EDB
Postgres for the AI Generation

Postgres as AI Knowledge Base

Torsten Steinbach
VP, Chief Architect Analytics & AI

Slides

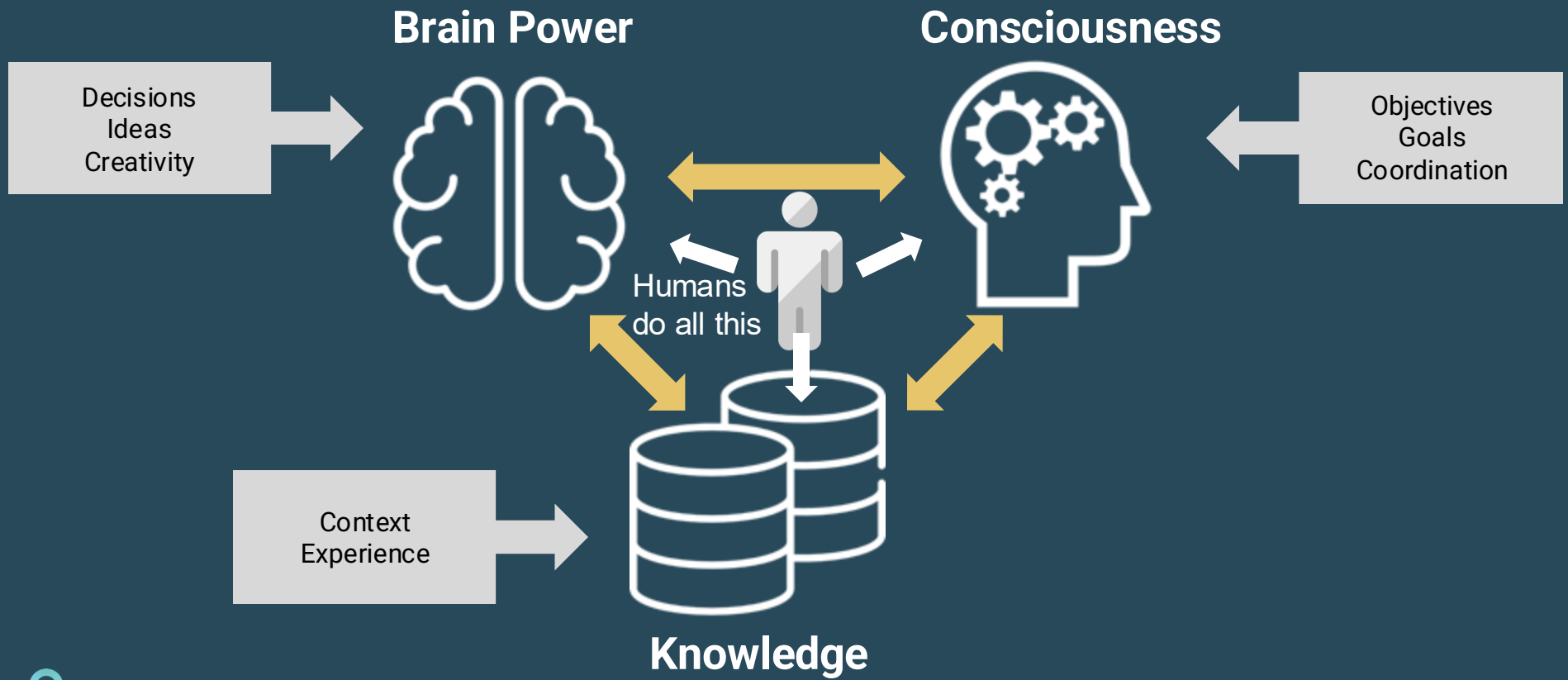
Content

1. Generative AI
2. Postgres as AI Knowledge Base
3. Agentic AI
4. AI Factory

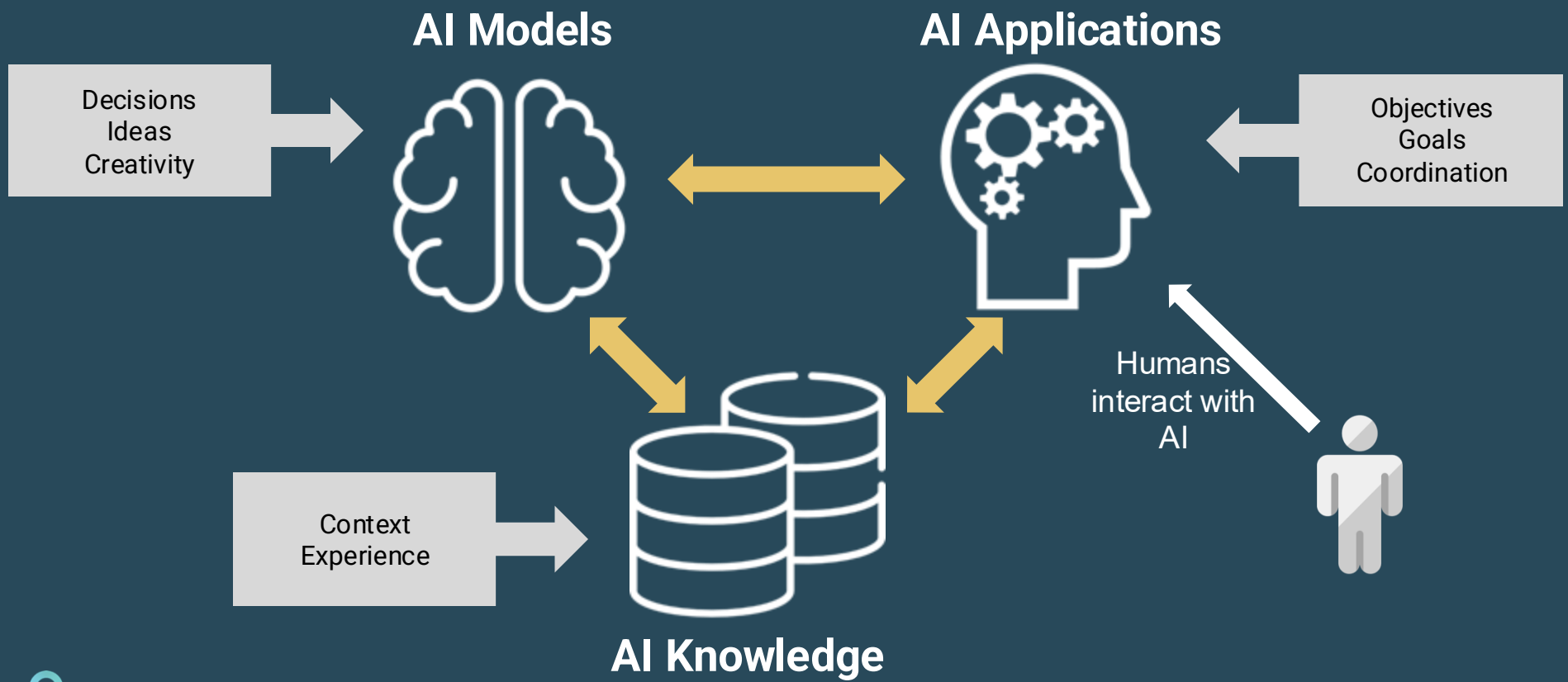
EDB
Postgres for the AI Generation

1. Generative AI

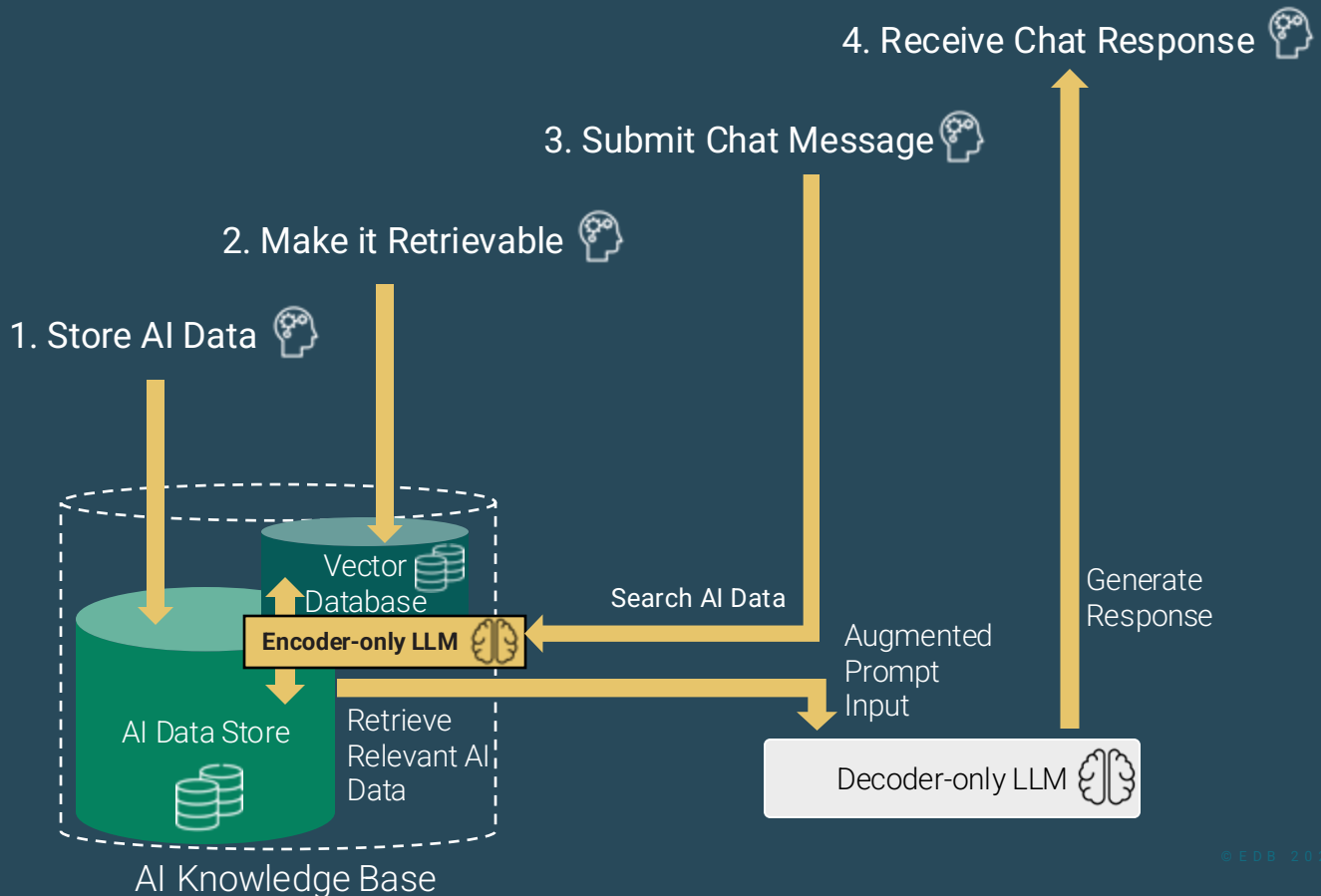
Human Intelligence



Artificial Intelligence



Chat Bots – The Omnipresent Gen AI Application

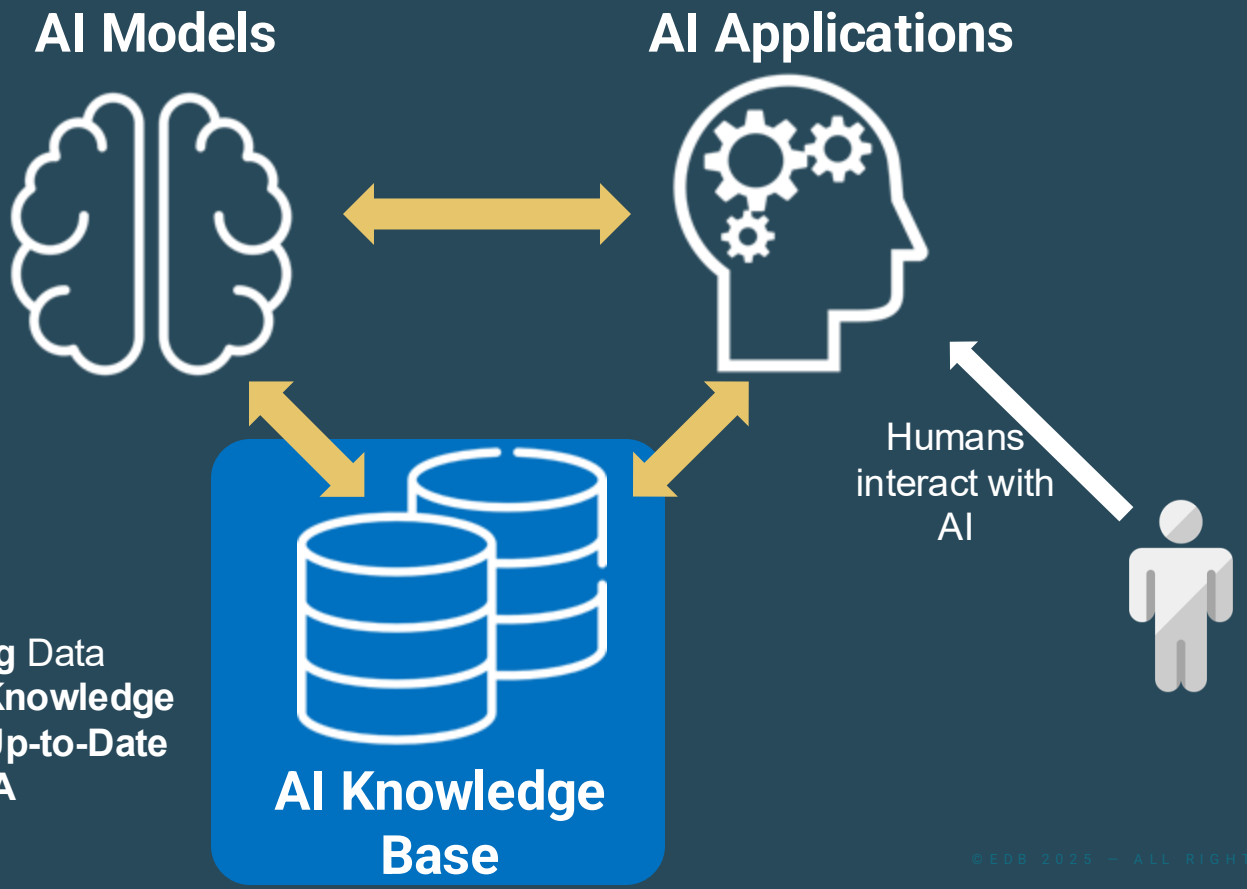


A.k.a.:
Retrieval
Augmented
Generation
= **RAG**

EDB
Postgres for the AI Generation

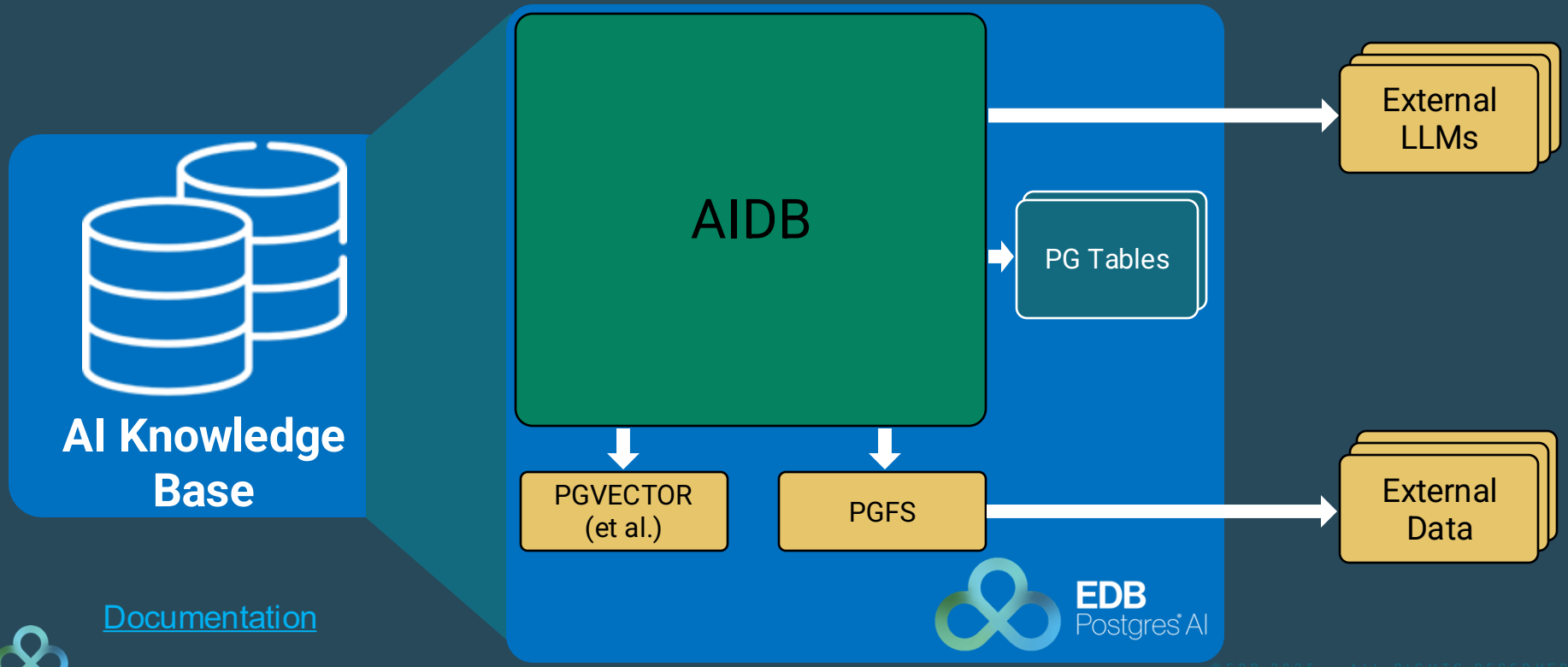
2. Postgres as AI Knowledge Base

AI Knowledge = A Database's Job



- **Storing** Data
- **Preparing** Data
- **Indexing & Searching** Data
- Data to **Retrievable Knowledge**
- Keeping Knowledge **Up-to-Date**
- Making Knowledge **HA**

EDB's AI Knowledge Base Stack in PG



Documentation

AI Knowledge Base – Meet Postgres

Storing Data (DATA, not vector embeddings)

- PG tables – but also: external storage (= natural habitat of unstructured)

Preparing Data – For consumption by RAG & agents

- Bite-sizing for AI (summarization, chunking)
- Normalizing data modality (PDF extraction, OCR)
- Generate vector embeddings

Indexing & Searching Data – (vector index)

- PGVECTOR is ONE answer
- Not the only one, not necessarily the best one…

Data to **Retrievable Knowledge**

- AI Knowledge Base for semantic retrieval
- Best practice: + reranking of the retrieved results

Keeping Knowledge **Up-to-Date**

- Automated Pipelines in Postgres
  - Including real-time execution

Making **Knowledge HA**

- The whole nine yards of Postgres

Storing AI Data with Postgres

Option 1: Regular PG tables

Option 2: EDB Postgres – Volumes

- Dedicated asset in PG to represent externally unstructured data
  - PDFs, HTMLs, Images, Voice etc. – just has mime type, no relational schema
  - Ideal for **large volumes** of data and **unstructured data**

```
SELECT pgfs.create_storage_location('ai_images', 's3://public-ai-images', NULL,
                                    '{"region": "eu-central-1", "skip_signature": "true"}', '{}');

SELECT aidb.create_volume('image_vol', 'ai_images', '/', 'Image');

SELECT * FROM aidb.list_volume_content('image_vol');
 file_name |  size  |      last_modified
-----------+--------+------------------------
 10000.jpg |   1030 | 2024-10-17 14:07:24 UTC
 10001.jpg |   1210 | 2024-10-17 14:07:24 UTC
 10002.jpg |    807 | 2024-10-17 14:07:24 UTC
 10003.jpg |  11564 | 2024-10-17 14:07:24 UTC
 10004.jpg |  20647 | 2024-10-17 14:07:24 UTC
 10005.jpg |  16677 | 2024-10-17 14:07:24 UTC
 10006.jpg |   2146 | 2024-10-17 14:07:24 UTC
 10007.jpg |  18895 | 2024-10-17 14:07:24 UTC
:
```

AI Knowledge Base – Meet Postgres

Storing Data (DATA, not vector embeddings) ✓

- PG tables – but also: external storage (= natural habitat of unstructured)

Preparing Data - For consumption by RAG & agents

- Bite-sizing for AI (summarization, chunking)
- Normalizing data modality (PDF extraction, OCR)
- Generate vector embeddings

Indexing & Searching Data – (vector index)

- PGVECTOR is ONE answer
- Not the only one, not necessarily the best one…

Data to **Retrievable Knowledge**

- AI Knowledge Base for semantic retrieval
- Best practice: + reranking of the retrieved results

Keeping Knowledge **Up-to-Date**

- Automated Pipelines in Postgres
  - Including real-time execution

Making Knowledge HA

- The whole nine yards of Postgres

Prep AI Data in PG – Bite-sizing – Primitives

1. Basic SQL Primitives in EDB Postgres:

- Chunking based on sentence and max chunk lengths
- Summarization with LLMs

```
SELECT * FROM aidb.chunk_text('This is one chunk. This too. And this.', '{"desired_length": 1, "max_length": 1000}');
 chunk_id |       chunk
----------+--------------------
        0 | This is one chunk.
        1 | This too.
        2 | And this.

SELECT aidb.create_model('text-nim-completions', 'nim_completions',
          '{"url":   "http://llama-3-1-8b-instruct-1xgpu-g6-predictor.default.svc.cluster.local/v1/chat/completions",
            "model": "meta/llama-3.1-8b-instruct"}');

SELECT *  FROM aidb.summarize_text('There are times when the night sky glows with bands of color. The bands may begin
as cloud shapes and then spread ..... imagined that they saw fiery dragons in the sky. Some even concluded that the
heavens were on fire.', '{"model": "text-nim-completions"}');
     summarize_text
------------------------------------------------------------------------------------------------------------------
Here's a summary:
The Aurora Borealis, also known as the Northern Lights, are colored lights that appear in the night sky as bands or
curtains of color, changing in brightness and hue. The colors can range from pale yellow to green, violet, blue, and
red. Despite observation by scientists for hundreds of years, the exact cause of the Northern Lights remains unknown.
In ancient times, people were often fearful of the spectacle, imagining it to be fiery dragons or the heavens ablaze.
```

Prep AI Data in PG – Bite-sizing – Preparers

2. Automatic Preparer Pipelines in EDB Postgres

- Example for chunking of text data on external storage:

```
SELECT * FROM aidb.list_volume_content('my_text_volume');
      file_name       | size |      last_modified
----------------------+------+-------------------------
 babysitter_story.txt |  559 | 2025-02-21 10:54:13 UTC
 chunking.txt         |  284 | 2025-02-21 10:54:13 UTC
 home_late_story.txt  |  587 | 2025-02-21 10:54:13 UTC
   :

SELECT aidb.create_volume_preparer(name => 'my_text_chunker',            operation => 'ChunkText',
                                   source_volume_name => 'my_text_volume', destination_table => 'chunked_text',
                                   destination_data_column => 'chunks',    destination_key_column => 'id',
                                   options => '{"desired_length": 1, "max_length": 100}'::JSONB

SELECT aidb.bulk_data_preparation('my_text_chunker');

SELECT * FROM chunked_text;
         id          |    chunks
---------------------+-----------------------------------------------------------------------------------
babysitter_story.txt | {"I don't want a babysitter.","I am eleven years old.","My babysitter is only....
chunking.txt         | {"Short sentence at 21.",The,purpose,of,this,function,is,to,partition,text,"data ....
home_late_story.txt  | {"It was beginning to get dark.","We were still not home yet.","I bet Dad is ....
 :
```

Preparer example for text chunking

You can set up the **same** kind of preparer **for all** following data preparation **primitives**.

Prep AI Data in PG – Normalizing Data Modality

For example: Extracting **text from PDFs with LLMs** in EDB Postgres

- PDF Parsing Primitive
- Automatic Preparer for PDFs in tables or volumes

```
SELECT pgfs.create_storage_location('pdf_bucket', 's3://noah-pdf-test', NULL,
                                    '{"region": "eu-central-1", "skip_signature": "true"}', '{}');
SELECT aidb.create_volume('pdf_volume', 'pdf_bucket', '/', 'application/pdf');

SELECT * FROM aidb.parse_pdf(aidb.read_volume_file('pdf_volume', 'text_and_images.pdf'),
                             '{"method": "Structured", "allow_partial_parsing": true}');
     parse_pdf
-----------------------------------------------------------------------------------------------------------
{"PDF  test file  Purpose: Provide example of this file type  Document file type:  PDF  Version: 1.0  Remark:
Example content:  The names \" John Doe \" for males, \" Jane Doe \" or \" Jane Roe \" for females, or \" Jonnie Doe
\" and \" Janie Doe \" for  children, or just \" Doe \" non -gender -specifically are used as  placeholder names for
a party whose true identity is un- known or must be withheld in a legal action,....
   :

SELECT aidb.create_volume_preparer(name => pdf_preparer',
    operation => 'ParsePdf', source_volume_name => 'pdf_volume',
    destination_table => 'pdf_extract_tab', destination_data_column => 'parsed_text', destination_key_column => 'id',
    options => '{ "method": "Structured" }'::JSONB ;

SELECT aidb.bulk_data_preparation(pdf_preparer');
```

Prep AI Data in PG – Normalizing Data Modality

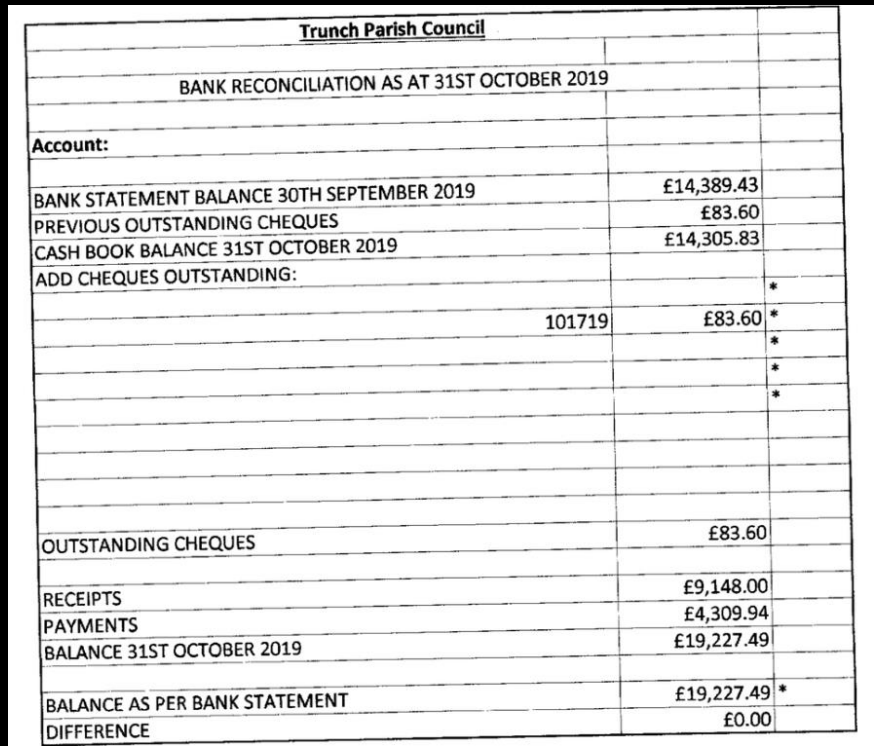
Example: Extract **text from images with OCR LLMs** in EDB Postgres

```
SELECT pgfs.create_storage_location('image_test_bucket', 's3://ocr-test', NULL,
                                    '{"region": "eu-central-1", "skip_signature": "true"}', '{}');
SELECT aidb.create_volume('ocr_volume', 'ocr_test_bucket', '/',
                          'application/png');


SELECT aidb.create_model(
       'paddle_ocr', 'nim_paddle_ocr',
       '{"url": "http://nim-baidu-paddleocr-1xgpu-g6-
       predictor.default.svc.cluster.local/v1/infer"}'::JSONB);

SELECT aidb.perform_ocr(
          aidb.read_volume_file('image_volume',
                                'ocr_bank_reconciliation.png'),
          options => '{"model": "forwarded_paddle_ocr"}' );

    perform_ocr
----------------------------------------------------------------------
{"Trunch Parish Council","BANK RECONCILIATION AS AT 31STOCTOBER
2019",Account:,"14,389.43","BANK STATEMENT BALANCE 30TH
SEPTEMBER 2019",83.60,"PREVIOUS OUTSTANDING
CHEQUES","14,305.83","CASH BOOK BALANCE 31ST OCTOBER 2019","ADD
CHEQUES OUTSTANDING:",*,101719,83.60*,*,*,83.60,"OUTSTANDING
CHEQUES","9,148.00",RECEIPTS,"4,309.94",PAYMENTS,"19,227.49","B
ALANCE 31STOCTOBER2019","19,227.49*","BALANCE AS PER BANK
STATEMENT",0.00,DIFFERENCE}
```



Trunch Parish Council

BANK RECONCILIATION AS AT 31ST OCTOBER 2019

Account:			
BANK STATEMENT BALANCE 30TH SEPTEMBER 2019		£14,389.43	
PREVIOUS OUTSTANDING CHEQUES		£83.60	
CASH BOOK BALANCE 31ST OCTOBER 2019		£14,305.83	
ADD CHEQUES OUTSTANDING:			*
	101719	£83.60	*
			*
			*
			*
OUTSTANDING CHEQUES		£83.60	
RECEIPTS		£9,148.00	
PAYMENTS		£4,309.94	
BALANCE 31ST OCTOBER 2019		£19,227.49	
BALANCE AS PER BANK STATEMENT		£19,227.49	*
DIFFERENCE		£0.00	

Prep AI Data in PG – Text Embeddings

Compute Vector **Embeddings for an individual text** in EDB Postgres:

- Single text, or array with texts:

```
SELECT aidb.create_model('text-nim-embeddings',
                         'nim_embeddings',
                         '{"url":  "http://nv-embedqa-e5-v5-1xgpu-predictor.default.svc.cluster.local/v1/embeddings",
                           "model" "nvidia/nv-embedqa-e5-v5"}');

SELECT aidb.encode_text('text-nim-embeddings', 'This is an example sentence');

      encode_text
-----------------------------------------------------------------------------------------------------------------
{-0.04385376,-0.055358887,-0.022216797,-0.0044403076,0.034606934,0.041503906,0.022109985,-0.02319336,0.035705566,-
0.004688263,0.07977295,0.013496399,0.045532227,-0.04119873,0.03805542,0.015357971,0.0043296814,0.009353638,-
0.0017747879,0.043 ....
(1 row)

SELECT aidb.encode_text_batch('text-nim-embeddings', ARRAY['The quick brown fox jumps over the lazy dog', 'Sphinx of
black quartz, judge my vow', 'How vexingly quick daft zebras jump!']);

     encode_text_batch_text
-----------------------------------------------------------------------------------------------------------------
{-0.033447266,-0.010803223,0.02633667,0.016586304,0.052642822,0.024108887,0.030563354,-0.01838684, .....
{0.018157959,-0.048614502,0.0028095245,-0.009590149,0.040527344,0.017578125,-0.017471313,-0.008155823, ....
{-0.0050201416,-0.025024414,-0.0056037903,0.033172607,0.058746338,0.005519867,-0.0036144257,-0.017990112, ....
(3 rows)
```

Prep AI Data in PG – Image Embeddings

Compute Vector **Embeddings for an image** on external volume:

```
SELECT pgfs.create_storage_location('image_test_bucket', 's3://public_ai_images', NULL,
                                    '{"region": "eu-central-1", "skip_signature": "true"}', '{}');
SELECT aidb.create_volume('image_vol', 'image_test_bucket', '/', 'Image');

SELECT aidb.create_model('my-clip',
                         'nim_clip',
                         '{"url":"http://nim-nvclip-1xgpu-g6-predictor.default.svc.cluster.local/v1/embeddings",
                           "model": "nvidia/nvclip-vit-h-14"}'::JSONB  );

SELECT aidb.encode_image('my-clip',
                          aidb.read_volume_file('image_vol','10000.jpg'));

     encode_image
-------------------------------------------------------------------------------------------------------------------------
{0.022478739,-0.012031131,-0.02170689,-0.023023805,-0.062088348,-0.002225018,-0.005005259,-0.017749019,-
0.08671932,0.029527195,-0.015034365,-0.007746646,0.018728843,-0.02170078,0.042869974,0.010556191,-
0.009796084,0.0073665897,0.008390636,0.005256727,0.019732524,0.033482935,-0.03045431,0.0014779746,-0.013152978,-
0.008432856,-0.0023803269,0.03748089,0.02944983,0.044753987,0.026918415,0.013772401,-0.009825112,-0.056826483,-
0.012649682,-0.0063284305,0.0060227807,-0.02340439,-0.0017279615,-0.086492814,-0.0044922675,0.005919139,-
0.051642597,0.0013064217,0.018946175,0.00236084,0.023778854,-0.006915893,0.011691907, ....
```

AI Knowledge Base – Meet Postgres

Storing Data (DATA, not vector embeddings) ✓

- PG tables – but also: external storage (= natural habitat of unstructured)

Preparing Data - For consumption by RAG & agents ✓

- Bite-sizing for AI (summarization, chunking)
- Normalizing data modality (PDF extraction, OCR)
- Generate vector embeddings

Indexing & Searching Data – (vector index)

- PGVECTOR is ONE answer
- Not the only one, not necessarily the best one…

Keeping Knowledge **Up-to-Date**

- Automated Pipelines in Postgres
  - Including real-time execution

Data to **Retrievable Knowledge**

- AI Knowledge Base for semantic retrieval
- Best practice: + reranking of the retrieved results

Making Knowledge HA

- The whole nine yards of Postgres

Indexing & Searching with Postgres

PGVECTOR widely known & adopted, but with built-in scale limitations

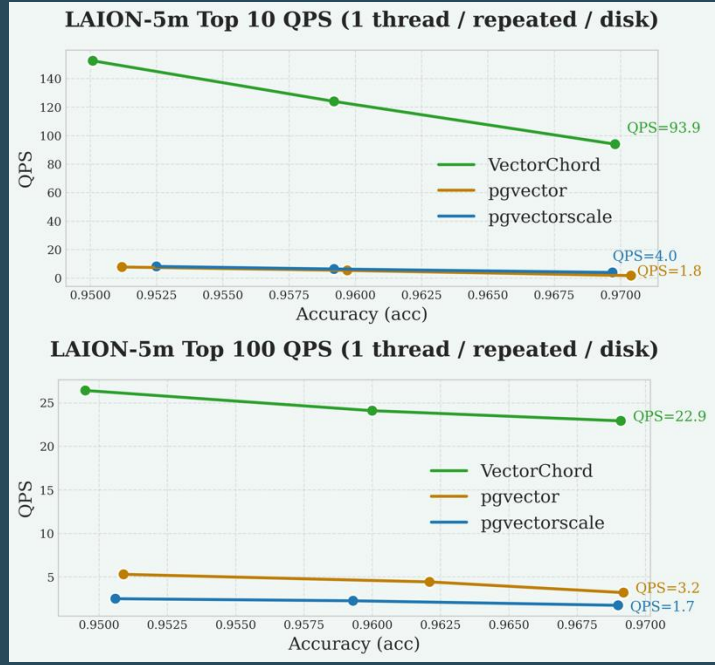
- HNSW: good recall, but build & updates expensive, very memory-sensitive for build & query, max 2000 vector dimensions
- IVF: Slow for high recall with large volumes. Recent innovations not adopted yet

Vector index alternatives for Postgres?

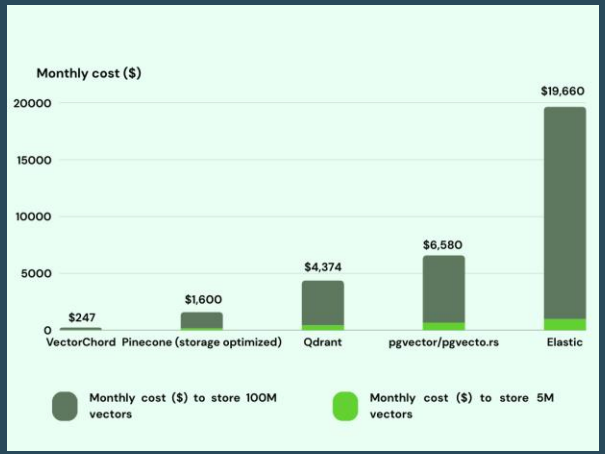
- Relevant innovations: 1. Disk-based algorithms, 2. Quantization
- Commercial only options:
  - SCaNN by Google, only in AlloyDB
  - DiskANN by Microsoft, only in Azure PostgreSQL
- Open options:
  - pgvectorscale
  - VectorChord
- All these alternatives can run on existing PGVECTOR columns

VectorChord

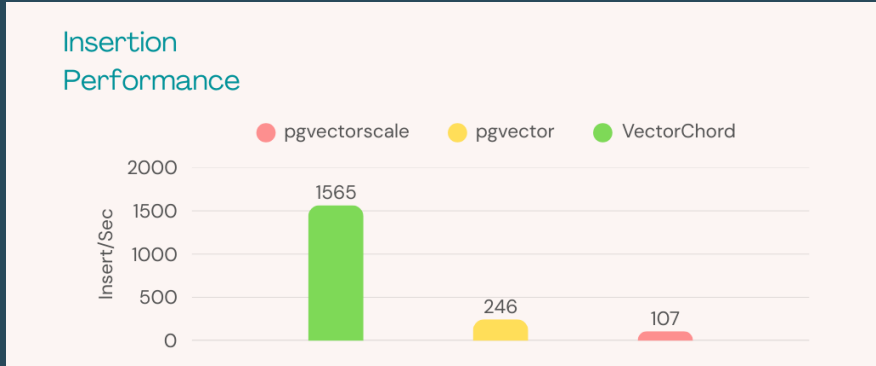
- Implements RaBitQ paper
- Binary quantization for 32x compression
  - Much **less compute** for searches
  - Up to **60K dimensions**
  - **High accuracy** with minimal reranking & **no tuning**
  - **Excellent QPS**
- Option for **index build** acceleration on **GPUs**
- **Negligible** Update/Insert/Delete **overhead**
- Market-leading **cost-performance** ratio



https://blog.vectorchord.ai/vector-search-over-postgresql-a-comparative-analysis-of-memory-and-disk-solutions



https://blog.vectorchord.ai/vector-search-at-10000-qps-in-postgresql-with-vectorchord



AI Knowledge Base – Meet Postgres

Storing Data (DATA, not vector embeddings) ✓

- PG tables – but also: external storage (= natural habitat of unstructured)

Preparing Data - For consumption by RAG & agents ✓

- Bite-sizing for AI (summarization, chunking)
- Normalizing data modality (PDF extraction, OCR)
- Generate vector embeddings

Indexing & Searching Data – (vector index) ✓

- PGVECTOR is ONE answer
- Not the only one, not necessarily the best one…

Data to **Retrievable Knowledge**

- AI Knowledge Base for semantic retrieval
- Best practice: + reranking of the retrieved results

Keeping Knowledge **Up-to-Date**

- Automated Pipelines in Postgres
  - Including real-time execution

Making Knowledge HA

- The whole nine yards of Postgres

Knowledge Retrieval – Create a Knowledge Base

Text Knowledge Base on Table Data:

```
SELECT * FROM aidb.models;
  name              |  provider      |  options
--------------------+----------------+-----------------------------------------------------------------------------
text-nim-embeddings | nim_embeddings | {"config={\"url\": \"http://nv-embedqa-e5-v5-1xgpu-predictor.default.svc....
 :

SELECT id, content FROM my_text_data;
  id   |                      content
-------+-----------------------------------------------
 43941 | Catwalk Women Brown Heels
 55018 | Lakme 3 in 1 Orchid  Aqua Shine Lip Color
 19337 | United Colors of Benetton Men Stripes Black Jacket

SELECT aidb.create_table_knowledge_base('my_text_kb', 'text-nim-embeddings', 'my_text_data', 'content', 'Text');
INFO:  using vector table: public.my_text_kb_vector
NOTICE:  auto-processing is set to "Disabled". Run "SELECT aidb.bulk_embedding('my_text_kb');" to compute embeddings.
 create_table_knowledge_base
-----------------------------
 my_text_kb

SELECT aidb.bulk_embedding('my_text_kb');
INFO:  my_text_kb: (re)setting state table to process all data...
INFO:  my_text_kb: Starting... Batch size 100, unprocessed rows: 3, count(source records): 3, count(embeddings): 0
INFO:  my_text_kb: Batch iteration finished, unprocessed rows: 0, count(source records): 3, count(embeddings): 3
INFO:  my_text_kb: finished, unprocessed rows: 0, count(source records): 3, count(embeddings): 3
```

Knowledge Retrieval – Create a Knowledge Base

Image Knowledge Base on Volume Data:

```
SELECT * FROM aidb.models;
  name   |  provider |  options
---------+-----------+-------------------------------------------------------------------------------------------------
my-clip | nim_clip  | {"config={\"url\": \"http://nim-nvclip-1xgpu-g6-predictor.default.svc.cluster.local/....
    :

SELECT * FROM aidb.list_volume_content('image_vol');
 file_name |  size  |      last_modified
-----------+--------+-------------------------
 10000.jpg |   1030 | 2024-10-17 14:07:24 UTC
    :

SELECT aidb.create_volume_knowledge_base('my_image_kb', 'my-clip', 'image_vol');
INFO:  using vector table: public.my_image_kb_vector
NOTICE:  auto-processing set to "Disabled". Run "SELECT aidb.bulk_embedding('my_image_kb');" to compute embeddings.
 create_volume_knowledge_base
------------------------------
 my_image_kb

SELECT aidb.bulk_embedding('my_image_kb');
INFO:  my_image_kb: (re)setting state table to process all data...
INFO:  my_image_kb: Starting... Batch size 100, count(source records): 0, scans completed: 0, count(embeddings): 0
INFO:  my_image_kb: Batch iteration finished, count(source records): 100, scans completed: 0, count(embeddings): 100
INFO:  my_image_kb: Batch iteration finished, count(source records): 200, scans completed: 0, count(embeddings): 200
  :
```

Knowledge Retrieval – Query a Knowledge Base

Semantic Text Retrieval:

- Retrieve text data keys
- Retrieve text data directly

```
SELECT id, content FROM my_text_data;
  id   |                      content
-------+-----------------------------------------------------
 43941 | Catwalk Women Brown Heels
 55018 | Lakme 3 in 1 Orchid  Aqua Shine Lip Color
 19337 | United Colors of Benetton Men Stripes Black Jacket

SELECT * FROM aidb.retrieve_key('my_text_kb', 'Soes for females', 2);
  key   |      distance
-------+--------------------
 43941 | 1.1791053497673203
 19337 | 1.2808311056755979
(2 rows)

SELECT * FROM aidb.retrieve_text('my_text_kb', 'Soes for females');
  key   |           value           |      distance
-------+---------------------------+--------------------
 43941 | Catwalk Women Brown Heels | 1.1791053497673203
(1 row)
```

Knowledge Retrieval – Query a Knowledge Base

Multimodal Retrieval:

- Text-2-Image Search
- Image-2-Image Search

```
SELECT * FROM aidb.retrieve_key('my_image_kb', 'white shoes', 2);
    key     |      distance
------------+--------------------
 1529.jpg   | 1.1975819603859001
 13841.jpg  |   1.20160891637948
(2 rows)

SELECT * FROM aidb.retrieve_key(' my_image_kb ', aidb.read_volume_file('image_vol','10000.jpg'), 2);
    key     |      distance
------------+--------------------
 1651.jpg   | 0.5176281886809757
 59968.jpg  | 0.5237269099660724
(2 rows)
```

Knowledge Retrieval – Query a Knowledge Base

Behind the Scenes

```
SELECT vector_table FROM aidb.knowledge_bases where name='my_image_kb';
    vector_table
--------------------
 my_image_kb_vector

SELECT SUBSTR(embeddings::text, 1, 110) FROM my_image_kb_vector;
                                                  substr
-----------------------------------------------------------------------------------------------------------
 [-0.034957077,0.05909659,0.0048654457,-0.012009802,0.04571122,-0.016535068,-0.0092142355,0.0064678527,0.047301
 [-0.0019288894,0.042462558,-0.035909444,0.0388534,0.052291196,-0.0247309,-0.018202478,-0.03987864,0.0488208,0.
   :
```

Knowledge Retrieval – Reranking

Increasing the accuracy and quality of semantic retrieval

- Get larger Top-K, revisit all values with LLM, cut off smaller reranked Top K

```
SELECT aidb.create_model(
              'my-reranker', 'nim_reranking',
              '{"url":"http://nim-nv-rerankqa-llama-l-1xgpu-g6-predictor.default.svc.cluster.local/v1/ranking",
                "model": "nvidia/llama-3.2-nv-rerankqa-1b-v2"}'::JSONB );

 create_model
--------------
 my-reranker
(1 row)

SELECT * FROM aidb.rerank_text('my-reranker',
                               'how can I open a door?',
                               '{Ask for help, Push the handle, Lie down and wait, Shout at it}'::text[])
ORDER BY logit_score DESC;

       text        | logit_score  | id
-------------------+--------------+----
 Push the handle   | -3.697265625 |  1
 Ask for help      |   -6.2578125 |  0
 Shout at it       |  -7.39453125 |  3
 Lie down and wait |      -11.375 |  2
(4 rows)
```

AI Knowledge Base – Meet Postgres

Storing Data (DATA, not vector embeddings) ✓

- PG tables – but also: external storage (= natural habitat of unstructured)

Preparing Data - For consumption by RAG & agents ✓

- Bite-sizing for AI (summarization, chunking)
- Normalizing modality (PDF extraction, OCR)
- Generate vector embeddings

Indexing & Searching Data – (vector index) ✓

- PGVECTOR is ONE answer
- Not the only one, not necessarily the best one…

Data to **Retrievable Knowledge** ✓

- AI Knowledge Base for semantic retrieval
- Best practice: + reranking of the retrieved results

Keeping Knowledge **Up-to-Date**

- Automated Pipelines in Postgres
  - Including real-time execution

Making Knowledge HA

- The whole nine yards of Postgres

S3 S3 S3

Up-to-Date Knowledge – Auto-Processing Modes

1. **Background** Embeddings
   - **Batched & parallelized** execution
   - Automatic **delta detection** & processing, including **check-pointing**
   - Perfect for AI data on external volumes
   - AI data in PG tables:
     - Perfect for initial embeddings loading on large source tables
     - No insert/update/delete overhead on source tables
   - Observability via real-time statistics table
2. **Live** Embeddings
   - Embeddings kept in sync within each transaction for the source table

Up-to-Date Knowledge – Background for Table

```
SELECT aidb.create_table_knowledge_base('large_text_kb', 'text-nim-embeddings', 'my_large_text_data', 'content', 'Text',
                                        auto_processing => 'Background');
INFO:  using vector table: public.large_text_kb_vector
INFO: large_text_kb: (Re)setting state table to process all data...
NOTICE:  auto-processing is set to "Background". Check progress with "SELECT * FROM aidb.kbstat;"


SELECT * FROM aidb.kbstat;
  knowledge base | auto processing | table: unprocessed rows | volume: scans completed | count(source records) | count(embeddings)
-----------------+-----------------+-------------------------+-------------------------+-----------------------+-------------------
 products_kb     | Background      |                   32540 |                         |                 44440 |             11900

SELECT * FROM aidb.kbstat;
  knowledge base | auto processing | table: unprocessed rows | volume: scans completed | count(source records) | count(embeddings)
-----------------+-----------------+-------------------------+-------------------------+-----------------------+-------------------
 products_kb     | Background      |                       0 |                         |                 44440 |             44440


INSERT INTO my_large_text_data VALUES (10000, 'I am a new text that is to be added to this KB');
INSERT 0 1

SELECT * FROM aidb.kbstat;
  knowledge base | auto proce                                    completed | count(source records) | count(embeddings)
-----------------+-----------                                  ----------+-----------------------+-------------------
 products_kb     | Background      |                       1 |                         |                 44441 |             44440

SELECT * FROM aidb.kbstat;
  knowledge base | auto processing | table: unprocessed rows | volume: scans completed | count(source records) | count(embeddings)
-----------------+-----------------+-------------------------+-------------------------+-----------------------+-------------------
 products_kb     | Background      |                       0 |                         |                 44441 |             44441
```

Exactly the same when **uploading** new **files** to a **Volume Knowledge Base**

Up-to-Date Knowledge – Real-Time For Table

```
SELECT aidb.create_table_knowledge_base('large_text_kb', 'text-nim-embeddings', 'my_large_text_data', 'content', 'Text',
                                        auto_processing => 'Live');

INSERT INTO my_large_text_data VALUES (1, 'I am a new text that is to be added to this KB');
INFO:  Running live embedding for knowledge_base test_kb_live. key: "1" content: "I am a new text that is to be added to this KB"
INSERT 0 1
```

AI Knowledge Base – Meet Postgres

Storing Data (DATA, not vector embeddings) ✓

- PG tables – but also: external storage (= natural habitat of unstructured)

Preparing Data - For consumption by RAG & agents ✓

- Bite-sizing for AI (summarization, chunking)
- Normalizing data modality (PDF extraction, OCR)
- Generate vector embeddings

Indexing & Searching Data – (vector index) ✓

- PGVECTOR is ONE answer
- Not the only one, not necessarily the best one…

Data to **Retrievable Knowledge** ✓

- AI Knowledge Base for semantic retrieval
- Best practice: + reranking of the retrieved results

Keeping Knowledge **Up-to-Date** ✓

- Automated Pipelines in Postgres
  - Including real-time execution

Making Knowledge HA

- The whole nine yards of Postgres

AI Knowledge Base – Meet Postgres

Storing Data (DATA, not vector embeddings) ✓

- PG tables – but also: external storage (= natural habitat of unstructured)

Preparing Data - For consumption by RAG & agents ✓

- Bite-sizing for AI (summarization, chunking)
- Normalizing data modality (PDF extraction, OCR)
- Generate vector embeddings

Indexing & Searching Data – (vector index) ✓

- PGVECTOR is ONE answer
- Not the only one, not necessarily the best one…

Data to **Retrievable Knowledge** ✓

- AI Knowledge Base for semantic retrieval
- Best practice: + reranking of the retrieved results

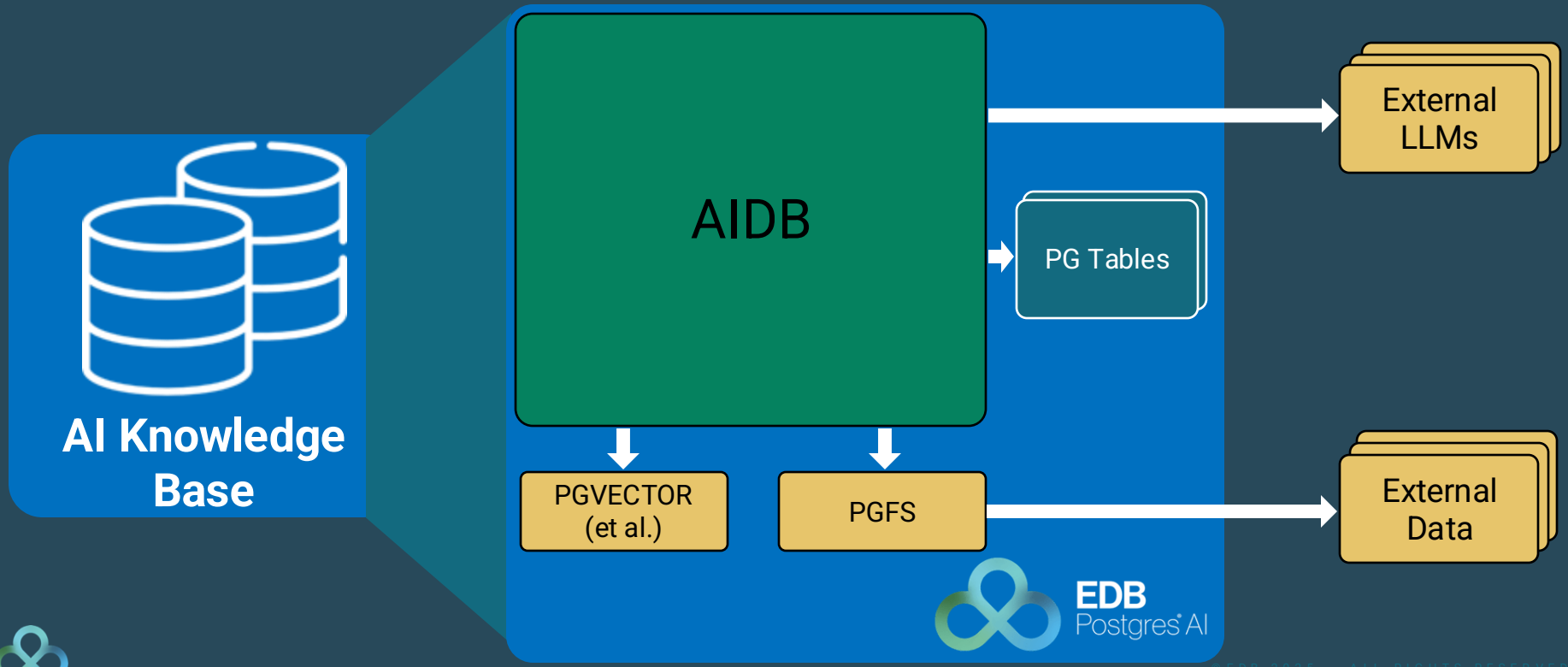
Keeping Knowledge **Up-to-Date** ✓

- Automated Pipelines in Postgres
  - Including real-time execution

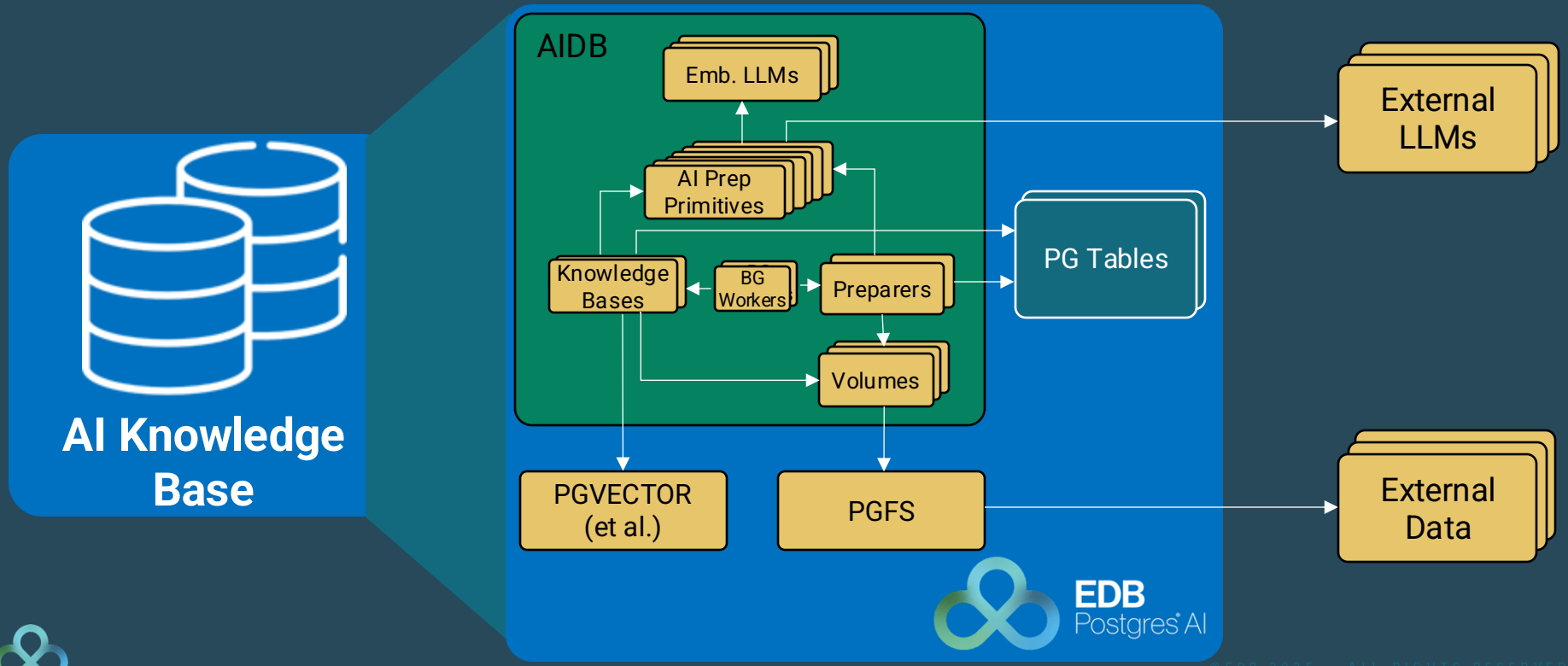
Making Knowledge HA ✓

- The whole nine yards of Postgres

AI Knowledge Base – EDB Summary



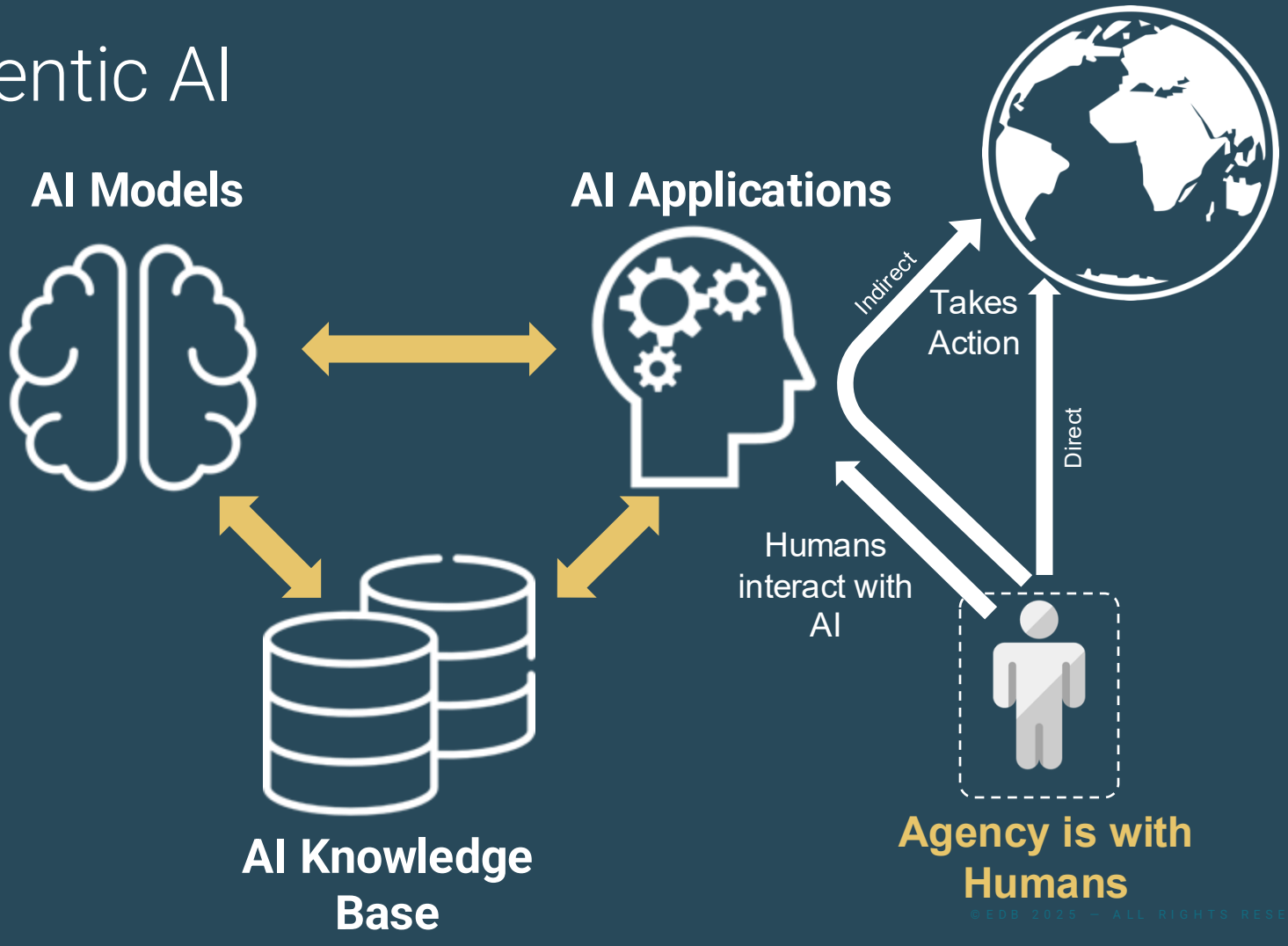
AI Knowledge Base – EDB Summary



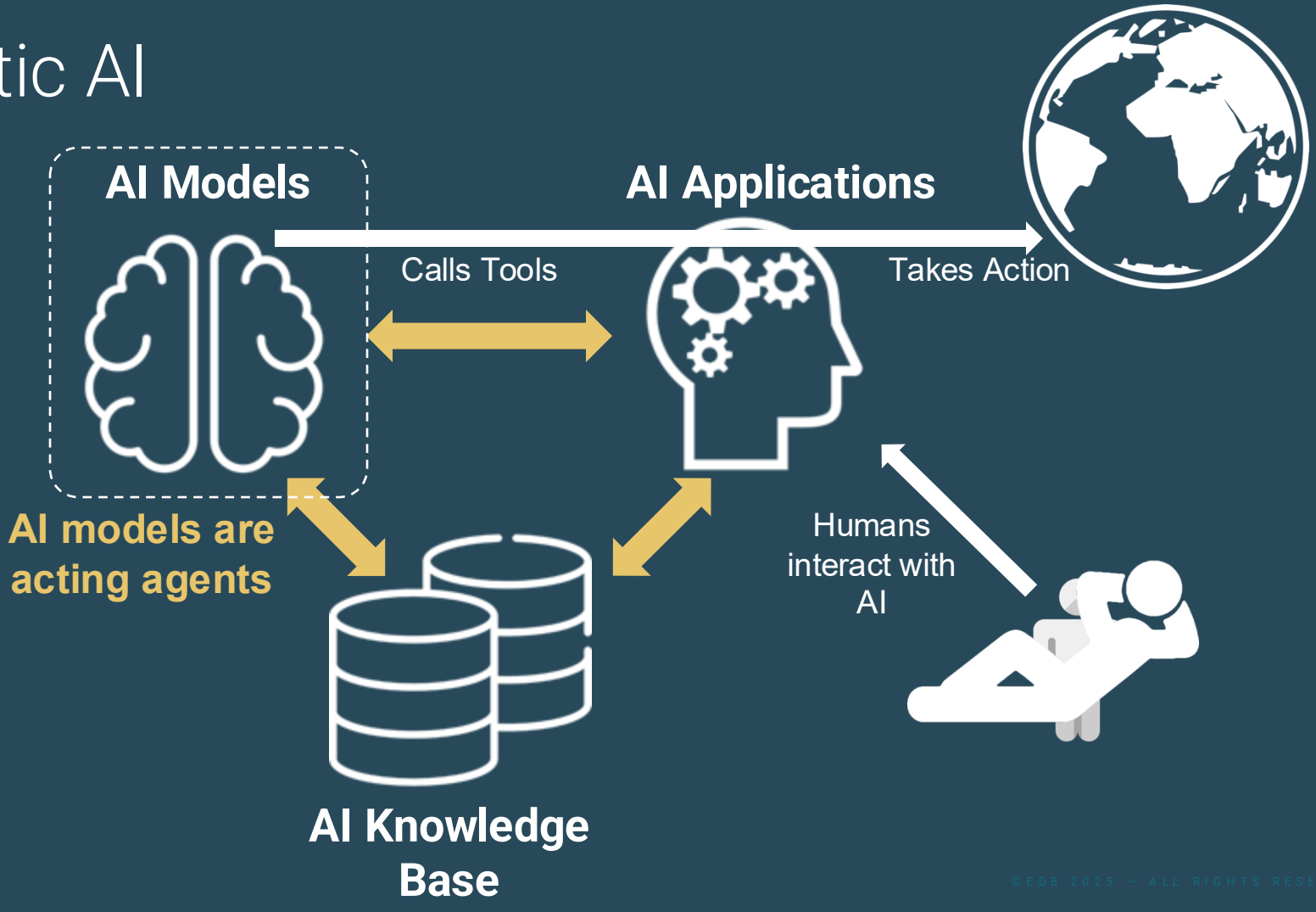
EDB
Postgres for the AI Generation

3. Agentic AI

Non-Agentic AI



Agentic AI



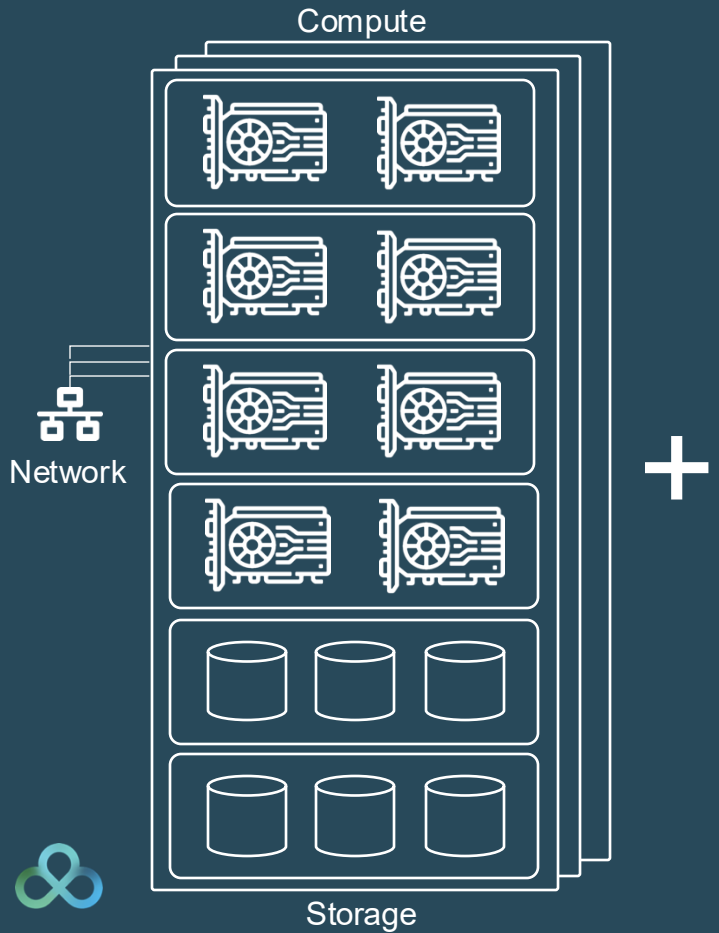
EDB
Postgres for the AI Generation

4. AI Factory

AI Factory

A specialized and **integrated AI system** with strong emphasis on **AI Inferencing** and efficiency. It comes with accordingly optimized combination of **HW**, **SW** and **Data**.

AI Factory



+

THIS is the **Software Stack of an AI Factory**

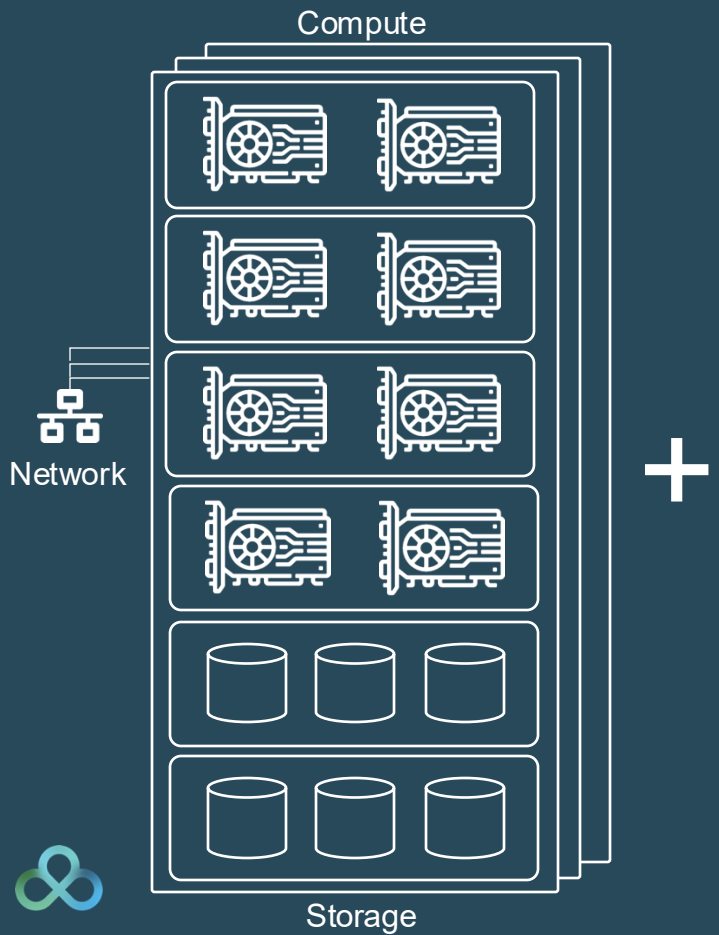
AI Models

AI Applications

AI Knowledge Base

Demo

EDB's AI Factory



+

EDB Hybrid Control Plane for Postgres

EDB Model Cluster Servers

EDB **GenAI Builder**

EDB PG Clusters with **AIDB + PGFS + PGVECTOR**

EDB
Postgres for the AI Generation

Thank You!

Slides