendations

What is PGAI?

- PGAI is a suite of AI tooling for PostgreSQL, designed to bring the capabilities of modern machine learning and AI directly into the database.
- It enables developers to us AI without leaving their Postgres environment
- Includes:
  - pgvector extension (community written)
  - pgvectorscale extension (Timescale written)
  - pgai extension (Timescale written)

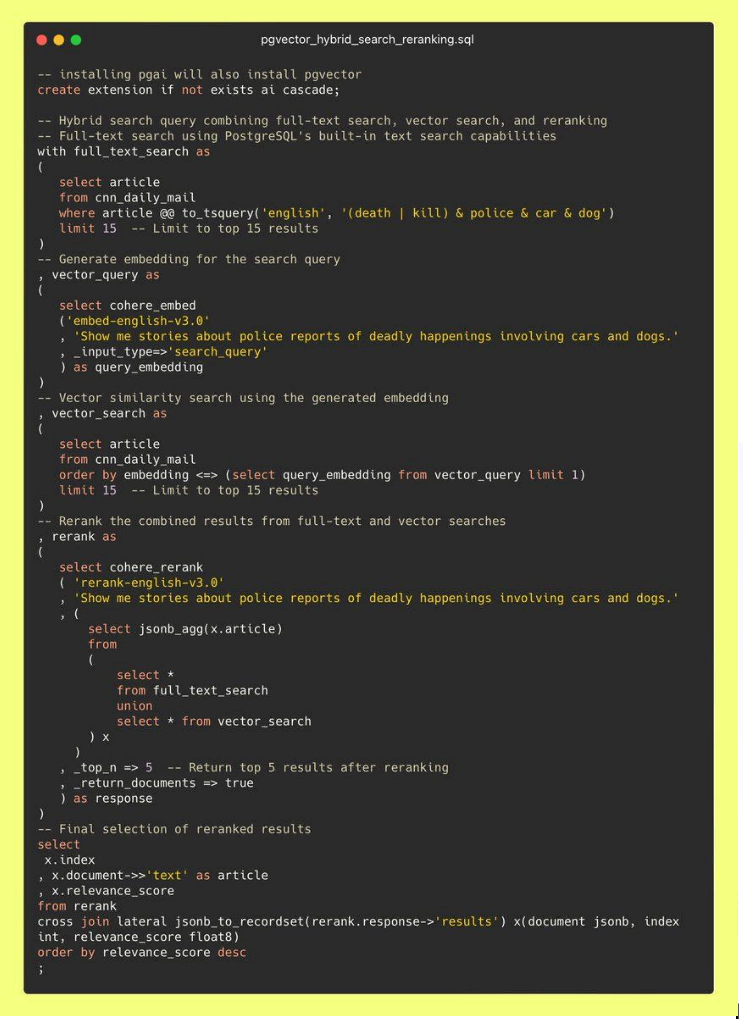
PostgreSQL extensions for AI applications

AI extension	**Why it's useful**	**License**
pgvector	• Gives PostgreSQL vector database super powers! • Vector data type, distance functions (cosine, L1, L2, inner product) • Vector search indexes (HNSW, IVFFLAT)	Open-source (PostgreSQL)
pgvectorscale	• Speeds up pgvector for large scale workloads • Complement to pgvector (you use them together) • High accuracy filtered search • Vector search index (StreamingDiskANN)	Open-source (PostgreSQL)
pgai	• Brings AI workflows to PostgreSQL • Embedding creation • In-database LLM reasoning (summarization, moderation, categorization)	Open-source (PostgreSQL)

The pgai reality

Hybrid Search in 1 PostgreSQL query!

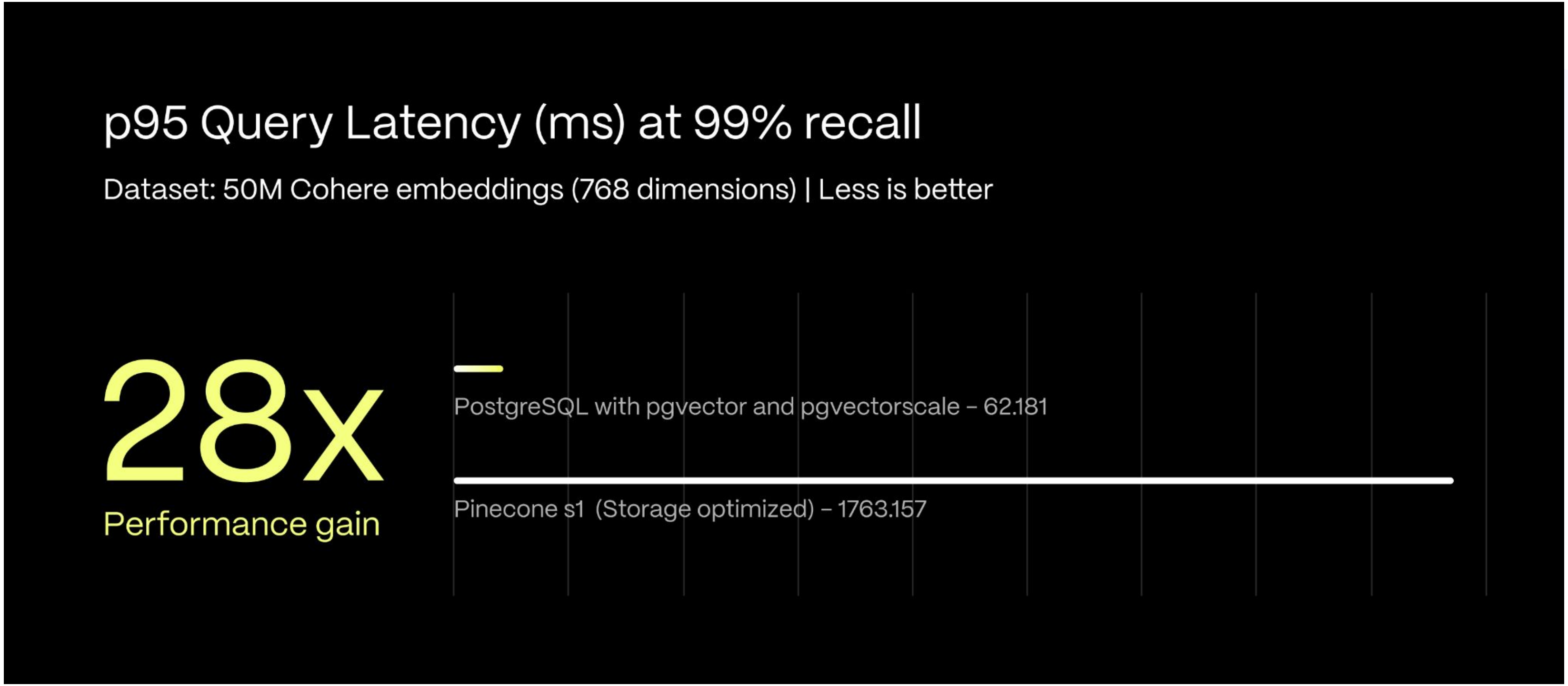
- Includes vector search,
- keyword search
- reranking via @cohere Rerank model



```sql
-- installing pgai will also install pgvector
create extension if not exists ai cascade;

-- Hybrid search query combining full-text search, vector search, and reranking
-- Full-text search using PostgreSQL's built-in text search capabilities
with full_text_search as
(
   select article
   from cnn_daily_mail
   where article @@ to_tsquery('english', '(death | kill) & police & car & dog')
   limit 15  -- Limit to top 15 results
)
-- Generate embedding for the search query
, vector_query as
(
   select cohere_embed
   ('embed-english-v3.0'
   , 'Show me stories about police reports of deadly happenings involving cars and dogs.'
   , _input_type=>'search_query'
   ) as query_embedding
)
-- Vector similarity search using the generated embedding
, vector_search as
(
   select article
   from cnn_daily_mail
   order by embedding <=> (select query_embedding from vector_query limit 1)
   limit 15  -- Limit to top 15 results
)
-- Rerank the combined results from full-text and vector searches
, rerank as
(
   select cohere_rerank
   ( 'rerank-english-v3.0'
   , 'Show me stories about police reports of deadly happenings involving cars and dogs.'
   , (
       select jsonb_agg(x.article)
       from
       (
           select *
           from full_text_search
           union
           select * from vector_search
       ) x
     )
   , _top_n => 5  -- Return top 5 results after reranking
   , _return_documents => true
   ) as response
)
-- Final selection of reranked results
select
 x.index
, x.document->>'text' as article
, x.relevance_score
from rerank
cross join lateral jsonb_to_recordset(rerank.response->'results') x(document jsonb, index
int, relevance_score float8)
order by relevance_score desc
;
```

The pgvectorscale reality



p95 Query Latency (ms) at 99% recall

Dataset: 50M Cohere embeddings (768 dimensions) | Less is better

28x

Performance gain

Call to Action!

We would love contributors on these projects

- If you’re a Python person, or interested in using AI models from Postgres

  ➡️ pgai, https://github.com/timescale/pgai

- If you’re a Rust person, or interested in how vectors are stored and queried in Postgres

  ➡️ pgvectorscale, https://github.com/timescale/pgvectorscale