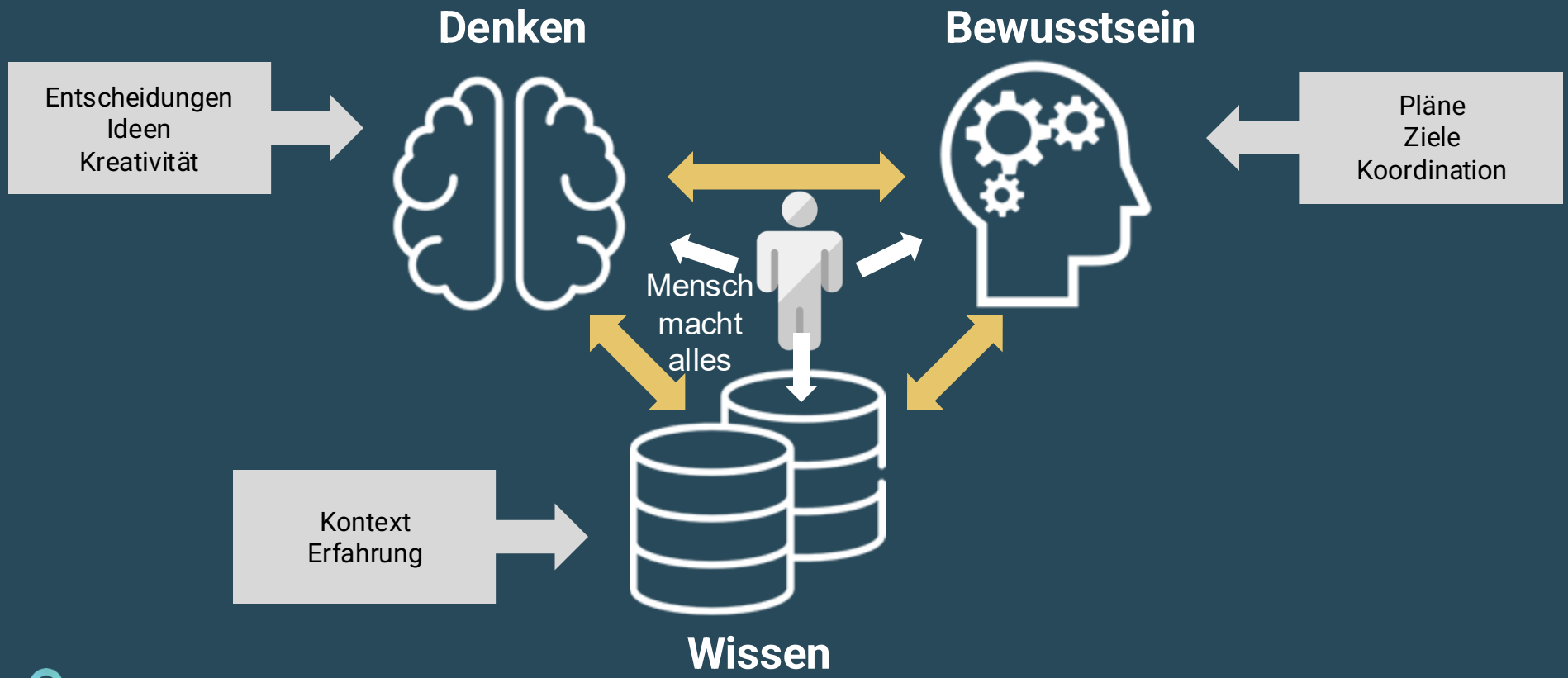
EDB POSTGRES AI

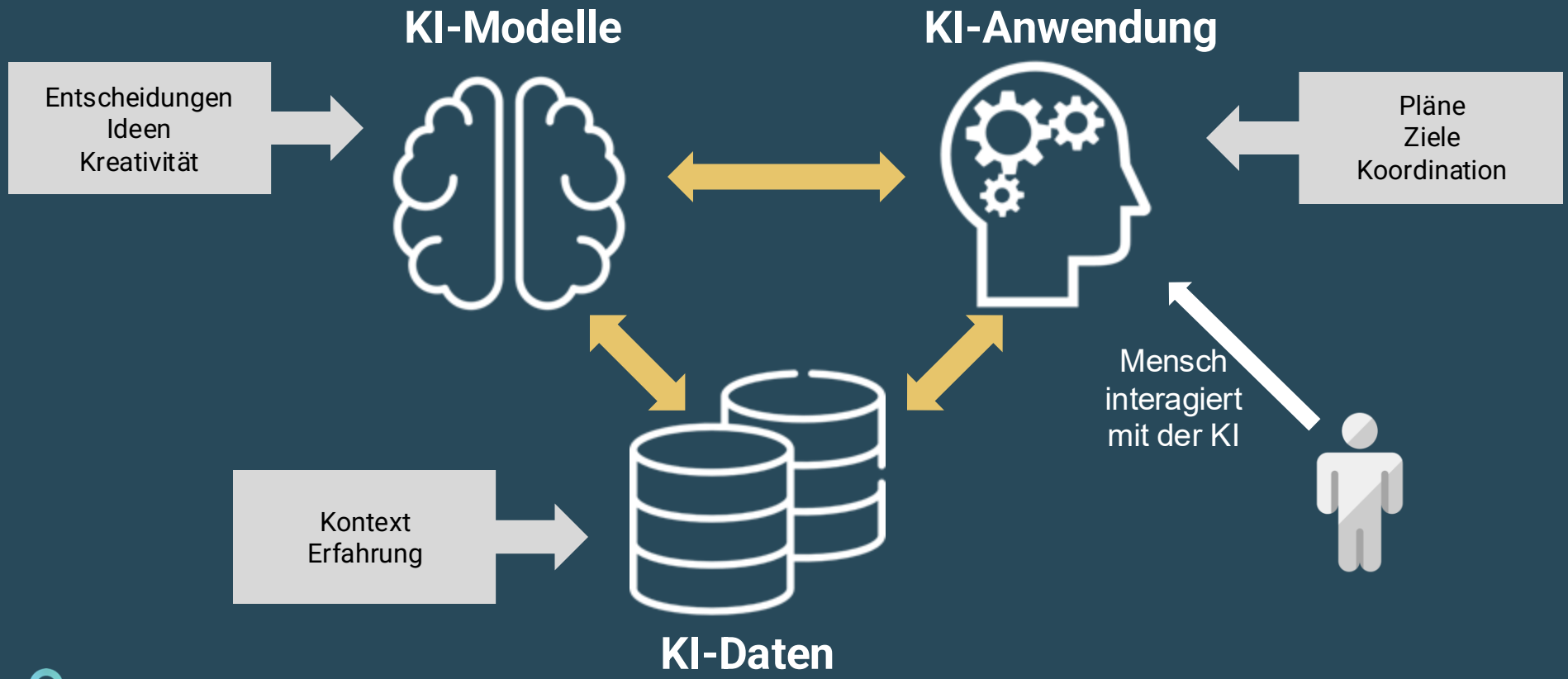
Souveräne KI mit Postgres: KI-Daten + Modelle + Anwendungen in Ihrer Infrastruktur

Torsten Steinbach
VP, Chief Architect Analytics & AI, EnterpriseDB

Menschliche Intelligenz

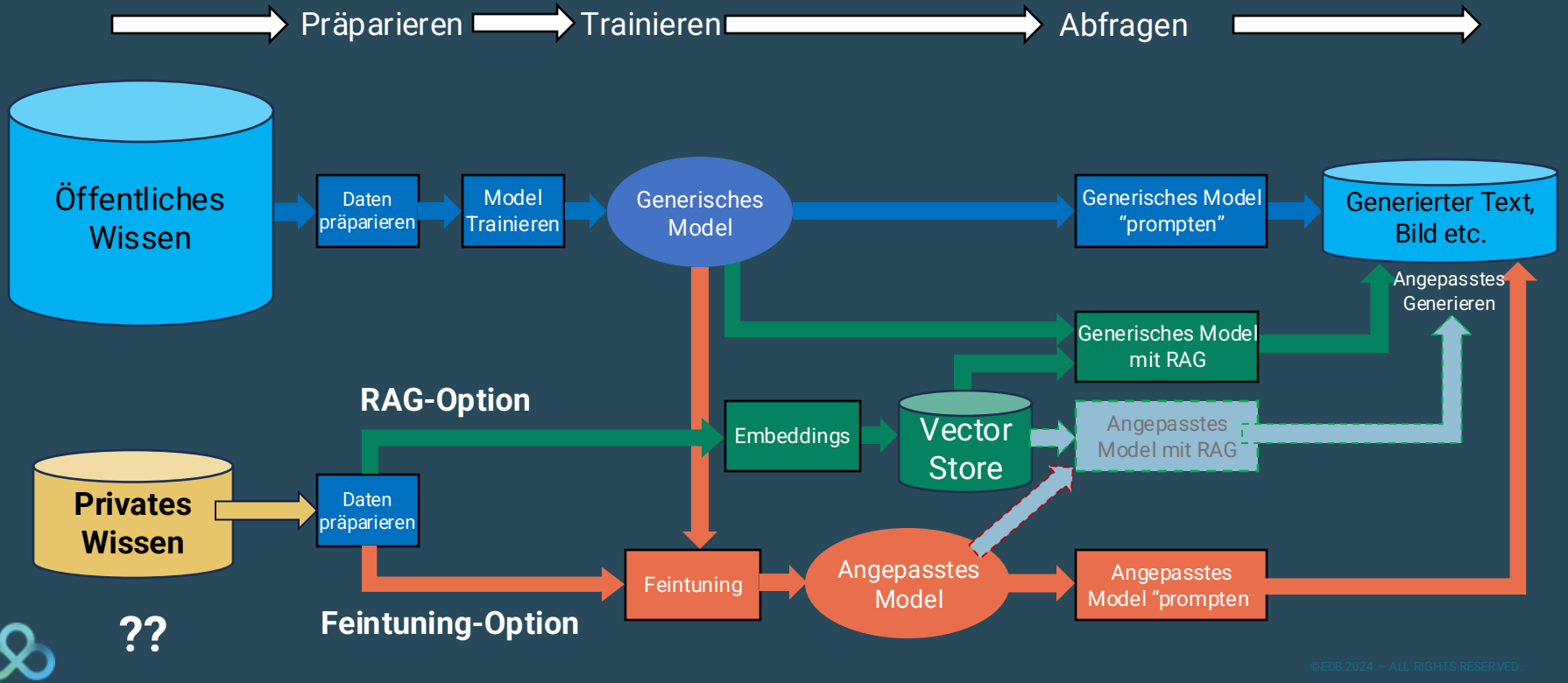


Künstliche Intelligenz



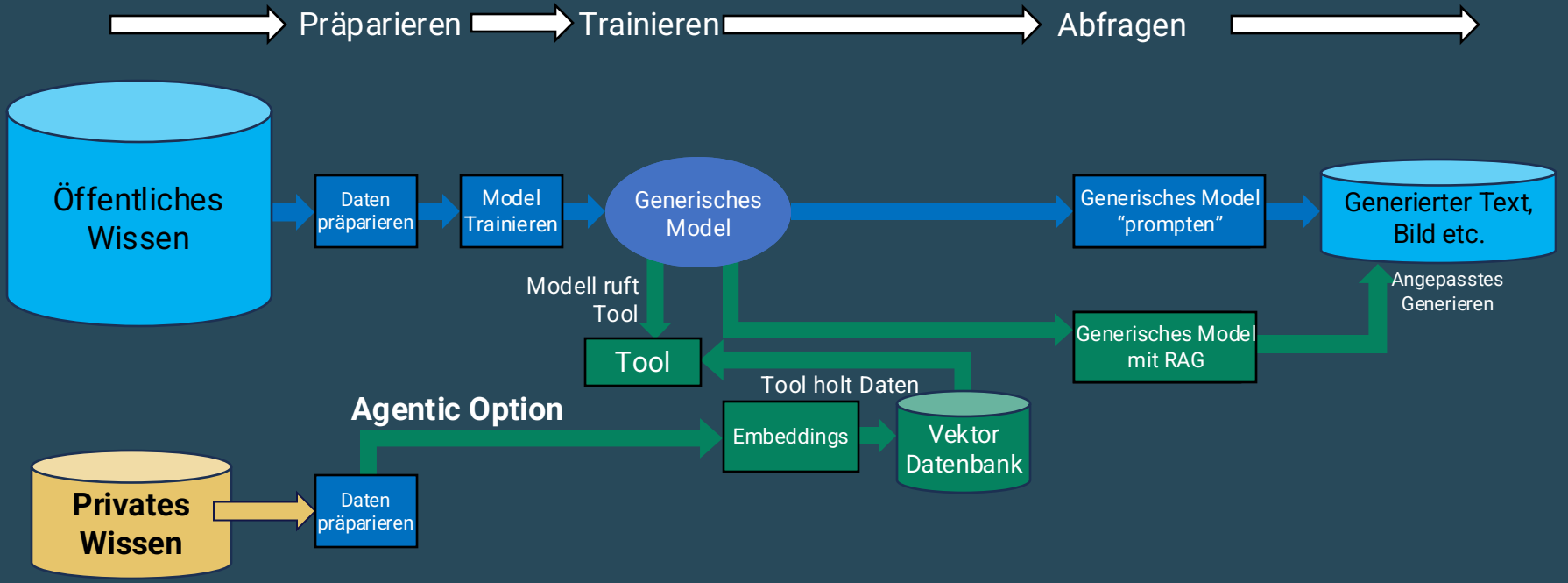
KI und Ihr Private Wissen (bzw. Daten)

Feintuning oder RAG – Front-lastig oder End-lastig

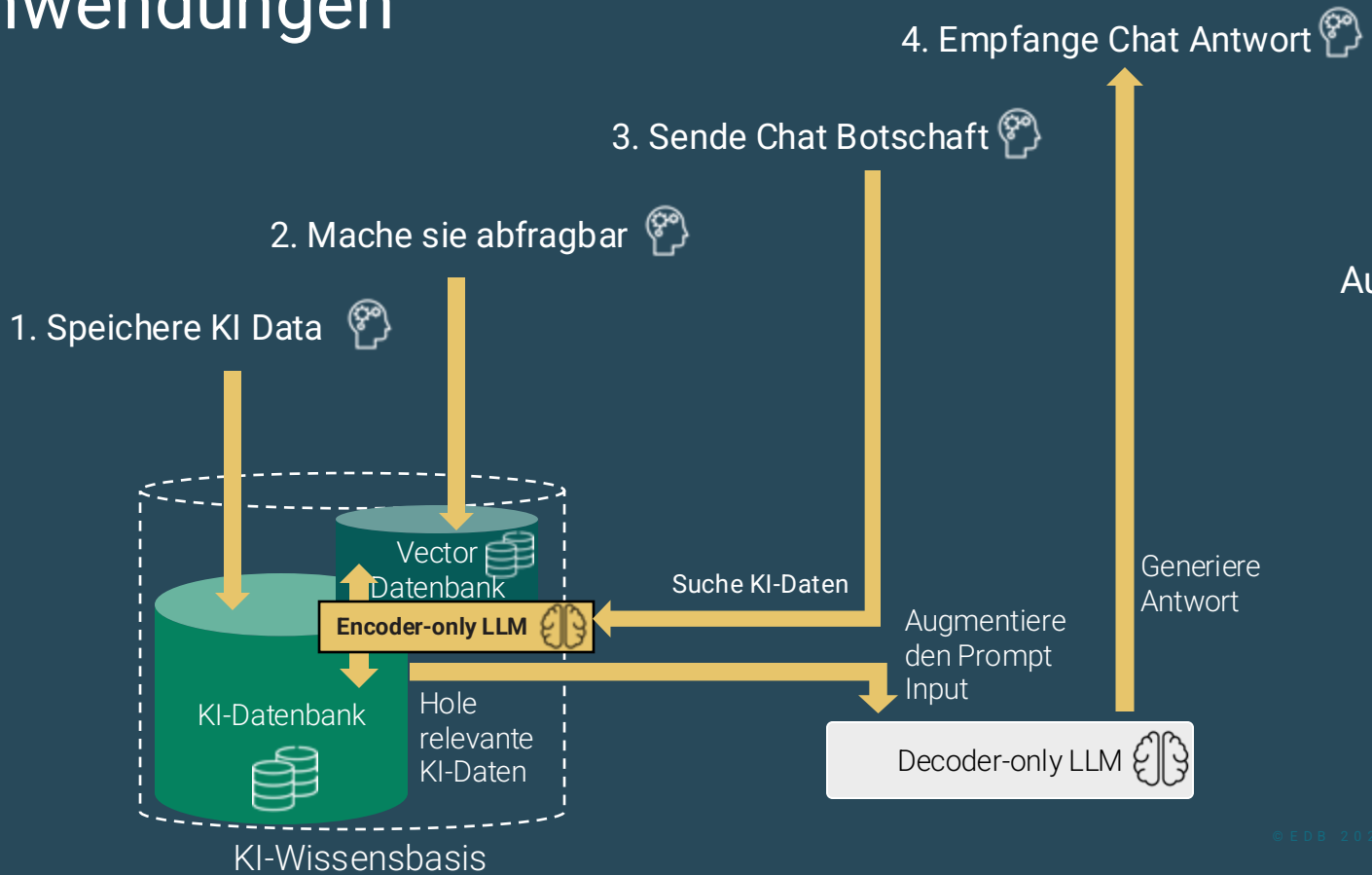


Agentische KI: Letzte Entwicklung für private Daten in der KI

End-lasting – ähnlich wie RAG



Chat Bots – Die allgegenwärtigen Generischen KI-Anwendungen



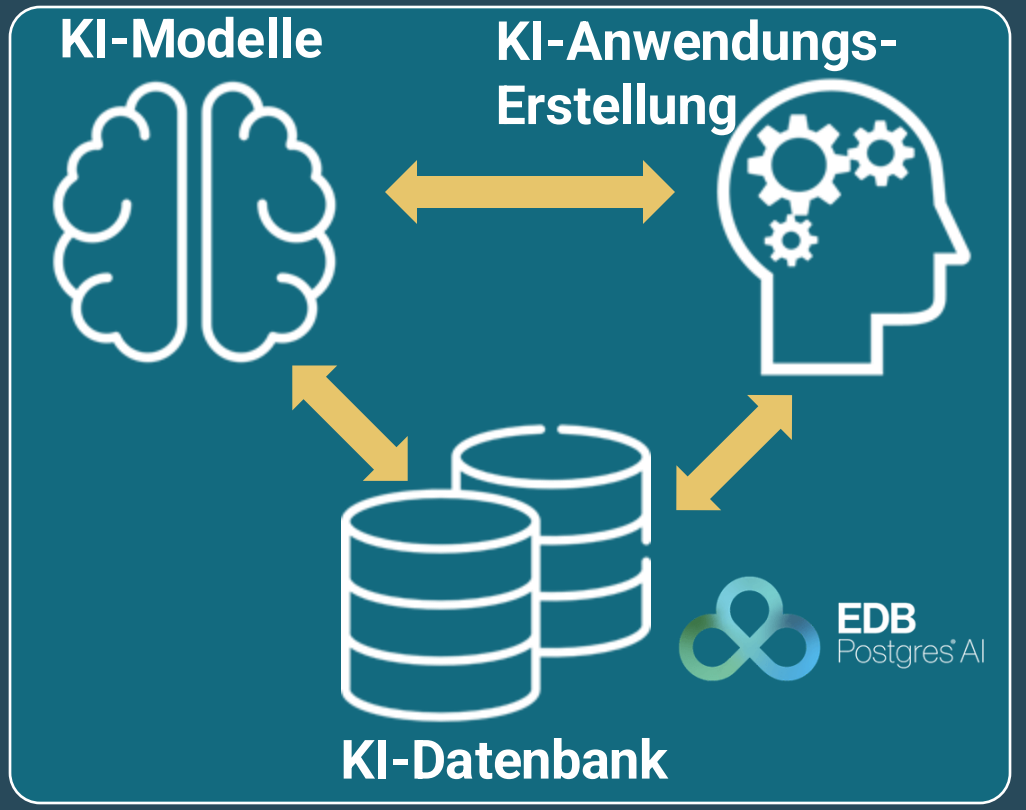
Auch bekannt als:
Retrieval
Augmented
Generation
= **RAG**

Die KI-Fabrik (AI Factory)

Ein **integriertes KI-System** mit Schwerpunkt auf effizienten und souveränen **Abfragen von KI-Modellen**, welches mit eines entsprechend **optimierten** Kombination von **HW**, **SW** und **Daten** ausgestattet ist.

Souveräne KI – EDB's KI-Softwareplattform

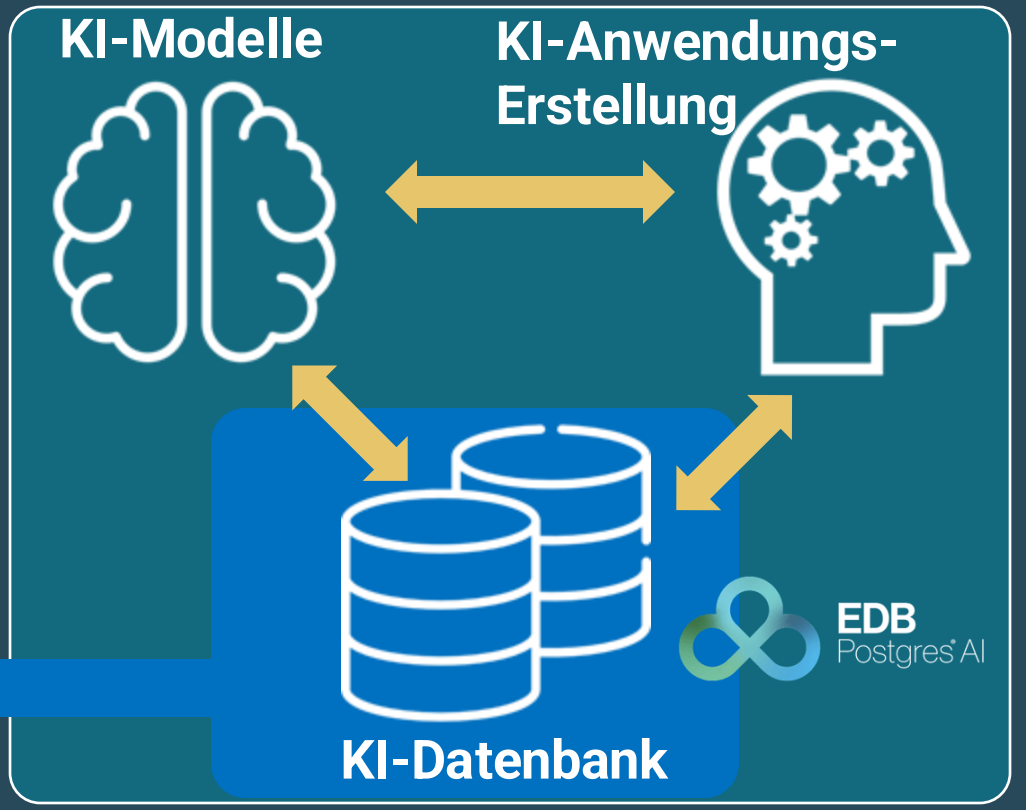
Demo



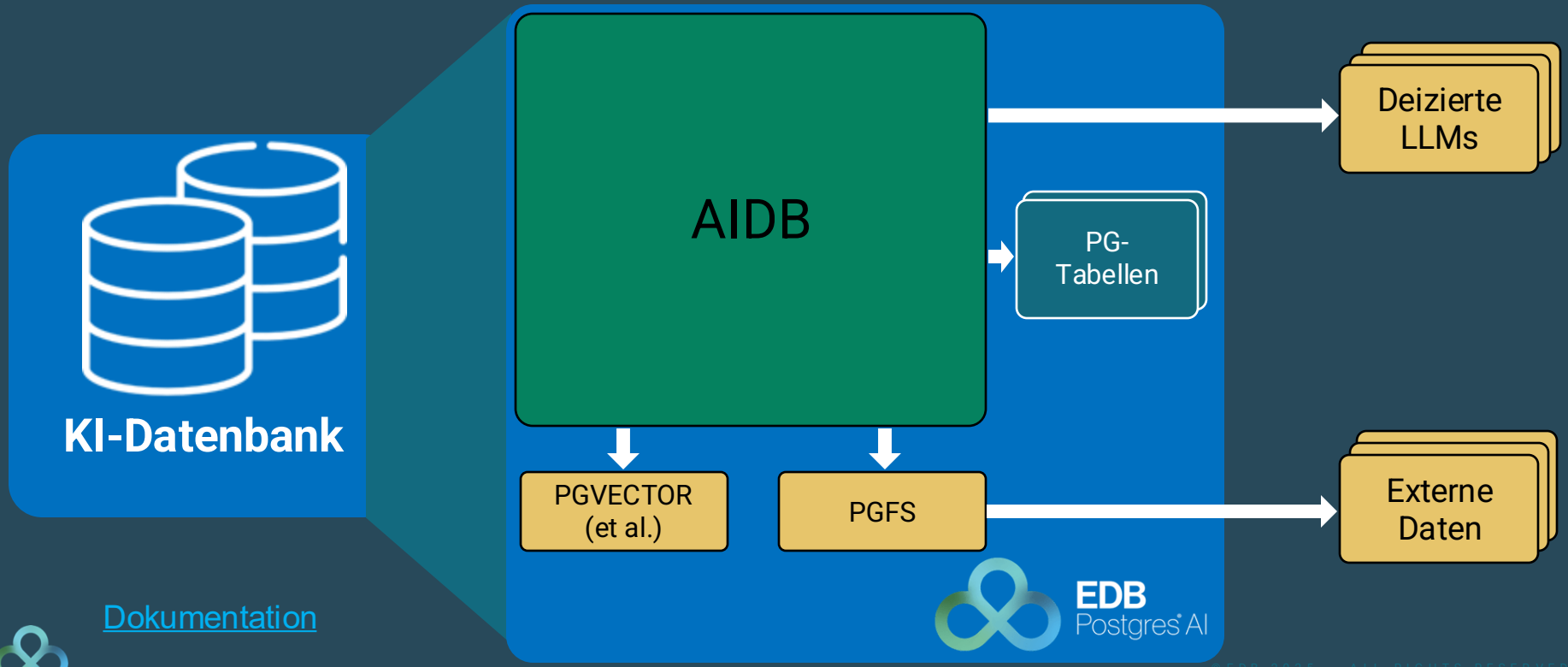
Demo

Souveräne KI – EDB's KI-Datenbank

- **Speichern** von Daten
- **Präparieren** von Data
- **Indizieren & Suchen** von Data
- **KI-Wissen abfragbar** machen
- KI-Wissen **aktuell halten**
- KI-Wissen **hochverfügbar**

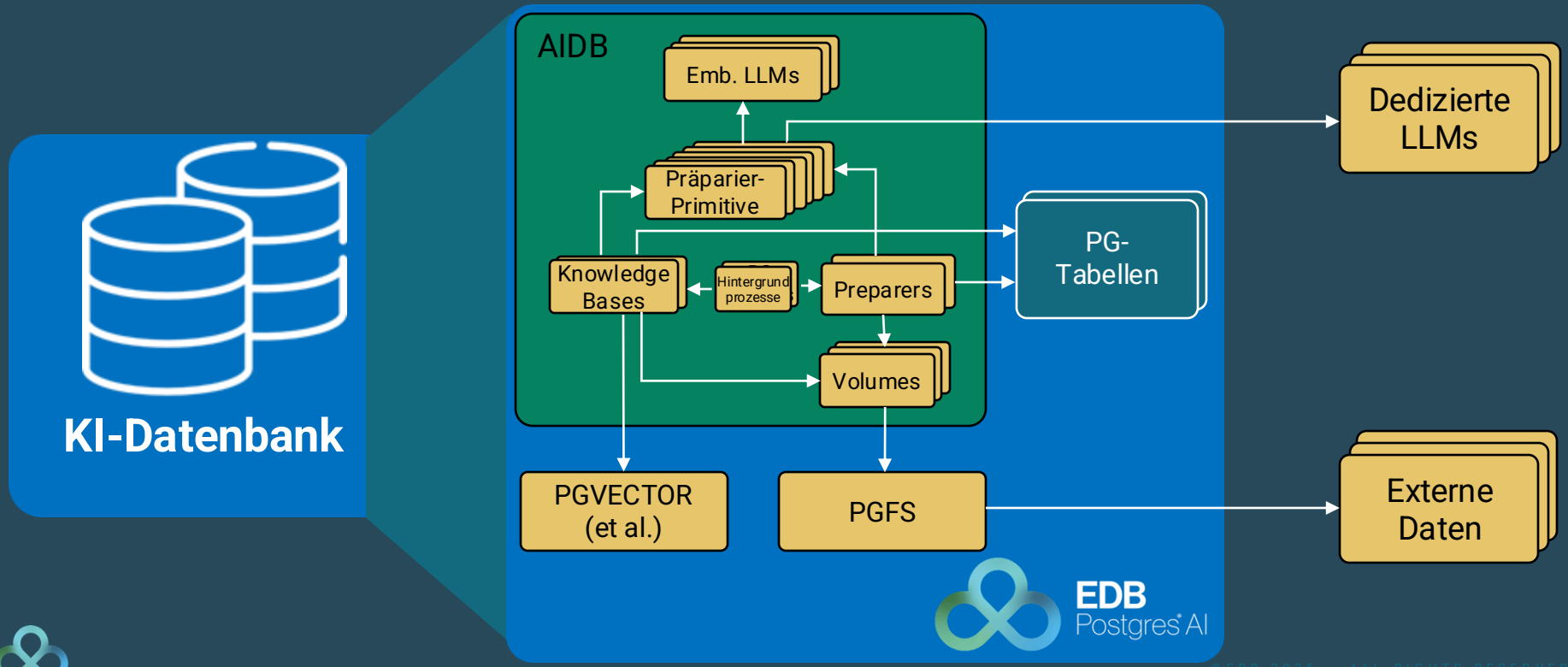


EDB’s KI-Datenbank-Stack in Postgres



Dokumentation

EDB’s KI-Datenbank-Stack in Postgres



KI-Datenbank – Hier kommt Postgres

Speichern von Daten (DATEN, nicht Vektor-Embeddings)
- PG-Tablen – UND: Externer Speicher (= typisch für unstruktierierte Daten)

Präparieren von Daten – zur Nutzung in RAG & KI-Agenten
- Mundgerechte KI-Happen (Summarization, Chunking)
- Normalisieren der Daten-Modalität (PDF-Extraktion, OCR)
- Generierung von Vektor-Embeddings

Indizieren & Suchen – (Vektorindex)
- PGVECTOR ist EINE Möglichkeit
- Aber nicht die einzige notwendigerweise beste …

KI-Wissen abfragbar machen
- KI-Wissen für semantische Abfragen
- Best Practice: + Reranking der Ergebnisse

KI-Wissen **aktuell halten**
- Automatische Pipelines in Postgres
  - Inklusive Echtzeitausführung

KI-Wissen **hochverfügbar**
- Heimspiel für Postgres

Speichern von KI-Daten in Postgres

Option 1: Reguläre PG-Tablen

Option 2: EDB Postgres – Volumes

- Dedizierte Entitäten in PG, die externe, **unstruktierierte Daten** reprästieren
  - PDFs, HTMLs, Bilder, Ton etc. – als **Mime-Types**, kein relationales Schema
  - Ideal für **große Daten-Volumen**

```
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
:
```

KI-Datenbank – Hier kommt Postgres

Speichern von Daten (DATEN, nicht Vektor-Embeddings) ✓

- PG-Tablen – UND: Externer Speicher (= typisch für unstruktierierte Daten)

Präparieren von Daten – zur Nutzung in RAG & KI-Agenten

- Mundgerechte KI-Happen (Summarization, Chunking)
- Normalisieren der Daten-Modalität (PDF-Extraktion, OCR)
- Generierung von Vektor-Embeddings

Indizieren & Suchen – (Vektorindex)

- PGVECTOR ist EINE Möglichkeit
- Aber nicht die einzige notwendigerweise beste …

KI-Wissen abfragbar machen

- KI-Wissen für semantische Abfragen
- Best Practice: + Reranking der Ergebnisse

KI-Wissen **aktuell halten**

- Automatische Pipelines in Postgres
  - Inklusive Echtzeitausführung

KI-Wissen **hochverfügbar**

- Heimspiel für Postgres

Prep – Mundgerechte KI-Happen – Primitives

1. Einfache SQL-Primitive in EDB Postgres:

- Chunking basierend auf Sätzen und maximaler Chunk-Länge
- Summarization mit 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 – Mundgerechte KI-Happen – Preparers

2. Automatische Preparer-Pipelines in EDB Postgres

- Beispiel für Chunking von Textdaten in externem Speicher:

```
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-Beispiel für Text-Chunking

Das gleiche funktioniert für alle die folgenden Primitive ebenso.

Prep – Normalisierung der Daten- Modalität

Beispiele: Text-Extraktion **von PDFs mittels LLMs** in EDB Postgres

- Primitive für PDF-Parsing
- Automatische Preparer für PDFs in Tabellen oder 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 – Normalisierung der Daten- Modalität

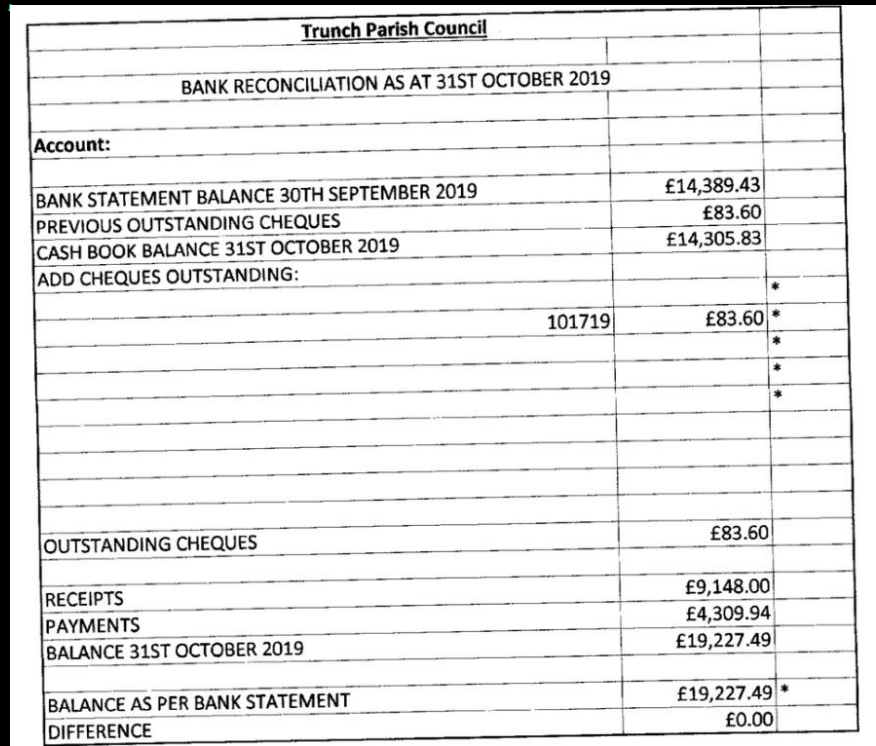
Beispiel: **Text-Extraktion aus Bildern mittels OCR LLMs** in EDB Postgres

```
SELECT pgfs.create_storage_location('image_test_bucket', 's3://edb-aidb-ocr-test-input', NULL,
                                    '{"region": "eu-central-1", "skip_signature": "true"}', '{} ');
SELECT aidb.create_volume('ocr_volume', 'image_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(
        'paddle_ocr',
        aidb.read_volume_file('ocr_volume',
                              'ocr_bank_reconciliation.png'));

    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 – Text-Embeddings

Erstellen von Vektor-**Embedding für einzelnen Text** in EDB Postgres:

- Einfacher text, oder Array mit Texten:

```
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 – Bild-Embeddings

Erstellen von Vektor-**Embeddings für ein Bild** in externem 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, ....
```

KI-Datenbank – Hier kommt Postgres

Speichern von Daten (DATEN, nicht Vektor-Embeddings) ✓

- PG-Tablen – UND: Externer Speicher (= typisch für unstruktierierte Daten)

Präparieren von Daten – zur Nutzung in RAG & KI-Agenten ✓

- Mundgerechte KI-Happen (Summarization, Chunking)
- Normalisieren der Daten-Modalität (PDF-Extraktion, OCR)
- Generierung von Vektor-Embeddings

Indizieren & Suchen – (Vektorindex)

- PGVECTOR ist EINE Möglichkeit
- Aber nicht die einzige notwendigerweise beste …

KI-Wissen abfragbar machen

- KI-Wissen für semantische Abfragen
- Best Practice: + Reranking der Ergebnisse

KI-Wissen **aktuell halten**

- Automatische Pipelines in Postgres
  - Inklusive Echtzeitausführung

KI-Wissen **hochverfügbar**

- Heimspiel für Postgres

Indizieren & Suchen mit Postgres

PGVECTOR ist **weit bekannt** und in Benutzung, aber mit eingebauten **Limitierungen**:

- HNSW: Gutes “Recall”, aber Index bauen und pflegen ist teuer, sehr Hauptspeicher-intensiv, maximal 2000 Vektor-Dimensionen
- IVF: Langsam für hohen “Recall” mit großen Datenvolumen. Neueste Innovationen noch nicht aufgegriffen

Vector-Index-Alternativen für Postgres?

- Innovationen: 1. Disk-basierte Algorithmen, 2. Quantisierung
- Kommerzielle Optionen:
  - SCaNN von Google, nur in AlloyDB
  - DiskANN von Microsoft, nur in Azure PostgreSQL
- Offene Optionen:
  - pgvectorscale
  - VectorChord
- All diese Alternativen können existierende PGVECTOR-Daten nutzen.

KI-Datenbank – Hier kommt Postgres

Speichern von Daten (DATEN, nicht Vektor-Embeddings) ✓

- PG-Tablen – UND: Externer Speicher (= typisch für unstruktierierte Daten)

Präparieren von Daten – zur Nutzung in RAG & KI-Agenten ✓

- Mundgerechte KI-Happen (Summarization, Chunking)
- Normalisieren der Daten-Modalität (PDF-Extraktion, OCR)
- Generierung von Vektor-Embeddings

Indizieren & Suchen – (Vektorindex) ✓

- PGVECTOR ist EINE Möglichkeit
- Aber nicht die einzige notwendigerweise beste …

KI-Wissen abfragbar machen

- KI-Wissen für semantische Abfragen
- Best Practice: + Reranking der Ergebnisse

KI-Wissen **aktuell halten**

- Automatische Pipelines in Postgres
  - Inklusive Echtzeitausführung

KI-Wissen **hochverfügbar**

- Heimspiel für Postgres

KI-Wissen abfragen –Knowledge Base anlegen

Text-Knowledge-Base für Tabellen-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
```

KI-Wissen abfragen –Knowledge Base anlegen

Bild-Knowledge-Base für Daten in Volumes:

```
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
  :
```

KI-Wissen abfragen – Knowledge-Base-Anfrage

Semantische Text-Anfrage:

```
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)
```

KI-Wissen abfragen – Knowledge-Base-Anfrage

Multi-Modale Abfrage:

- Text-zu-Bild-Suche
- Bild-zu-Bild-Suche

```
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)
```

KI-Wissen abfragen – Knowledge-Base-Anfrage

Hinter den Kulissen

```
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.
   :
```

KI-Wissen abfragen – Reranking

Erhöhen der Qualität von semantischen Abfragen

- Alle Treffer mit LLM neu evaluieren und sortieren

```
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)
```

KI-Datenbank – Hier kommt Postgres

Speichern von Daten (DATEN, nicht Vektor-Embeddings) ✓

- PG-Tablen – UND: Externer Speicher (= typisch für unstruktierierte Daten)

Präparieren von Daten – zur Nutzung in RAG & KI-Agenten ✓

- Mundgerechte KI-Happen (Summarization, Chunking)
- Normalisieren der Daten-Modalität (PDF-Extraktion, OCR)
- Generierung von Vektor-Embeddings

Indizieren & Suchen – (Vektorindex) ✓

- PGVECTOR ist EINE Möglichkeit
- Aber nicht die einzige notwendigerweise beste …

KI-Wissen abfragbar machen ✓

- KI-Wissen für semantische Abfragen
- Best Practice: + Reranking der Ergebnisse

KI-Wissen **aktuell halten**

- Automatische Pipelines in Postgres
  - Inklusive Echtzeitausführung

KI-Wissen **hochverfügbar**

- Heimspiel für Postgres

KI-Wissen aktuell halten – Auto-Processing

1. **Hintergrund**-Embeddings
   - **Als Batch mit parallelem Ausführen**
   - Automatische **Delta-Erkennung** & -Prozessierung, inklusive **Check-Pointing**
   - Perfekt für KI-Daten in externen Volumes
   - KI-Daten in PG-Tabellen:
     - Perfekt für initiales Embeddings-Berechnen von großen existierenden Tabellen
     - Kein Insert/Update/Delete Overhead auf den Tabellen
   - Überwachen durch Echtzeit-Statistik-Tabelle
2. **Live** Embeddings
   - Embeddings werden automatisch mit jeder Transaktion auf der Tabelle berechnet

KI-Wissen aktuell halten – Hintergrund-Prozess

```
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**

KI-Daten aktuell halten – Echtzeit-Prozess

```
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
```

KI-Datenbank – Hier kommt Postgres

Speichern von Daten (DATEN, nicht Vektor-Embeddings) ✓

- PG-Tablen – UND: Externer Speicher (= typisch für unstruktierierte Daten)

Präparieren von Daten – zur Nutzung in RAG & KI-Agenten ✓

- Mundgerechte KI-Happen (Summarization, Chunking)
- Normalisieren der Daten-Modalität (PDF-Extraktion, OCR)
- Generierung von Vektor-Embeddings

Indizieren & Suchen – (Vektorindex) ✓

- PGVECTOR ist EINE Möglichkeit
- Aber nicht die einzige notwendigerweise beste …

KI-Wissen abfragbar machen ✓

- KI-Wissen für semantische Abfragen
- Best Practice: + Reranking der Ergebnisse

KI-Wissen **aktuell halten** ✓

- Automatische Pipelines in Postgres
  - Inklusive Echtzeitausführung

KI-Wissen **hochverfügbar**

- Heimspiel für Postgres

KI-Datenbank – Hier kommt Postgres

Speichern von Daten (DATEN, nicht Vektor-Embeddings) ✓

- PG-Tablen – UND: Externer Speicher (= typisch für unstruktierierte Daten)

Präparieren von Daten – zur Nutzung in RAG & KI-Agenten ✓

- Mundgerechte KI-Happen (Summarization, Chunking)
- Normalisieren der Daten-Modalität (PDF-Extraktion, OCR)
- Generierung von Vektor-Embeddings

Indizieren & Suchen – (Vektorindex) ✓

- PGVECTOR ist EINE Möglichkeit
- Aber nicht die einzige notwendigerweise beste …

KI-Wissen abfragbar machen ✓

- KI-Wissen für semantische Abfragen
- Best Practice: + Reranking der Ergebnisse

KI-Wissen **aktuell halten** ✓

- Automatische Pipelines in Postgres
  - Inklusive Echtzeitausführung

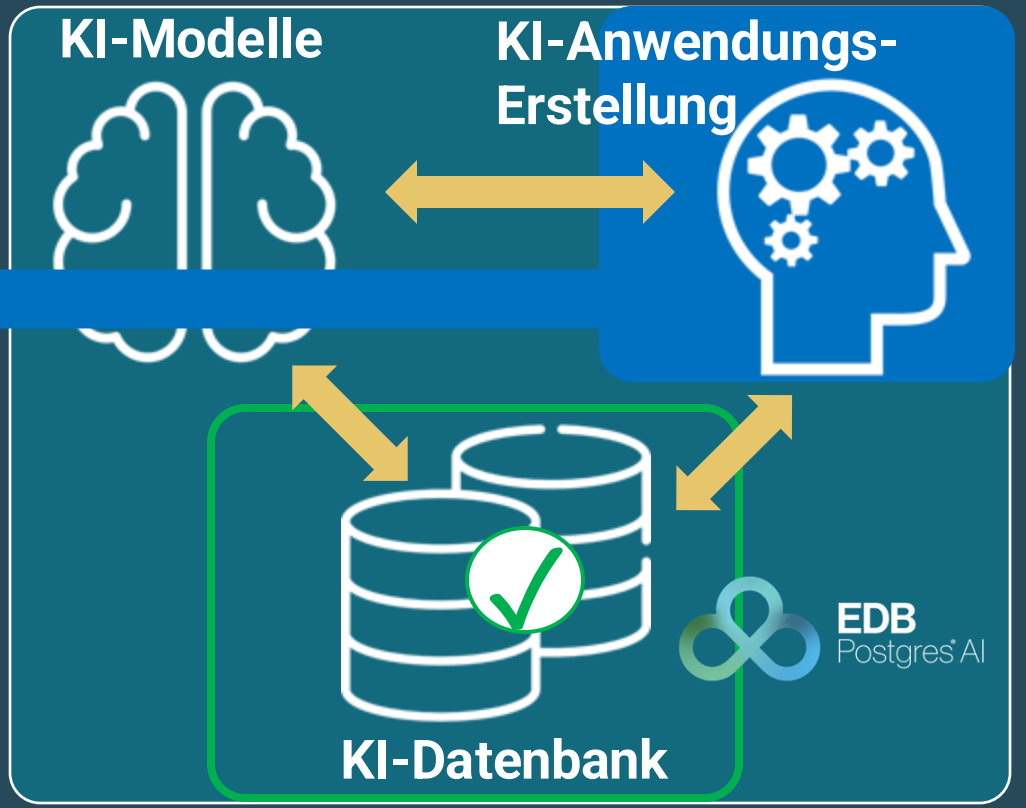
KI-Wissen **hochverfügbar** ✓

- Heimspiel für Postgres

Demo

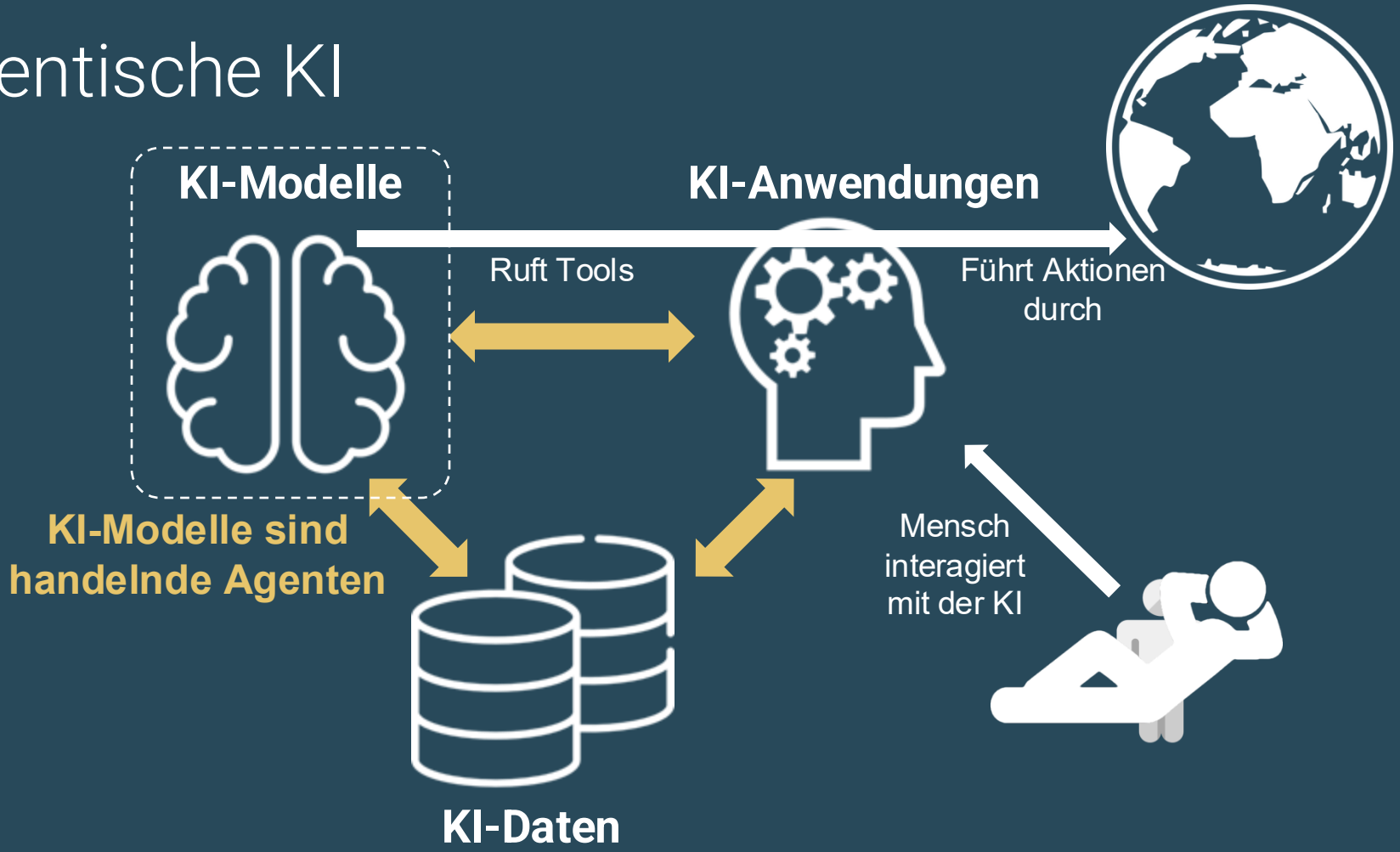
Souveräne KI – EDB's KI-Anwendungserstellung

- KI-Assistenten erstellen
- KI-Entwicklung beschleunigen
- KI-Datenbank nutzen
- Agentische KI



Nicht-Agentische KI

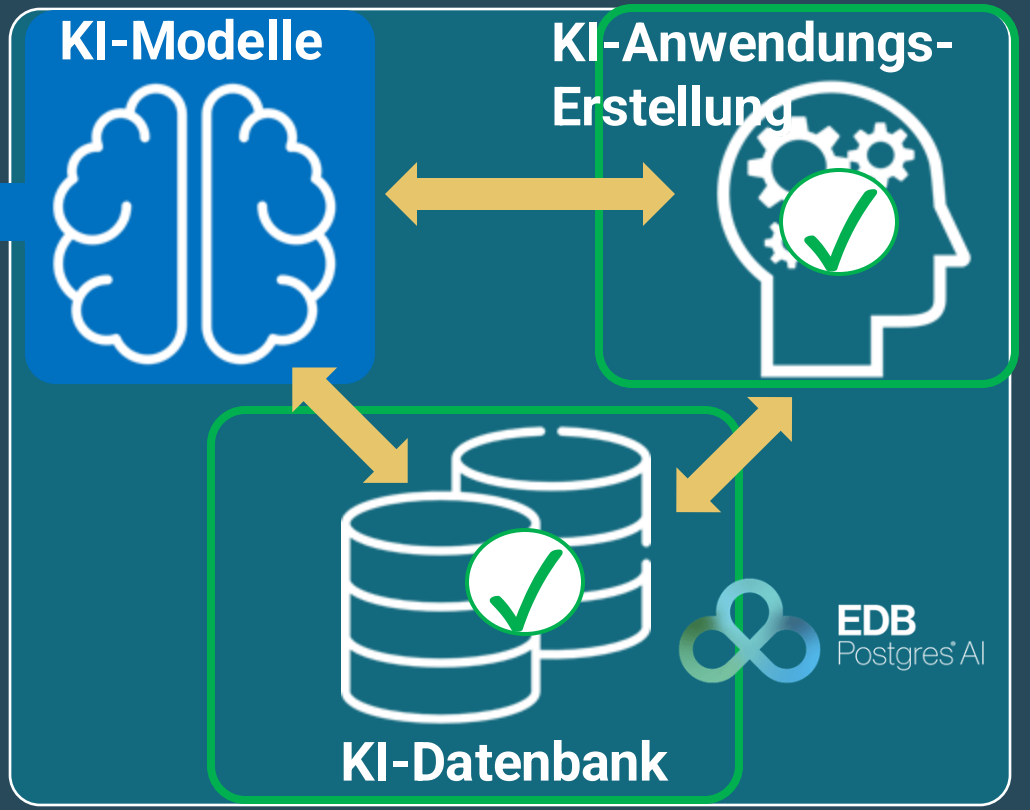
Agentische KI



Demo

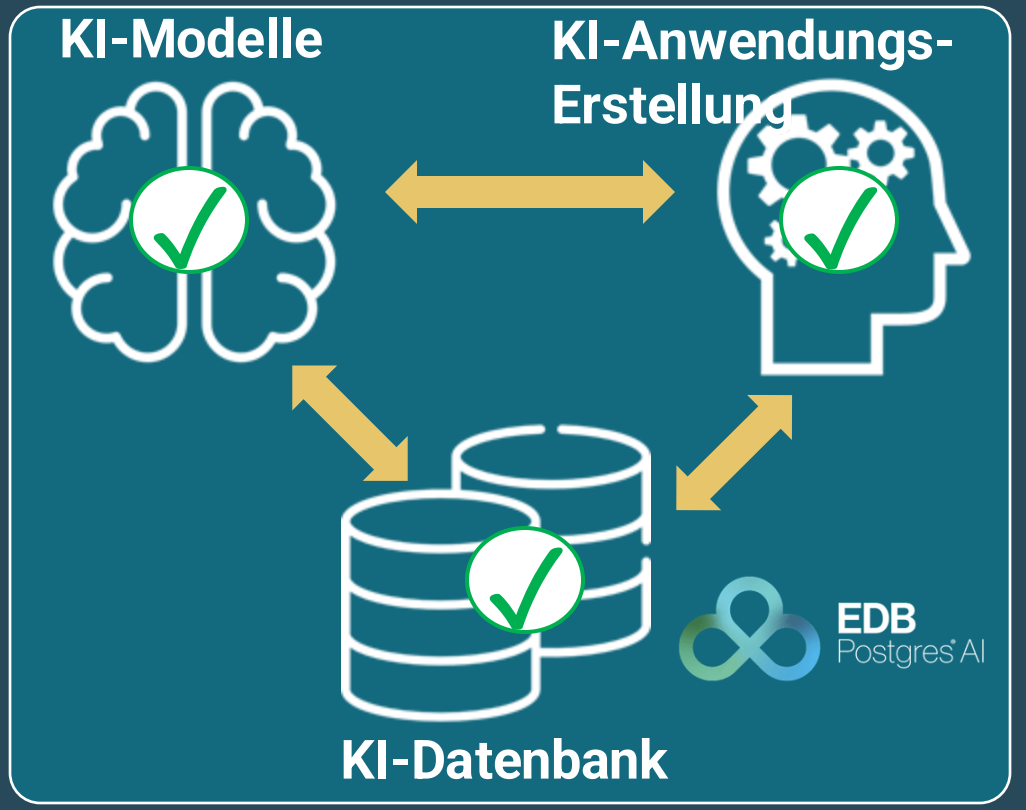
Souveräne KI – EDB's KI-Model-Serving

- **Installieren & Skalieren von LLMs**
- **OpenAI-kompatibles Inferencing** (mittels kserve)
- **GPUs ausnutzen**
- **KI-Datenbanken unterstützen**
- **KI-Anwendungen unterstützen**



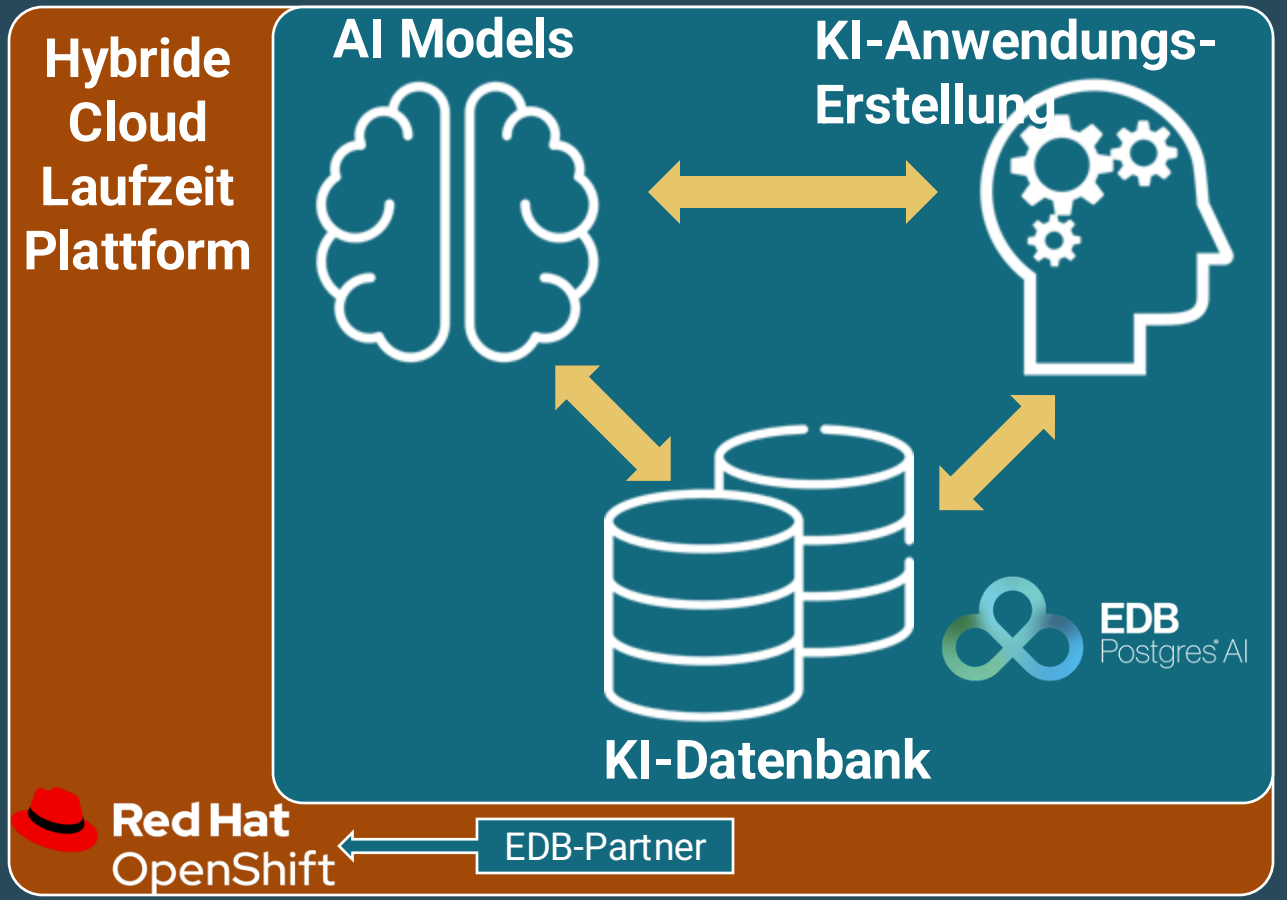
Souveräne KI – EDB's KI-Softwareplattform

Demo



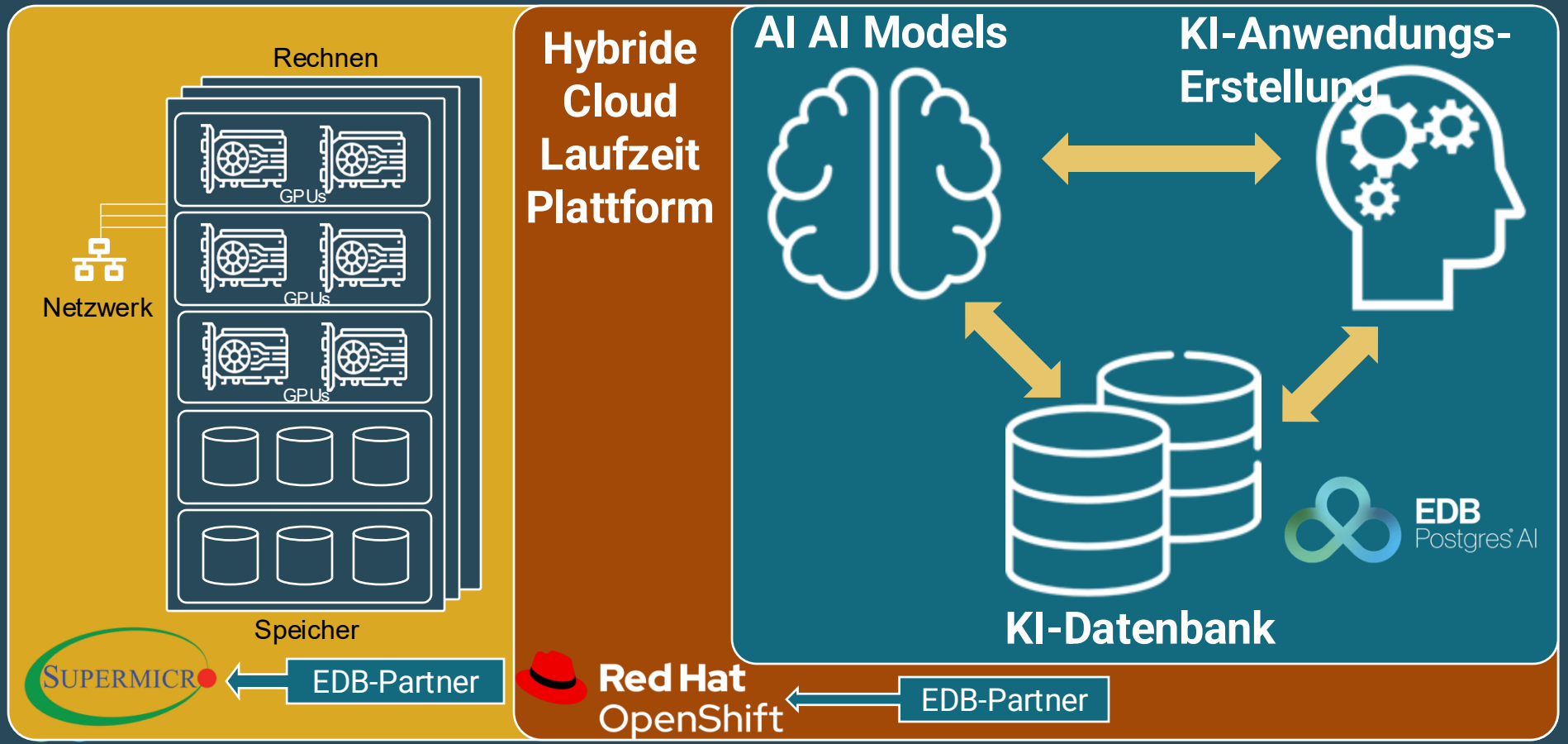
Demo

Souveräne KI – Egal Wo – Mit OpenShift



Demo

Souveräne KI – Systeme – With HW



Schauen sie unser our Launch-Event vom

June 17, 2025

Meet the Future of Postgres® and AI