The New Postgres AI Ecosystem

Shaun Thomas

Principal Software Engineer

February 18th, 2026

pgEdge®

Who are pgEdge?

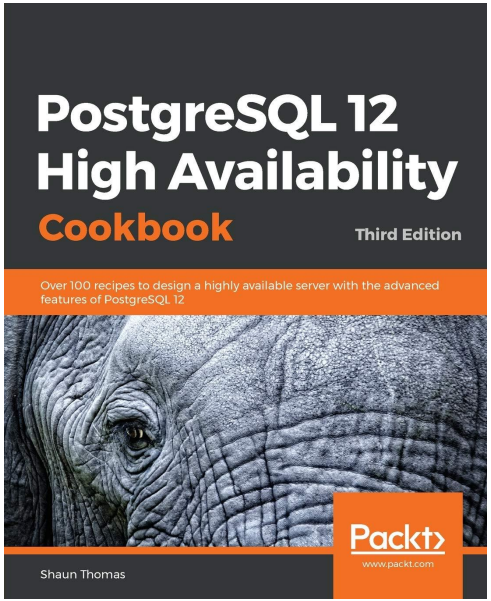
- Distributed Postgres
- Active-Active clusters
- Postgres AI Tools
- Cloud Services
- Platform Automation
- Ultra High Availability

www.pgedge.com

Who am I?

- Author
- Speaker
- Blogger
- Mentor
- Dev

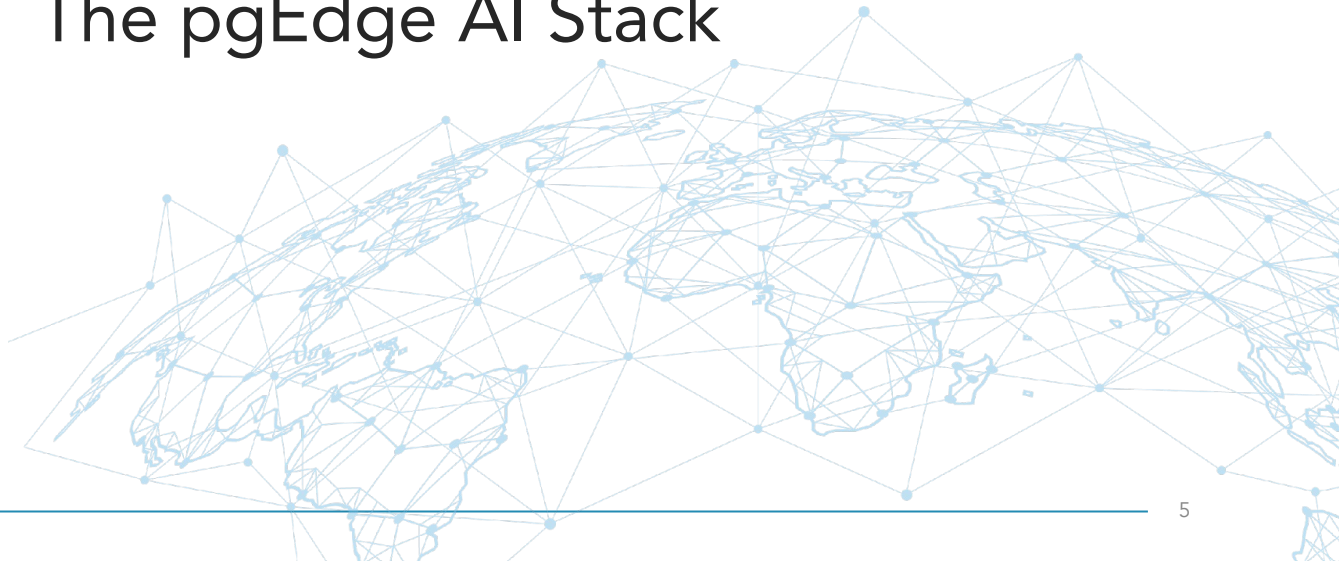
shaun.thomas@
pgedge.com



What's Our Agenda?

Stuff We're Talking About

The pgEdge AI Stack



pgedge-docloader

Transform and load your documents

https://github.com/pgEdge/pgedge-docloader/

pgedge-vectorizer

Vectorize those documents

https://github.com/pgEdge/pgedge-vectorizer/

pgedge-rag-server

Interrogate the documents

https://github.com/pgEdge/pgedge-rag-server/

pgedge-postgres-mcp

Let an LLM do all the hard work

https://github.com/pgEdge/pgedge-postgres-mcp/

Everyone Loves pgvector!

pgvector

Bestows vector abilities to Postgres

https://github.com/pgvector/pgvector

Added by pgvector

- Vector similarity searches
- Multiple new vector types (single, half, binary, sparse)
- Vector distance operators (`<->`, `<#>`, `<=>`, `<+>`, `<~>`, `<%>`)
- New vector index types (HNSW, IVF-FLAT)

Why Isn’t pgvector Enough?

Reason 1: Loading Data

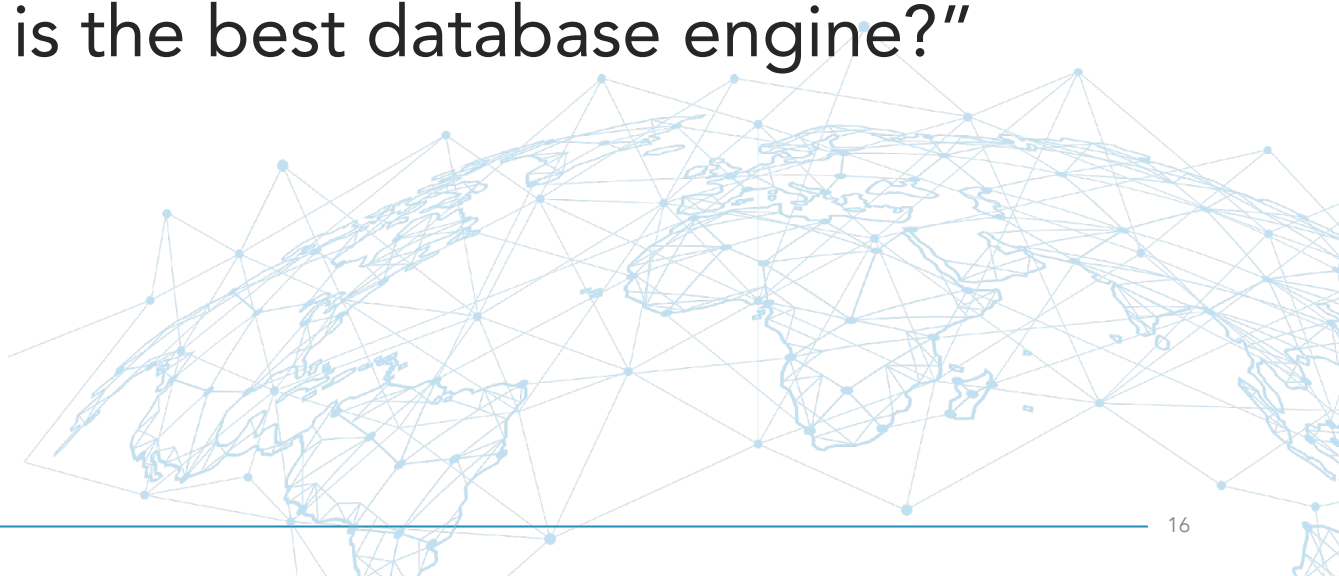
Documents come in many forms.

Reason 2: Obtaining Vectors

Pgvector needs vectors!

How Do I Turn This:

"What is the best database engine?"



Into this?

[-0.02595623,0.04631714,-0.053539883,0.011895365,0.0758225,-0.04304593,-0.006637965,-0.08208234,-0.04918979,-0.020363959,-0.038359903,0.01744871,-0.057595164,0.034587763
,-0.020651337,0.002429941,0.0018788559,-0.018510725,-0.09920806,0.12411486,-0.09942987,0.038612444,0.057046242,-0.015014563,0.03681107,0.029042058,-0.056116235,-0.007918
157,0.06834828,-0.027709357,-0.012434633,-0.0062096403,-0.015024162,-0.0882817,-0.010005957,0.0217961,0.020747224,0.00043326707,0.029898426,0.063303724,-0.023971524,-0.0
35034273,0.12894247,0.03956573,0.04099617,-0.036992185,0.039790176,-0.038303692,0.03762054,-0.016138878,-0.026407361,0.010406044,0.031098412,-0.059915572,0.0296487,0.018
585488,-0.0127668455,0.0698031,-0.023116358,-0.03830573,0.058555316,0.053015016,0.009442912,0.065988995,-0.025956836,0.0072427755,-0.035602763,0.049767125,0.027460659,0.
011989594,1.022185e-05,-0.04103233,-0.017008793,-0.026518255,-0.057895917,0.02913324,-0.007884655,-0.036250923,0.018677657,-0.051816877,0.036574055,-0.018310225,0.106847
58,0.015361703,-0.0068149795,-0.002467204,0.045794293,-0.03188524,-0.014328101,-0.04377825,-0.02258047,-0.05837506,0.008181678,-0.07910704,0.03463214,-0.020189477,-0.092
740774,-0.0002254515,-0.00661493,0.1312322,0.02023139,0.016226936,0.050397724,0.0049572135,0.009400744,-0.045763697,-0.071638376,-0.014594109,-0.018446293,0.028820504,0.
0023369463,0.053181294,0.058653817,-0.06454964,0.049355283,0.07178324,0.027783332,-0.067031115,-0.06841928,0.015850065,-0.002914686,0.009294329,-0.078147724,-0.01781891,
-0.07263269,0.017262291,-0.0061519933,0.014498569,0.07934687,-0.011039961,-0.014350844,0.009714252,0.07571004,-0.059741423,-0.061780307,-0.07044488,0.0017138183,-0.03665
142,0.06618329,-0.056741964,0.024425812,0.043776017,-0.05947509,0.02473815,-0.033279914,0.06721659,0.012232149,0.0015699323,0.007885537,0.00707865,0.013194744,-0.0681915
66,-0.12272909,0.06650073,-0.02412729,0.04940419,0.08976135,0.016346294,-0.042974483,0.0075128144,0.13506782,0.013340274,0.013941901,-0.0135494545,0.019012375,-0.0450566
34,-0.024806282,-0.025400957,0.009210025,-0.085539885,-0.0014276546,-0.047662564,0.028403034,-0.031291023,0.00994239,0.013966853,0.029291267,-0.06537566,-0.0023040709,-0
.022339806,0.05957562,0.0032288802,-0.026567612,0.054026626,0.07418133,-0.11601187,0.14578743,0.06701949,0.089334145,0.013379732,0.039292034,-0.029553873,0.020182345,-0.
027620139,0.033731233,0.029958928,0.021263465,0.0116131,0.024114138,0.036053922,0.010862184,-0.11032744,0.029497253,0.03680072,0.015323135,-0.02569687,0.020646175,-0.003
09678,0.075037666,-0.012467476,-0.012603479,0.05536957,0.06923356,0.041376483,-0.05493469,0.07284344,-0.0024210871,0.024228476,-0.054416776,0.09758099,0.015991757,-0.026
029492,0.005204354,2.1359543e-32,0.02700274,-0.06537937,-0.057982467,-0.058108676,0.024990669,0.008049355,0.016007772,-0.019222062,0.055540632,0.014360761,0.02189043,-0.
039927147,-0.06621141,-0.007778538,-0.032505617,-0.015146801,0.030141199,0.047050603,-0.0278275,0.04865551,0.07719417,-0.048471287,-0.069588214,-0.050331596,0.041957315,
0.12916774,0.10859817,0.009190485,-0.05403324,-0.08558693,0.04856777,0.010237227,-0.09778996,0.032434497,-0.05686069,0.11311847,0.0040654135,-0.055423062,0.044098742,-0.
08351652,-0.0066194735,0.0051483805,-0.013018369,0.09141706,-0.011138346,0.03484014,-0.09798947,0.009890583,0.052184697,-0.016177202,-0.12128752,-0.05317396,-0.038664415
,0.053813018,0.025762321,-0.010391627,-0.027447335,-0.09687913,-0.040417686,0.05761224,-0.0049005016,-0.03860952,-0.10431886,0.09482661,0.08394817,-0.05782826,-0.0233843
07,-0.033743203,0.01319146,0.020009484,-0.06339285,0.008339009,-0.10972377,-0.09203553,0.02314593,-0.026981864,-0.0098597845,-0.00695105,-0.04888554,-0.054383095,-0.0033
409367,-0.016765589,-0.020665072,0.03518574,-0.0975508,0.03954543,-0.027971495,0.022485066,-0.03068828,0.044939965,0.014050996,-0.02814454,-0.056048892,-0.027148627,-0.0
22608032,7.592085e-32,0.024655852,-0.03308234,-0.119617596,-0.020011967,0.08908686,-0.10233242,0.041305285,-0.019912839,0.008432649,0.08246976,-0.007695544,-0.013220983,
0.04306117,-0.061375756,0.10317889,-0.0032164725,-0.06101632,-0.054768626,0.06190977,0.020685453,0.091767095,-0.030094603,-0.010625265,0.011956352,-0.001202916,0.0814042
24,0.00017668601,0.053858735,0.11105762,0.03965099,0.055190314,-0.0008298795,-0.03585047,0.02358887,-0.07300523,-0.09976991,0.04071222,-0.017766878,0.083444,-0.014780061
,0.1179988,-0.047808193,0.027711963,0.010073332,0.06527614,-0.081142455,-0.04021762,0.07025154,0.06898177,-0.022367012,-0.06016291,0.020527564,-0.0048388843,-0.015055914
,0.06347836,-0.028675102,-0.04353604,0.0039767306,0.0139750345,-0.10406179,0.03652024,0.05376024,-0.07579619,0.003702582]

Reason 3: Content is Too Big!

You can’t transform a 10-page article into a single vector!

Reason 4: Postgres Isn’t an LLM

Where do you send results

once you have them?

Reason 5: Vectors Aren’t Always Enough

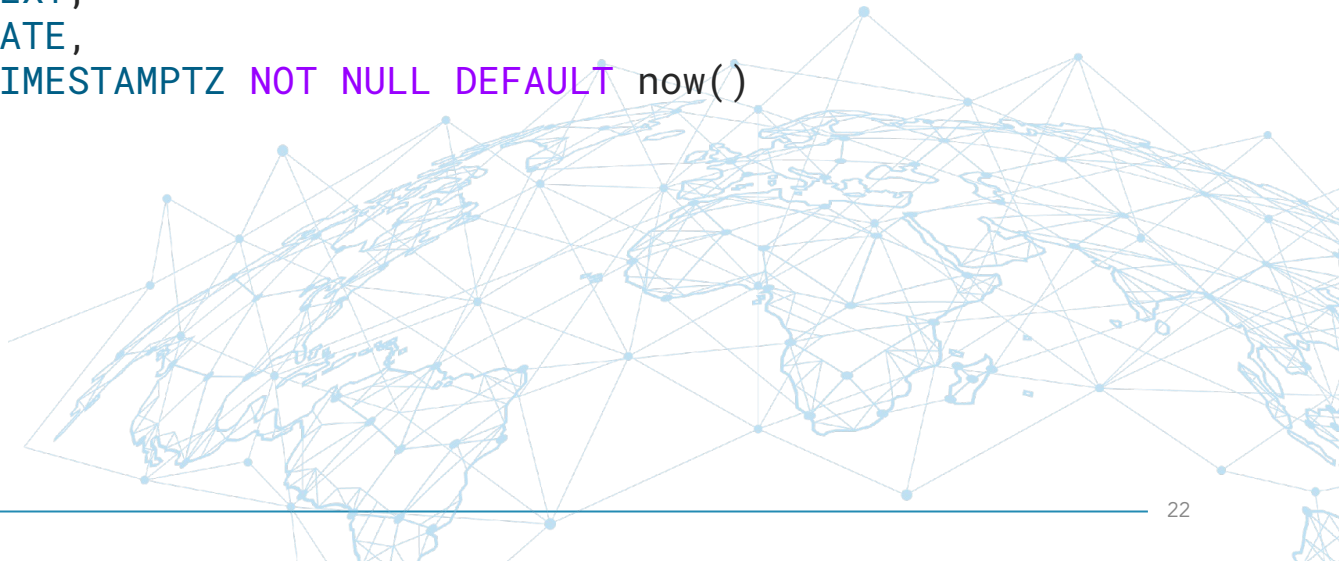
LLM interaction can

go beyond vectors.

Preparation Steps

A Place for Blogs

```
CREATE TABLE blog_articles (
    id             BIGINT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    author         TEXT,
    title          TEXT,
    content        TEXT,
    file_name      TEXT,
    publish_date   DATE,
    last_updated   TIMESTAMPTZ NOT NULL DEFAULT now()
);
```



Some Indexes

```
CREATE INDEX idx_blog_article_title ON blog_articles (title);
CREATE INDEX idx_blog_article_author ON blog_articles (author);
CREATE INDEX idx_blog_article_file_name ON blog_articles (file_name);
CREATE INDEX idx_blog_article_last_updated ON blog_articles (last_updated);
```

Solving ETL

What is pgedge-docloader?

It's a CLI tool written in Go that:

- Parses many document types for content (html, md, rst, xml, sgml).
- Converts document contents into Markdown.
- Extracts common document metadata (title, created, filename, etc).
- Loads into target table.
- Can pull documents from git for processing.

https://docs.pgedge.com/pgedge-docloader/

https://github.com/pgEdge/pgedge-docloader/

Installing the Tool

```
git clone https://github.com/pgEdge/pgedge-docloader.git
cd pgedge-docloader
make deps
make build
make install
```

Create a Dedicated User

```
CREATE USER docloader;
GRANT SELECT, INSERT, UPDATE ON blog_articles TO docloader;
GRANT USAGE, SELECT ON SEQUENCE blog_articles_id_seq TO docloader;
```

Load Some Documents

```
pgedge-docloader \
  --source ./content \
  --db-host localhost --db-port 5518 --db-name postgres \
  --db-user docloader \
  --db-table blog_articles \
  --col-doc-title title \
  --col-doc-content content \
  --set-column author="Shaun Thomas" \
  --col-file-name file_name \
  --col-file-modified publish_date \
  --col-row-updated last_updated
```

Sample Results

```
Processing files from: ./content
Processed 592 file(s), skipped 10 file(s)
Connecting to database docloader@localhost:5518/postgres

=== Processing Summary ===
Files processed: 592
Files skipped:   10
Rows inserted:   0
Rows updated:    592
==========================
```

Handling Vectorization

What is pgedge-vectorizer?

It's a Postgres extension that provides:

- Functions to transform text to embedding vectors.
- Functions to split content into chunks for better relevance.
- Processing queue + workers for generating embeddings.
- Compatible with OpenAI, Voyage AI, and Ollama.

https://docs.pgedge.com/pgedge-vectorizer/

https://github.com/pgEdge/pgedge-vectorizer/

Installing the Extension

```
git clone https://github.com/pgEdge/pgedge-vectorizer.git
cd pgedge-vectorizer
make
sudo make install
```

Configuring With OpenAI

```
shared_preload_libraries = 'pgedge_vectorizer'
pgedge_vectorizer.provider = 'openai'
pgedge_vectorizer.api_key_file = '~/.openai-key-file'
pgedge_vectorizer.model = 'text-embedding-3-small'
pgedge_vectorizer.num_workers = 5
pgedge_vectorizer.databases = 'postgres'
```

Configuring With Ollama

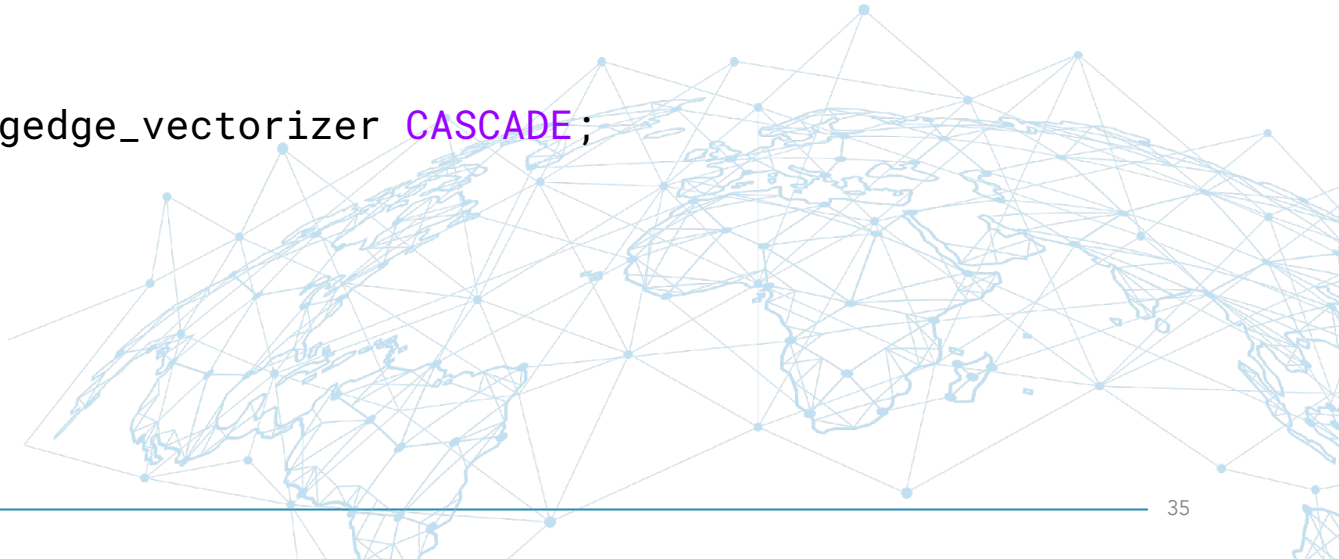
```
shared_preload_libraries = 'pgedge_vectorizer'
pgedge_vectorizer.provider = 'ollama'
pgedge_vectorizer.api_url = 'http://my-ollama-host:11434'
pgedge_vectorizer.model = 'embeddinggemma'
pgedge_vectorizer.num_workers = 5
pgedge_vectorizer.databases = 'postgres'
```

Create the Extension

```
CREATE EXTENSION vector;
CREATE EXTENSION pgedge_vectorizer;
```

OR:

```
CREATE EXTENSION pgedge_vectorizer CASCADE;
```



Functionality Test

```
SELECT count(*)
  FROM unnest(pgedge_vectorizer.generate_embedding(
         'This is a test'
       )::REAL[]);

 count
-------
   768
```

Processing Vectors

```
SELECT pgedge_vectorizer.enable_vectorization(
    source_table   := 'blog_articles',
    source_column  := 'content',
    chunk_strategy := 'markdown',
    chunk_size     := 400,
    chunk_overlap  := 50
);
```

What Processing Does

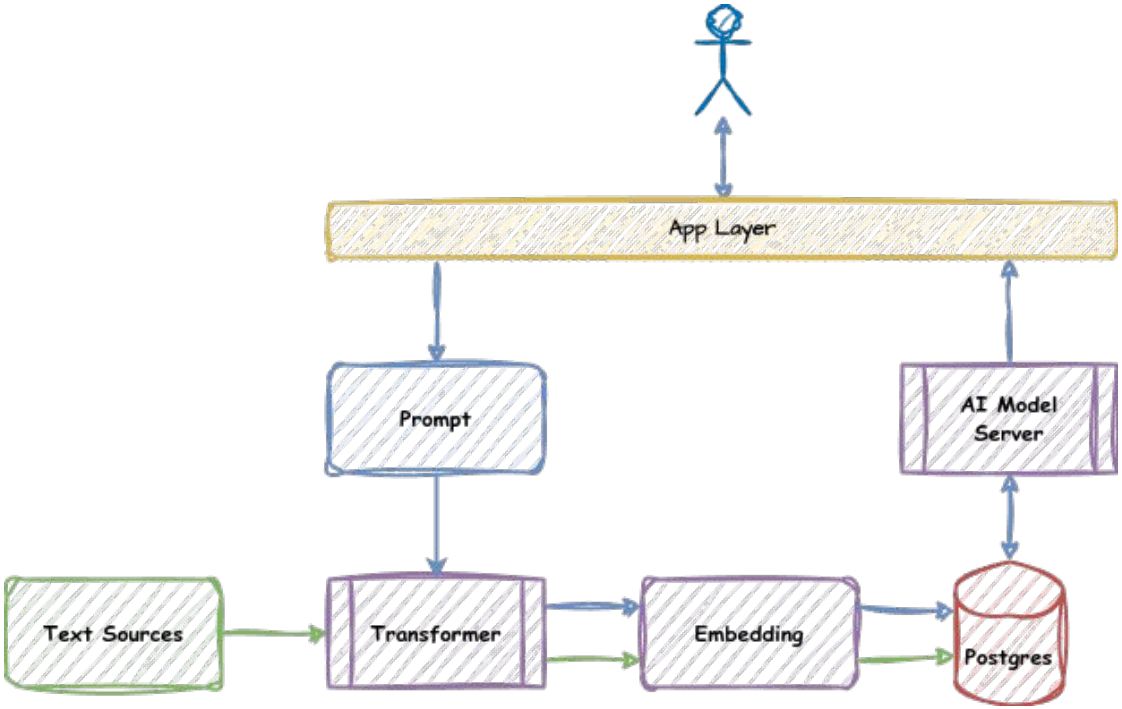
- Creates a table for content chunks.
- Splits content into chunks for efficient embeddings.
- Queues all content chunks for embedding generation.
- Workers handle all subsequent queue processing.
- Adds source trigger to enqueue new / updated content.
- Generates vectors / embeddings for pgvector!

Fetching Results

```
SELECT a.title, c.content
  FROM blog_articles_content_chunks c
  JOIN blog_articles a ON (c.source_id = a.id)
 ORDER BY c.embedding <=> pgedge_vectorizer.generate_embedding(
         'What is the best database engine?'
       )
 LIMIT 5;
```

Retrieval Augmented Generation

What a RAG Service Does



What is pgedge-rag-server?

It's a Go service that will:

- Monitor for HTTP/HTTPS calls.
- Query embedding tables for best semantic matches.
- Send results to LLM as references.
- Return LLM response.

https://docs.pgedge.com/pgedge-rag-server/

https://github.com/pgEdge/pgedge-rag-server/

Installing the Service

```
git clone https://github.com/pgEdge/pgedge-rag-server.git
cd pgedge-rag-server
make build
```

Configuring With YAML

```
server:
    listen_address: "0.0.0.0"
    port: 8080

pipelines:
    - name: "blog-rag"
      description: "Cool Blog Searching with RAG"
      database:
          host: "localhost"
          port: 5518
          database: "postgres"
          username: "postgres"
      tables:
          - table: "blog_articles_content_chunks"
            text_column: "content"
            vector_column: "embedding"

      embedding_llm:
          provider: "ollama"
          model: "embeddinggemma"
      rag_llm:
          provider: "ollama"
          model: "glm-4.7-flash"
      top_n: 5
```

Service Invocation

```
./bin/pgedge-rag-server -config pgedge-rag-server.yaml
```

Works great with a Docker image!

Sample Query

```
curl -X POST http://localhost:8080/v1/pipelines/blog-rag \
  -H "Content-Type: application/json" \
  -d '{"query": "What is the best database engine?"}'
```

Sample Results


```
{
  "answer": "Based on the provided context, there are two perspectives
regarding the \"best\" database engine:\n\n*   **Postgres:** Several
documents state that **Postgres** is the best RDBMS engine currently
available.\n*   **Context-dependent:** One document notes that Postgres is
an excellent database engine, but \"whether it's the 'best' depends on your
specific needs.\"",
  "tokens_used": 1506
}
```


Use A View Instead

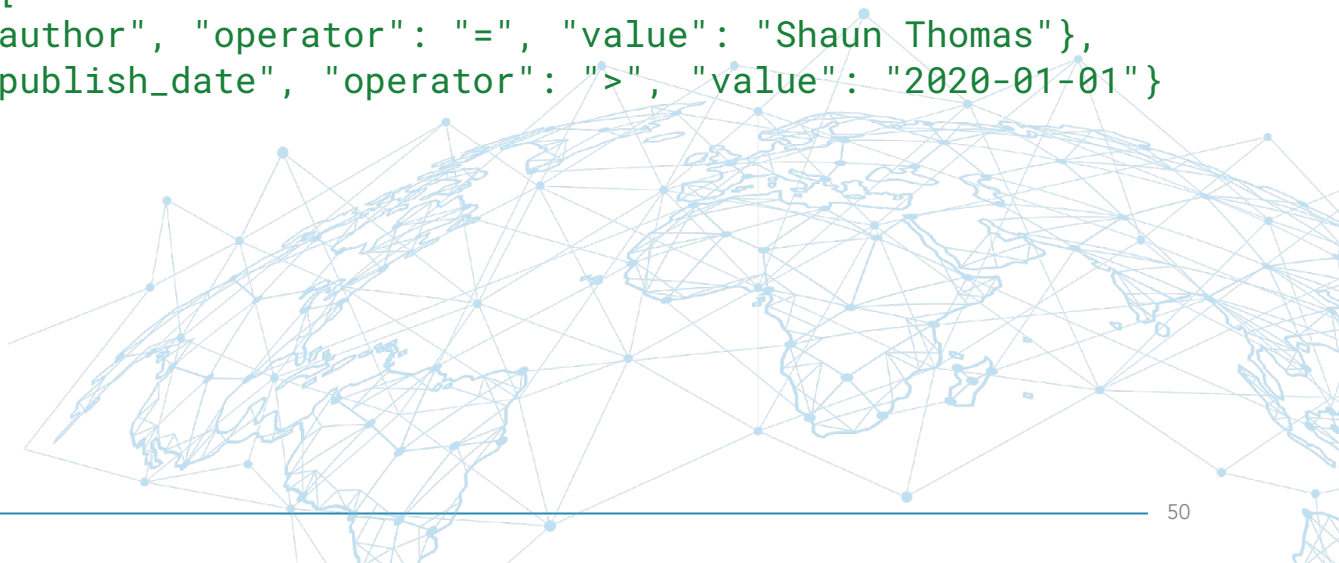
```
tables:
  - table: "v_blog_article_embeddings"
    text_column: "chunk_content"
    vector_column: "embedding"
    metadata_columns:
      - "author"
      - "title"
      - "file_name"
      - "publish_date"
```

Make Sure the View Exists

```
CREATE OR REPLACE VIEW v_blog_article_embeddings AS
SELECT c.*, a.title, a.author, a.file_name,
       a.content AS source_content, a.publish_date
  FROM blog_articles a
  JOIN blog_articles_content_chunks c ON (c.source_id = a.id);
```

Search Metadata Columns

```
curl -X POST http://localhost:8080/v1/pipelines/blog-rag \
  -H "Content-Type: application/json" \
  -d '{
    "query": "What is the best database engine?",
    "filter": {
      "conditions": [
        {"column": "author", "operator": "=", "value": "Shaun Thomas"},
        {"column": "publish_date", "operator": ">", "value": "2020-01-01"}
      ],
      "logic": "AND"
    }
  }'
```



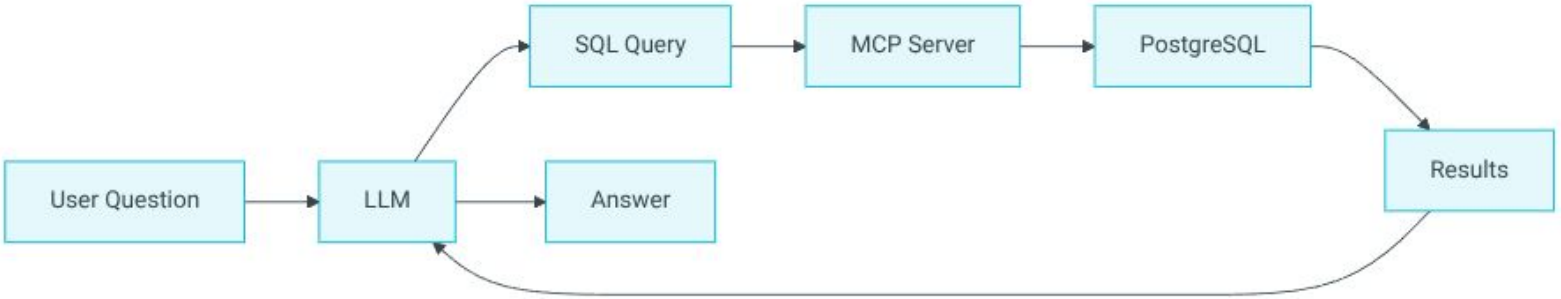
Extra Functionality

The service can also:

- Show content for each result sent to the LLM.
- Show relative match score for each reference.
- Stream tokens to the receiver as they're generated.
- Re-rank references using BM25 for keyword similarity.

Model Context Protocol

What an MCP Service Does



What is pgedge-postgres-mcp?

It's a Go service that will:

- Monitor for STDIO or HTTP/HTTPS calls.
- Use LLM to translate user queries into tool calls or SQL queries.
- Sends queries to the database, executes tool calls.
- Use LLM to interpret results.
- Return LLM response.
- BETA ONLY!

https://docs.pgedge.com/pgedge-postgres-mcp-server/

https://github.com/pgEdge/pgedge-postgres-mcp/

Installing the Service

```
git clone https://github.com/pgEdge/pgedge-postgres-mcp.git
cd pgedge-postgres-mcp
make build
```

Copy Sample Configs

```
cp examples/pgedge-postgres-mcp-http.yaml.example bin/postgres-mcp.yaml
cp examples/pgedge-postgres-mcp-users.yaml.example bin/postgres-mcp-users.yaml
cp examples/pgedge-postgres-mcp-tokens.yaml.example bin/postgres-mcp-tokens.yaml

# Create a symlink so the startup script sees the config it expects
ln -s postgres-mcp.yaml bin/pgedge-postgres-mcp-http.yaml
```

Configure HTTP

```
http:
    enabled: true
    address: ":8080"
    tls:
        enabled: false
        cert_file: "./server.crt"
        key_file: "./server.key"
        chain_file: ""
    auth:
        enabled: true
        token_file: "./postgres-mcp-tokens.yaml"
        user_file: "./postgres-mcp-users.yaml"
        max_failed_attempts_before_lockout: 5
        rate_limit_window_minutes: 15
        rate_limit_max_attempts: 10
```

Configure Databases

```
databases:
    - name: "main"
      host: "localhost"
      port: 5518
      database: "postgres"
      user: "postgres"
      password: ""  # Leave empty to use .pgpass file
      sslmode: "disable"
      pool_max_conns: 10
      pool_min_conns: 2
      pool_max_conn_idle_time: "5m"
      available_to_users: []
      allow_writes: false
      allow_llm_switching: true

      allowed_pl_languages:
          - plpgsql       # Always safe - standard PL/pgSQL
```

Configure Models

```
embedding:
    enabled: false
    provider: "ollama"
    model: "embeddinggemma:latest"

llm:
    enabled: true
    provider: "openai"
    model: "claude-sonnet-4-5-20250929"

    anthropic_api_key_file: "~/.anthropic-api-key"

    max_tokens: 4096
    temperature: 0.7
```

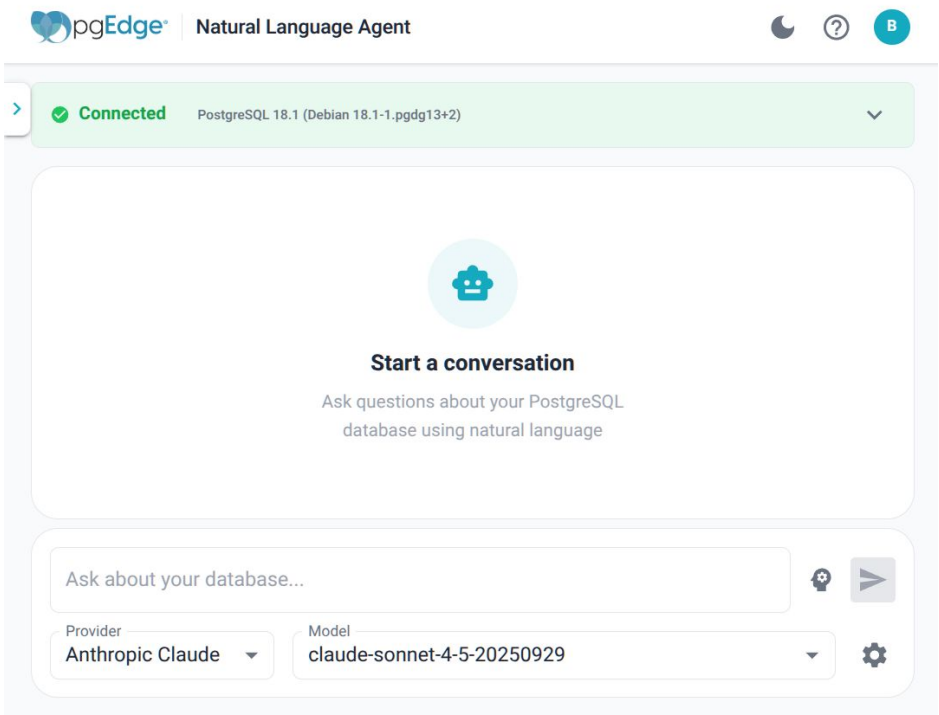
Create A User

```
./bin/pgedge-postgres-mcp -add-user \
    -username bones -user-note "Bones"
```

Start the Server and Client

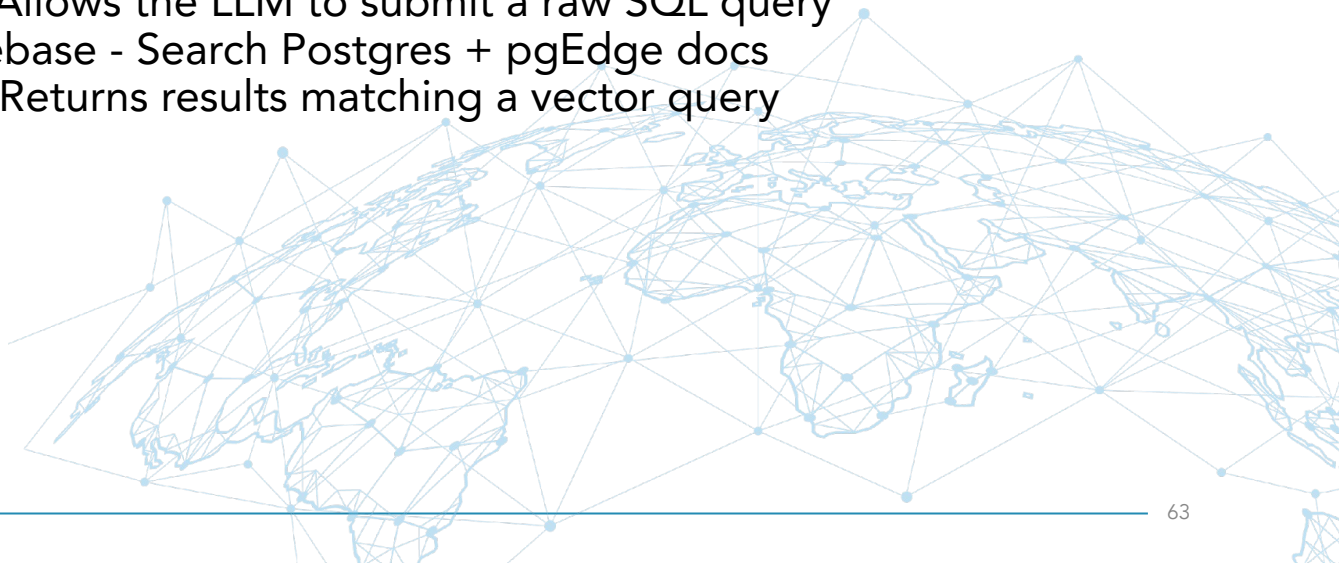
```
./start_web_client.sh
```

Enjoy!


pgEdge
Natural Language Agent
B
Connected
PostgreSQL 18.1 (Debian 18.1-1.pgdg13+2)
Start a conversation
Ask questions about your PostgreSQL database using natural language
Ask about your database...
Provider
Anthropic Claude
Model
claude-sonnet-4-5-20250929

What are the Tools?

- count_rows - Count rows from a particular table or query
- custom_tool - Invoke a custom user tool
- execute_explain - EXPLAIN ANALYZE a query
- generate_embedding - Transform text directly into an embedding
- get_schema_info - Return list of database objects, or one in depth
- query_database - Allows the LLM to submit a raw SQL query
- search_knowledgebase - Search Postgres + pgEdge docs
- similarity_search - Returns results matching a vector query



Questions