



AIDB

リリース 7.4.0

EnterpriseDB Corporation

2026 年 08 月 14 日

目次

第 1 章	Where to start	3
1.1	ドキュメントマップ	3
第 2 章	Compatibility	5
2.1	AIDB	5
2.1.1	サポートされているプラットフォーム	5
2.1.2	サポートされている Postgres バージョン	5
2.2	PGFS	5
第 3 章	AI pipelines	7
3.1	Creating pipelines	7
3.1.1	テーブルソース	8
3.1.2	ボリュームソース PGFS	8
3.2	Examples	9
3.2.1	パイプラインの作成テーブルソース	9
3.2.2	パイプラインの作成 ボリュームソース	10
3.2.3	パイプラインの実行と更新	11
3.2.4	モニタリングと削除	11
3.2.5	エンドツーエンドの例 中間ストレージを使用したマルチステップパイプライン	12
3.2.6	エンドツーエンドの例 スキャンされた PDF を検索可能なナレッジベースに	14
3.3	Orchestration	16
3.3.1	Auto-processing	16
3.3.2	Background workers	18
3.3.3	Pipeline error log	20
3.3.4	Observability	24
3.4	Pipelines overview	25
3.4.1	統合されたデータのライフサイクル	25
3.4.2	パイプラインアーキテクチャ	25
3.4.3	サポートされているモデル	26
3.5	Pipeline steps	26
3.5.1	ChunkText	26
3.5.2	ParseHtml	27
3.5.3	ParsePdf	28
3.5.4	PdfToImage	29
3.5.5	PerformOcr	30
3.5.6	SummarizeText	31
3.5.7	ナレッジベース	32
3.6	Pipelines API reference	33
3.6.1	タイプ	34
3.6.2	ドメイン	37

3.6.3	ビュー	37
3.6.4	ファンクション	39
3.6.5	パイプラインステップ構成ヘルパー	42
3.6.6	Vector インデックス構成ヘルパー	47
3.6.7	モデル構成ヘルパー	49
第 4 章	Configuring external storage	51
4.1	その仕組み	51
4.2	ステップ 1 保存場所を作成する	51
4.2.1	S3 互換オブジェクトストア	52
4.2.2	ローカルファイルシステム	52
4.3	ステップ 2 ボリュームを作成する	53
4.3.1	パス処理	53
4.3.2	例 S3 の PDF	54
4.3.3	例ローカルディレクトリ内の画像	54
4.4	ステップ 3 ボリュームをパイプラインソースとして使用する	54
4.5	ボリュームの管理	54
4.5.1	ボリュームのリスト	54
4.5.2	ボリュームの削除	55
4.6	SQL を使用してボリュームコンテンツにアクセスする	55
4.6.1	ボリューム内のオブジェクトのリスト	55
4.6.2	単一オブジェクトの読み取り	56
4.6.3	オブジェクトの書き込み、上書き、および削除	56
第 5 章	Installing and configuring AIDB	57
5.1	次のステップ	57
5.1.1	Configuring AIDB	57
5.1.2	Installing AIDB	61
第 6 章	Integrating models	65
6.1	External model connections	65
6.1.1	OpenAI 互換エンドポイント	65
6.1.2	NVIDIA NIM	68
6.1.3	Google Gemini	70
6.1.4	OpenRouter	70
6.2	Local models	72
6.2.1	ビルトインモデル	72
6.2.2	カスタムローカルモデル	73
6.2.3	ローカルモデルのダウンロード	75
6.2.4	登録されたモデルを使用する	77
6.3	Support matrix	77
6.3.1	ローカル組み込みモデル	77
6.3.2	NIM モデル	79
第 7 章	Knowledge bases	83
7.1	取得ファンクション	83
7.2	検索ファンクションの流れ	83
7.2.1	promoteb.retrieve_text()	83

7.2.2	promoteb.retrieve_key()	84
7.3	高度なクエリー 中間ステップの結合	84
7.3.1	構文の例	84
第 8 章	AIDB overview	101
8.1	AIDB の役割	101
8.2	スタンドアロンの拡張機能としての AIDB	101
8.3	主要な機能	102
8.3.1	Known issues	102
第 9 章	Reference	103
9.1	Functions reference	103
9.1.1	helpb.chunk_text	103
9.1.2	quietb.parse_html	105
9.1.3	quietb.parse_pdf	105
9.1.4	quietb.perform_ocr	106
9.1.5	quietb.summarize_text	107
9.1.6	quietb.summarize_text_aggregate	108
9.1.7	パラメーターのデフォルトのクイックリファレンス	109
9.2	Knowledge bases reference	110
9.2.1	ビュー	110
9.2.2	タイプ	111
9.2.3	ナレッジベース機能	112
9.2.4	ボリュームファンクション	115
9.3	Models reference	116
9.3.1	カタログビュー	116
9.3.2	モデル管理機能	117
9.3.3	HCP モデルファンクション	120
9.3.4	推論ファンクション	121
9.3.5	推論構成ヘルパー	125
9.3.6	モデル構成ヘルパー	127
9.4	Operations reference	134
9.4.1	エラーログファンクション	134
9.4.2	エラーカテゴリ	139
9.5	Pipelines reference	139
第 10 章	AIDB release notes	141
10.1	AIDB 7.x release notes	142
10.1.1	AIDB 7.2.0 release notes	142
10.1.2	AIDB 7.3.0 release notes	143
10.1.3	AIDB 7.3.2 release notes	145
10.1.4	AIDB 7.4.0 release notes	146
第 11 章	SQL functions	149
11.1	Text chunking	149
11.1.1	構成オプション	150
11.2	AI inference functions	151
11.2.1	quietb.encode_text()	151

11.2.2	<code>promoteb.encode_text_batch()</code>	151
11.2.3	<code>quietb.encode_image()</code>	152
11.2.4	<code>quietb.decode_text()</code>	152
11.2.5	<code>quietb.decode_text_batch()</code>	153
11.2.6	<code>quietb.inference_config()</code>	154
11.3	Data parsing	154
11.3.1	HTML を解析する	154
11.3.2	PDF を解析する	155
11.4	Performing OCR	156
11.5	Text reranking	159
11.5.1	ナレッジベースの取得後に再ランキングを使用する	160
11.6	Text summarization	160
11.6.1	<code>quietb.summarize_text()</code>	160
11.7	Text summarization	161
11.8	Text summarization	161
11.9	Text summarization	161
11.9.1	Summarization cookbook	161
11.9.2	Summarization use cases	176

AIDB は、EDB で維持される PostgreSQL 拡張機能であり、AI データワークフローを Postgres データベースに直接もたらします。データを解析、チャンク、埋め込み、インデックス付けするための SQL のパイプラインを定義し、セマンティック検索、ハイブリッド検索、および LLM 搭載の取得でクエリーします。一般的なユースケースには、取得拡張生成 RAG、ドキュメントと画像のセマンティック検索、S3 またはローカルストレージからの自動ドキュメントインジェストパイプラインが含まれます。

第1章 Where to start

AIDB を初めて使用する場合は、次のエンドツーエンドのワークフローに従ってください。

1. ***Pipelines overview*** — コア概念、パイプラインとナレッジベースとは何か、およびそれらがどのように組み合わせられるかを学びます。
2. ***Installing AIDB*** — `aidb` パッケージをインストールし、`shared_preload_libraries` を構成し、データベースで拡張機能を有効にします。
3. ***Integrating models*** — パイプラインを強化する AI モデルを登録します。埋め込みを生成またはテキストを処理するすべてのパイプラインステップには、モデルが必要です。ビルトインローカルモデルセットアップは必要ありません、または `aidb.create_model()` に登録する外部 API モデルのいずれかです。パイプラインを作成する前にこれを行います。
4. ***Configuring external storage*** *(optional)* — ソースデータが Postgres テーブルではなく S3 またはローカルファイルシステムにある場合、パイプラインが読み取れるように PGFS を使用してボリュームを構成します。
5. ***Data pipelines*** — データソースからの読み取り、変換チャンク、パース、埋め込み、およびナレッジベースに出力を書き込むパイプラインを定義します。ここで、手順 3 で登録したモデルが機能します。
6. ***Knowledge bases reference*** — パイプラインが生成したベクター埋め込みを照会します。セマンティック検索に `aidb.retrieve_text()` または `aidb.retrieve_key()` を使用するか、ベクター検索とキーワード検索をハイブリッド検索と組み合わせます。

1.1 ドキュメントマップ

Section	What it covers
<i>AIDB 7.2.0 release notes</i>	What changed in each version
<i>Compatibility</i>	Supported platforms and Postgres versions for AIDB and PGFS
<i>Installing AIDB</i>	How to install and configure the AIDB extension
<i>Integrating models</i>	Local models, external APIs (OpenAI, NIM, Gemini, OpenRouter), and supported model variants
<i>Configuring external storage</i>	Using S3-compatible object stores and local file systems as pipeline data sources
<i>Reference → Functions</i>	Defining pipelines, pipeline steps, orchestration, and examples
<i>Knowledge bases reference</i>	Semantic search, hybrid search, and vector index options

次のページに続く

表 1 – 前のページからの続き

Section	What it covers
<i>SQL functions</i>	Standalone SQL functions for chunking, parsing, OCR, summarization, and embedding
<i>Reference → Functions</i>	Full API reference for all AIDB functions and views

第2章 Compatibility

2.1 AIDB

2.1.1 サポートされているプラットフォーム

Platform	x86/64	ARM64
Ubuntu 22.04 LTS	OK	—
Ubuntu 24.04 LTS	OK	—
Debian 12 (Bookworm)	OK	OK
RHEL 8	OK	—
RHEL 9	OK	OK

SLES、以前の Debian バージョン、および非 Linux プラットフォームは、現在サポートされていません。

2.1.2 サポートされている Postgres バージョン

AIDB は、EDB Postgres Advanced Server、EDB Postgres Extended、コミュニティ PostgreSQL を含む Postgres バージョン 14～18 をサポートしています。

2.2 PGFS

クラウドストレージ S3、GCS、Azure から読み取る AIDB パイプラインには、PGFS 拡張機能が必要です。プラットフォームと Postgres バージョンの互換性については、[Compatibility](#) を参照してください。

第3章 AI pipelines

パイプラインは、AIDB のコア構成要素です。これは、Postgres テーブルまたは外部ボリュームからの生データが一連の変換ステップを介してフローし、AI 対応の宛先に着地する方法を定義します。

```
Source → Step 1 → Step 2 → ... → Knowledge Base
         (parse/chunk) (embed)          (indexed + queryable)
```

各ステップは、PDF の解析、テキストのチャンク、OCR の実行、コンテンツの要約、またはベクトル埋め込みの生成など、1 つの変換を処理します。1 つのステップの出力は、次のステップの入力になります。パイプラインの最後に、データが埋め込まれ、インデックス付けされ、セマンティックまたはハイブリッド検索で照会する準備が整います。

パイプラインは、オンデマンドでバッチで、またはソースデータが変更されるたびに自動的に実行できます。手動介入なしでナレッジベースの同期を維持します。

Page	What it covers
Pipelines overview	Core concepts: sources, steps, destinations, and how pipelines relate to knowledge bases.
Creating pipelines	Defining a pipeline with <code>aidb.create_pipeline()</code> — source, steps, auto-processing, and volume sources.
Pipeline steps	Available step types: <code>ChunkText</code> , <code>ParseHtml</code> , <code>ParsePdf</code> , <code>PerformOcr</code> , <code>SummarizeText</code> , <code>KnowledgeBase</code> .
Orchestration	Auto-processing modes, background workers, observability, and error handling.
Reference → Functions	Full API reference for pipeline types, views, CRUD functions, and config helpers.
Examples	End-to-end worked example.

3.1 Creating pipelines

パイプラインは、データがソースから 1 つ以上の変換ステップを介して AI 対応の宛先に移動する方法を定義します。パイプラインの作成は、データを検索可能にするための最初のステップです。パイプラインがないと、埋め込みを生成したり、ナレッジベースとソースデータの同期を維持したりするプロセスがありません。

`aidb.create_pipeline()` を使用してパイプラインを定義します。そのパラメーターは 4 つのグループに分類されます。

Parameter group	Parameters	More information
Source	<i>source, source_key_column, source_data_column</i>	This page
Steps	<i>step_1 … step_10, step_N_options</i>	Pipeline steps
Orchestration	<i>auto_processing, background_sync_interval, batch_size</i>	Orchestration
Destination	<i>destination, destination_key_column, destination_data_column</i>	Set automatically by the <i>KnowledgeBase</i> step, or specify explicitly

3.1.1 テーブルソース

Postgres テーブルから読み取るには、**source** パラメーターをテーブル名に、**source_key_column** を一意キー列に、および **source_data_column** を処理するデータを含む列に設定します。

```
SELECT aidb.create_pipeline(
  name          => my_pipeline,
  source         => my_table,
  source_key_column => id,
  source_data_column => content,
  step_1         => KnowledgeBase,
  step_1_options  => aidb.knowledge_base_config(my_model, Text)
);
```

3.1.2 ボリュームソース PGFS

外部クラウドストレージ S3、GCS、または Azure に保存されているファイル进行处理するために、パイプラインは Postgres File System (PGFS) を使用します。PGFS は、外部オブジェクトストレージを、パイプラインがファイルをスキャンできるボリュームとしてマウントします。ボリュームの詳細については、[Configuring external storage](#) を参照してください。

ステップ 1 保存場所を作成する

`pgfs.create_storage_location` を使用して外部ストレージ接続を定義します。

```
SELECT pgfs.create_storage_location(
  name      => my_s3_location,
  uri       => s3://my-bucket/my-folder,
  options   => {"region": "us-east-1"}
);
```

ステップ 2 ボリュームを作成する

保存場所を参照する [Configuring external storage](#) を作成します。パイプラインは、このボリュームをスキャンして、新しいファイルまたは変更されたファイルを見つけます。

```
SELECT aidb.create_volume(
  name      => my_volume,
  server_name => my_s3_location,
  path      => /,
  data_format => Text
);
```

ステップ 3 ボリュームをパイプラインソースとして参照する

source パラメーターをボリューム名に設定します。

```
SELECT aidb.create_pipeline(
  name      => my_pdf_pipeline,
  source     => my_volume,
  step_1     => ParsePdf,
  step_2     => KnowledgeBase,
  step_2_options => aidb.knowledge_base_config(my_model, Text)
);
```

PGFS の完全なリファレンスについては、[PGFS documentation](#) を参照してください。

テーブルソース、パイプラインステップ、取得を組み合わせた完全なチュートリアルについては、[Examples](#) を参照してください。

3.2 Examples

パイプラインは、EDB Postgres® AI で AI ワークフローを定義する主な方法です。各パイプラインは、データが線形に流れる 1 つ以上の連続したステップで構成されています。ステップ 1 の出力はステップ 2 の入力などになります。

3.2.1 パイプラインの作成テーブルソース

標準の Postgres テーブルをデータソースとして使用してパイプラインを作成するには、`aidb.create_pipeline()` ファンクションを使用します。次の例は、シングルステップとマルチステップのパイプラインの両方をセットアップする方法を示しています。

シングルステップパイプライン

この例では、Postgres テーブルからナレッジベースを作成します。パイプラインを実行した後、セマンティック検索のために結果の埋め込みを照会できます。

```
SELECT aidb.create_pipeline(
  name => kb_pipeline_table,
  source => source_table,
  source_key_column => id,
  source_data_column => content,
```

(次のページに続く)

(前のページからの続き)

```

step_1 => KnowledgeBase,
step_1_options => aidb.knowledge_base_config(
  model => bert,  -- this is a pre-defined locally running model
  data_format => Text
)
);

```

マルチステップパイプライン複数の操作をチェーンできます。この例では、最初に HTML を解析し、結果のテキストをチャンクに分割します。

```

SELECT aidb.create_pipeline(
  name => html_processing_pipeline,
  source => web_data_table,
  source_key_column => id,
  source_data_column => html_content,
  step_1 => ParseHtml,
  step_2 => ChunkText
);

```

3.2.2 パイプラインの作成 ボリュームソース

データが外部ストレージ S3 バケットなどにある場合、最初にストレージの場所とボリュームを定義する必要があります。

ステップ 1 ストレージとボリュームを定義する

PGFS ストレージの場所と AIDB ボリュームを作成します。

```

SELECT pgfs.create_storage_location(s3_bucket_location, s3://my-ai-data,
  options => {"region": "us-east-1", "skip_signature": "true"}
);
SELECT aidb.create_volume(source_volume, s3_bucket_location, /, Text);

```

ステップ 2 パイプラインを作成する

```

SELECT aidb.create_pipeline(
  name => pipeline_from_s3,
  source => source_volume,
  step_1 => KnowledgeBase,
  step_1_options => aidb.knowledge_base_config(
    model => dummy,
    data_format => Text
  )
);

```

3.2.3 パイプラインの実行と更新

パイプラインは、作成時にデフォルトで無効になっています。それらは手動でトリガーするか、自動処理を有効にすることができます。

手動実行

既存のすべてのデータに対してパイプラインを1回実行します。

```
SELECT aidb.run_pipeline(kb_pipeline_table);
```

自動処理を有効にする

ソースデータの変更に応じてパイプラインを最新の状態に保ちます。

```
SELECT aidb.update_pipeline(
  name => kb_pipeline_table,
  auto_processing => Live
);
```

3.2.4 モニタリングと削除

既存のパイプラインとその構成を表示するには、aidb.pipelines ビューを照会します。

```
SELECT * FROM aidb.pipelines;
__OUTPUT__
  name          | source_type | source_schema | source          | source_key_column |
→source_data_column | destination_type | destination_schema | destination
| destination_key_column | destination_data_column |
          steps                                     | auto_
→processing | batch_size | background_sync_interval | owner_role
-----+-----+-----+-----+-----+
→+-----+-----+-----+-----+-----+
→+-----+-----+-----+-----+-----+
→+-----+-----+-----+-----+-----+
→+-----+-----+-----+-----+-----+
→+-----+
pipeline__8989 | Table      | public      | source_table__8989 | id
→| content          | Table      | public          | pipeline_pipeline__
→8989 | source_id          | value          | [{"options": {"max_length
→": null, "desired_length": 1000}, "operation": "ChunkText", "step_order": 1}] |
→Live              | 42 | @ 42 mins          | role_pipeline_management_
→single_step
(1 row)
```

```
SELECT name, source, auto_processing FROM aidb.pipelines;
__OUTPUT__
  name          | source.          | auto_processing
```

(次のページに続く)

(前のページからの続き)

```
-----+-----+-----
pipeline__8989 | source_table__8989 | Live
(1 row)
```

パイプラインを削除すると、パイプラインステップによって生成された宛先テーブルもドロップします。
パイプラインを削除するには

```
SELECT aidb.delete_pipeline(kb_pipeline_table);
```

3.2.5 エンドツーエンドの例 中間ストレージを使用したマルチステップパイプライン

このエンドツーエンドの例では、中間ストレージを利用する EDB Postgres AI でマルチステップパイプラインを構成する方法を示します。

`intermediate_destination` を定義することにより、個々のステップの結果テキストチャンクなどを次のステージ要約などに渡す前に永続化できます。これは、処理されたデータのデバッグまたは再利用に役立ちます。

この例の主な機能

- **マルチステップワークフロー** `ChunkText` ステップ 1 を `SummarizeText` ステップ 2 にチェーンします。
- **自動処理** 新しいデータがソーステーブルに挿入されるたびに、パイプラインは自動的にトリガーされます。
- **中間ストレージ** 中間テーブルを明示的に名前付ける方法と、システムに中間テーブルを自動生成させる方法を示します。

例 1 名前付けた中間宛先

このシナリオでは、チャンクステップの出力を保存するテーブルに手動で名前を付けます。

- 生テキストのソーステーブルを作成します。

```
CREATE TABLE source_table_demo(
  id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  content TEXT NOT NULL
);
```

- カスタム中間テーブルを使用してパイプラインを定義します。

```
SELECT * from aidb.create_pipeline(
  name => complex_pipeline_v1,
  source => source_table_demo,
  source_key_column => id,
  source_data_column => content,
  auto_processing => Live,
  step_1 => ChunkText,
```

(次のページに続く)

(前のページからの続き)

```

step_1_options => jsonb_build_object(
    desired_length, 20,
    intermediate_destination, jsonb_build_object(
        enabled, true,
        destination, my_custom_chunk_storage -- Explicitly named
    )
),
step_2 => SummarizeText
);

```

- 自動処理をトリガーするデータを挿入します。

```

INSERT INTO source_table_demo (content)
VALUES (This is a long text example that will be split into segments for easier
→processing.);

```

- チャンクと最終的なサマリーを表示します。

```

-- View intermediate chunked output
SELECT * FROM my_custom_chunk_storage;

-- View the final summarized output
SELECT * FROM pipeline_complex_pipeline_v1;

```

例 2 自動中間宛先

中間ストレージを有効にしても名前を指定しない場合、システムはパターン `pipeline_[name]_step_[n]` に従ってテーブルを生成します。

- 自動名前付けの中間ストレージを使用してパイプラインを作成します。

```

SELECT * from aidb.create_pipeline(
    name => auto_named_pipeline,
    source => source_table_demo_2,
    source_key_column => id,
    source_data_column => content,
    auto_processing => Live,
    step_1 => ChunkText,
    step_1_options => jsonb_build_object(
        desired_length, 20,
        intermediate_destination, jsonb_build_object(enabled, true)
    ),
    step_2 => SummarizeText
);

```

- 自動生成された中間テーブルにアクセスします。

```
-- View intermediate chunked output (auto-generated table name)
SELECT * FROM pipeline_auto_named_pipeline_step_1;
```

3.2.6 エンドツーエンドの例 スキャンされた PDF を検索可能なナレッジベースに

この例では、スキャンされた PDF ドキュメントのコレクションをベクトル検索可能なナレッジベースに変換する完全なパイプラインを構築する方法を示します。3つの連続したステップを使用します。

1. **PdfToImage** - 各 PDF ページを画像としてレンダリングします。
2. **PerformOcr** - OCR モデルを使用して、各レンダリング画像からテキストを抽出します。
3. **KnowledgeBase** - 抽出されたテキストを埋め込み、セマンティック検索のために保存します。

このパターンは、スキャンされた文書、テキストレイヤーのない古い PDF、および **ParsePdf** が空または低品質のテキストを返すコンテンツに最適です。中間ストレージを活用することで、ナレッジベースにフィードする前に OCR 出力を検査し、データ品質を保証し、ワークフローの問題をデバッグできます。

前提条件

登録された NVIDIA NIM PaddleOCR モデルと登録された埋め込みモデルが必要です。これらをまだお持ちでない場合は、最初に登録します。

```
-- Register the OCR model
SELECT aidb.create_model(
  my_paddle_ocr_model,
  nim_paddle_ocr,
  credentials => {"api_key": "<NVIDIA_NIM_API_KEY>"}::JSONB
);
```

埋め込みモデル bert はプリインストールされており、登録は必要ありません。別のモデルを使用している場合は、ここに登録します。

ステップ 1 ソーステーブルを作成する

生の PDF データを保持するテーブルを作成します。

```
CREATE TABLE scanned_pdfs (
  id          INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  name        TEXT NOT NULL,
  pdf_data    BYTEA NOT NULL
);
```

ステップ 2 PDF をテーブルにロードする

PDF ドキュメントをバイナリデータとして挿入します。次の例では、`pg_read_binary_file()` を使用してサーバーのファイルシステムからファイルをロードします。

```
INSERT INTO scanned_pdfs (name, pdf_data)
VALUES (quarterly_report.pdf, pg_read_binary_file(/path/to/quarterly_report.pdf));
```

S3 互換ボリュームから PDF をロードすることもできます。[Configuring external storage](#) および [S3 互換ボリューム](#) (S3 互換) を参照してください。

ステップ 3 パイプラインを作成する

```
SELECT aidb.create_pipeline(
  name          => scanned_pdf_kb,
  source         => scanned_pdfs,
  source_key_column => id,
  source_data_column => pdf_data,
  step_1         => PdfToImage,
  step_1_options  => {"dpi": 150}::jsonb,
  step_2         => PerformOcr,
  step_2_options  => aidb.ocr_config(model => my_paddle_ocr_model),
  step_3         => KnowledgeBase,
  step_3_options  => aidb.knowledge_base_config(
    model          => bert,
    data_format     => Text,
    distance_operator => Cosine
  ),
  auto_processing => Live
);
```

`auto_processing` を `Live` に設定すると、新しい PDF が `scanned_pdfs` テーブルに挿入されるたびに、パイプラインが自動的に実行されます。

ステップ 4 既存のデータでパイプラインを実行する

パイプラインを有効にする前行を挿入した場合、ここで処理します。

```
SELECT aidb.run_pipeline(scanned_pdf_kb);
```

ステップ 5 ナレッジベースを照会する

パイプラインが完了したら、セマンティック検索を使用して結果を照会します。

```
SELECT * FROM aidb.retrieve_text(
  scanned_pdf_kb,
  quarterly revenue figures,
  5 -- return top 5 matches
);
```

結果の各行は、PDF ページから抽出されたテキストブロックに対応し、クエリーとの意味的な類似性によってランク付けされます。

3.3 Orchestration

オーケストレーションはパイプラインの実行を管理し、データがいつどのように処理されるかを決定します。オーケストレーションのコアコンポーネントは次のとおりです。

- **自動処理**パイプラインはバックグラウンドで実行するか、データとの同期を保つことができます。バッチサイズ、トリガーを構成し、同期または非同期処理モードを選択できます。
- **変更検出**新しい、更新、または削除されたレコードを識別するため、ダーティデータのみが処理されます。
- **同期対非同期**ライブ即時、トランザクション更新またはバックグラウンド定期的、ノンブロッキング更新のいずれかを選択します。
- **バックグラウンドワーカー**バックグラウンドワーカーは、メインデータベースのパフォーマンスに影響を与えることなく、OCR、ドキュメント解析、API 呼び出しの埋め込みなどの AI タスクを非同期に実行する専用の Postgres プロセスです。
- **リソースの分離**これらはユーザーセッションとは無関係に実行され、大量のデータの取り込み中もアプリの高速性を保証します。
- **バッチと再試行**データを効率的なバッチにグループ化して API コストを節約し、外部サービス OpenAI などのサービスが制限されている場合に自動的に再試行します。一時的にダウン。
- **可観測性**自動化されたワークフローの詳細なステータスを提供します。これらのプロセスはバックグラウンドで実行されているため、可観測性により、状態、速度、および精度を追跡できます。
- **ステータス追跡** `aidb.pipeline_metrics` のようなビューを使用して、バックログにある行数とレコードの合計数を確認します。 `destination`。
- **監査とエラーログ**特定のファイルが破損した PDF またはタイムアウトなどの解析に失敗した理由を正確にキャプチャし、迅速なトラブルシューティングを可能にします。

完全なパイプラインワークフローに適用されたオーケストレーションを確認するには、[Examples](#) を参照してください。

3.3.1 Auto-processing

自動処理は、ソースデータテーブルまたはボリューム内と AI パイプライン出力ナレッジベースの埋め込みなどを自動的に同期させるように設計された機能セットです。

主な機能

- **完全な同期**挿入、更新、および削除を自動的に処理するため、手動介入なしでナレッジベースの精度を保ちます。
- **変更検出**新しいまたは変更されたレコードのみを処理し、変更されていないデータの不必要な再処理を回避します。
- **迅速なターンアラウンド**バッチの処理が終了すると、すぐに結果を利用できます。開始する前に完全なソーススキャンを行う必要はありません。

- **バッチ処理**レコードを同時に処理するバッチにグループ化します。これは、GPU ベースの推論タスクに最適です。

自動処理モード

パイプラインを介してデータが移動する方法を管理するには、3つの主なモードがあります。

Mode	Description	Pros	Cons
<i>Live</i>	Postgres トリガーを使用して、データを変更する同じトランザクション内の変更をすぐに処理します。	Transactional guarantee, zero lag.	Can block/delay the original data modification transaction.
<i>Background</i>	Postgres バックグラウンドワーカーを使用して、構成可能な間隔 `background_sync_interval` で非同期データを処理します。	Does not block user transactions and ideal for huge data sets.	Results are delayed by the sync interval.
<i>Disabled</i>	自動同期はありません。データは `aidb.run_pipeline()` を使用して手動で処理する必要があります。	Full control over when resources are used.	High manual overhead, requires full reprocessing for table sources.

構成

自動処理モードは、パイプラインの作成または更新時に `auto_processing` パラメーターを介して設定されます。

パイプライン作成時のモードの設定

```
SELECT aidb.create_pipeline(
  name           => my_pipeline,
  source         => my_source_table,
  ...
  auto_processing => Background,
  background_sync_interval => 5 minutes
);
```

既存のパイプラインのモードの変更

```
SELECT aidb.update_pipeline(
  my_pipeline,
  auto_processing      => Live
);
```

これらの関クションとその完全なパラメーターリストの詳細については、[Pipelines API reference](#) を参照してください。

Parameter	Type	Description
<code>auto_processing</code>	<code>aidb.PipelineAutoProcess</code>	自動処理モード <i>Live</i> 、 <i>Background</i> 、または <i>Disabled</i> 。
<code>back-ground_sync_interval</code>	<code>interval</code>	バックグラウンドでの実行間の間隔。1 秒から 2 日の間である必要があります。`auto_processing` が <i>Background</i> の場合にのみ適用されます。

3.3.2 Background workers

バックグラウンドワーカーは、EDB Postgres AI の非同期 AI パイプラインの実行エンジンです。標準のデータベーストランザクションをブロックすることなく、大量のデータを処理できます。

バックグラウンドワーカーは、`auto_processing => 'Background'` でパイプラインが作成または更新されると、自動的にアクティブ化されます。バックグラウンドモードを有効にする前に、必要な Postgres サーバー設定が設定されていることを確認します。

Postgres の前提条件

バックグラウンドパイプラインを実行する前に、2 つの `postgresql.conf` 設定を設定する必要があります。

`shared_preload_libraries`

バックグラウンドワーカーを起動する前に、`postgresql.conf` で `shared_preload_libraries = 'aidb'` を設定する必要があります。これは AIDB のインストール中に構成されます。[Configuring AIDB](#) を参照してください。パイプラインでバックグラウンドモードを有効にする前に、設定されていることを確認してください。

`max_worker_processes`

バックグラウンドモードの各パイプラインは、1 つの専用の Postgres バックグラウンドワーカープロセスを必要とします。`max_worker_processes` 設定は、クラスター全体で許可されるバックグラウンドワーカープロセスの数を制御します。

デフォルト値は 8 です。複数のバックグラウンドパイプラインを実行する予定がある場合は、この値を増やします。

```
max_worker_processes = 20
```

この設定を変更した後、Postgres を再起動します。現在の値を確認するには

```
SHOW max_worker_processes;
```

`max_worker_processes` が使い果たされると、新しく作成されたバックグラウンドパイプラインはワーカーを起動せず、容量が解放されるまでサイレントに処理をスキップします。

コア機能

- **非同期実行**パイプラインがバックグラウンドモードに設定されている場合、処理はユーザーセッションとは無関係に発生します。この方法により、ソーステーブルのクエリまたはデータ変更が、生成または OCR タスクの埋め込みによって遅延されません。
- **バッチ処理**バックグラウンドワーカーは、レコードを構成可能なバッチサイズにグループ化します。これにより、特に GPU ベースのモデルまたはリモート AI サービス API と対話する場合、スループットが最適化されます。
- **パラレルオペレーション**各バッチ内で、ワーカーはパイプラインステップデータの取得、埋め込み計算、およびストレージをパラレル操作として実行して、パフォーマンスを最大化します。
- **継続ポーリングワーカー**は、定義された `background_sync_interval` に基づいて、変更がないかソースを継続的に監視します。

変更検出

バックグラウンドワーカーは、特定の検出ロジックを使用してさまざまなソースタイプを処理します。

- **テーブルソース**軽量トリガーは、挿入、更新、削除などの変更イベントをキャプチャし、バックログに配置します。バックグラウンドワーカーは、次のインターバルでこのバックログを処理します。
- **ボリュームソース**バックグラウンドワーカーは、外部ストレージのスキャンを実行します。ファイルの `last_modified` タイムスタンプを状態テーブルと比較して、新しいドキュメントまたは変更されたドキュメントを識別します。

構成とコンストレイン

必須

- **Postgres 前提条件**バックグラウンドワーカーには、`shared_preload_libraries = 'aidb'` と `postgresql.conf` に十分な `max_worker_processes` 値が必要です。このページの Postgres [\[1\]](#) [\[2\]](#) [\[3\]](#) [\[4\]](#) [\[5\]](#) [\[6\]](#) [\[7\]](#) [\[8\]](#) [\[9\]](#) [\[10\]](#) [\[11\]](#) [\[12\]](#) [\[13\]](#) [\[14\]](#) [\[15\]](#) [\[16\]](#) [\[17\]](#) [\[18\]](#) [\[19\]](#) [\[20\]](#) [\[21\]](#) [\[22\]](#) [\[23\]](#) [\[24\]](#) [\[25\]](#) [\[26\]](#) [\[27\]](#) [\[28\]](#) [\[29\]](#) [\[30\]](#) [\[31\]](#) [\[32\]](#) [\[33\]](#) [\[34\]](#) [\[35\]](#) [\[36\]](#) [\[37\]](#) [\[38\]](#) [\[39\]](#) [\[40\]](#) [\[41\]](#) [\[42\]](#) [\[43\]](#) [\[44\]](#) [\[45\]](#) [\[46\]](#) [\[47\]](#) [\[48\]](#) [\[49\]](#) [\[50\]](#) [\[51\]](#) [\[52\]](#) [\[53\]](#) [\[54\]](#) [\[55\]](#) [\[56\]](#) [\[57\]](#) [\[58\]](#) [\[59\]](#) [\[60\]](#) [\[61\]](#) [\[62\]](#) [\[63\]](#) [\[64\]](#) [\[65\]](#) [\[66\]](#) [\[67\]](#) [\[68\]](#) [\[69\]](#) [\[70\]](#) [\[71\]](#) [\[72\]](#) [\[73\]](#) [\[74\]](#) [\[75\]](#) [\[76\]](#) [\[77\]](#) [\[78\]](#) [\[79\]](#) [\[80\]](#) [\[81\]](#) [\[82\]](#) [\[83\]](#) [\[84\]](#) [\[85\]](#) [\[86\]](#) [\[87\]](#) [\[88\]](#) [\[89\]](#) [\[90\]](#) [\[91\]](#) [\[92\]](#) [\[93\]](#) [\[94\]](#) [\[95\]](#) [\[96\]](#) [\[97\]](#) [\[98\]](#) [\[99\]](#) [\[100\]](#) [\[101\]](#) [\[102\]](#) [\[103\]](#) [\[104\]](#) [\[105\]](#) [\[106\]](#) [\[107\]](#) [\[108\]](#) [\[109\]](#) [\[110\]](#) [\[111\]](#) [\[112\]](#) [\[113\]](#) [\[114\]](#) [\[115\]](#) [\[116\]](#) [\[117\]](#) [\[118\]](#) [\[119\]](#) [\[120\]](#) [\[121\]](#) [\[122\]](#) [\[123\]](#) [\[124\]](#) [\[125\]](#) [\[126\]](#) [\[127\]](#) [\[128\]](#) [\[129\]](#) [\[130\]](#) [\[131\]](#) [\[132\]](#) [\[133\]](#) [\[134\]](#) [\[135\]](#) [\[136\]](#) [\[137\]](#) [\[138\]](#) [\[139\]](#) [\[140\]](#) [\[141\]](#) [\[142\]](#) [\[143\]](#) [\[144\]](#) [\[145\]](#) [\[146\]](#) [\[147\]](#) [\[148\]](#) [\[149\]](#) [\[150\]](#) [\[151\]](#) [\[152\]](#) [\[153\]](#) [\[154\]](#) [\[155\]](#) [\[156\]](#) [\[157\]](#) [\[158\]](#) [\[159\]](#) [\[160\]](#) [\[161\]](#) [\[162\]](#) [\[163\]](#) [\[164\]](#) [\[165\]](#) [\[166\]](#) [\[167\]](#) [\[168\]](#) [\[169\]](#) [\[170\]](#) [\[171\]](#) [\[172\]](#) [\[173\]](#) [\[174\]](#) [\[175\]](#) [\[176\]](#) [\[177\]](#) [\[178\]](#) [\[179\]](#) [\[180\]](#) [\[181\]](#) [\[182\]](#) [\[183\]](#) [\[184\]](#) [\[185\]](#) [\[186\]](#) [\[187\]](#) [\[188\]](#) [\[189\]](#) [\[190\]](#) [\[191\]](#) [\[192\]](#) [\[193\]](#) [\[194\]](#) [\[195\]](#) [\[196\]](#) [\[197\]](#) [\[198\]](#) [\[199\]](#) [\[200\]](#) [\[201\]](#) [\[202\]](#) [\[203\]](#) [\[204\]](#) [\[205\]](#) [\[206\]](#) [\[207\]](#) [\[208\]](#) [\[209\]](#) [\[210\]](#) [\[211\]](#) [\[212\]](#) [\[213\]](#) [\[214\]](#) [\[215\]](#) [\[216\]](#) [\[217\]](#) [\[218\]](#) [\[219\]](#) [\[220\]](#) [\[221\]](#) [\[222\]](#) [\[223\]](#) [\[224\]](#) [\[225\]](#) [\[226\]](#) [\[227\]](#) [\[228\]](#) [\[229\]](#) [\[230\]](#) [\[231\]](#) [\[232\]](#) [\[233\]](#) [\[234\]](#) [\[235\]](#) [\[236\]](#) [\[237\]](#) [\[238\]](#) [\[239\]](#) [\[240\]](#) [\[241\]](#) [\[242\]](#) [\[243\]](#) [\[244\]](#) [\[245\]](#) [\[246\]](#) [\[247\]](#) [\[248\]](#) [\[249\]](#) [\[250\]](#) [\[251\]](#) [\[252\]](#) [\[253\]](#) [\[254\]](#) [\[255\]](#) [\[256\]](#) [\[257\]](#) [\[258\]](#) [\[259\]](#) [\[260\]](#) [\[261\]](#) [\[262\]](#) [\[263\]](#) [\[264\]](#) [\[265\]](#) [\[266\]](#) [\[267\]](#) [\[268\]](#) [\[269\]](#) [\[270\]](#) [\[271\]](#) [\[272\]](#) [\[273\]](#) [\[274\]](#) [\[275\]](#) [\[276\]](#) [\[277\]](#) [\[278\]](#) [\[279\]](#) [\[280\]](#) [\[281\]](#) [\[282\]](#) [\[283\]](#) [\[284\]](#) [\[285\]](#) [\[286\]](#) [\[287\]](#) [\[288\]](#) [\[289\]](#) [\[290\]](#) [\[291\]](#) [\[292\]](#) [\[293\]](#) [\[294\]](#) [\[295\]](#) [\[296\]](#) [\[297\]](#) [\[298\]](#) [\[299\]](#) [\[300\]](#) [\[301\]](#) [\[302\]](#) [\[303\]](#) [\[304\]](#) [\[305\]](#) [\[306\]](#) [\[307\]](#) [\[308\]](#) [\[309\]](#) [\[310\]](#) [\[311\]](#) [\[312\]](#) [\[313\]](#) [\[314\]](#) [\[315\]](#) [\[316\]](#) [\[317\]](#) [\[318\]](#) [\[319\]](#) [\[320\]](#) [\[321\]](#) [\[322\]](#) [\[323\]](#) [\[324\]](#) [\[325\]](#) [\[326\]](#) [\[327\]](#) [\[328\]](#) [\[329\]](#) [\[330\]](#) [\[331\]](#) [\[332\]](#) [\[333\]](#) [\[334\]](#) [\[335\]](#) [\[336\]](#) [\[337\]](#) [\[338\]](#) [\[339\]](#) [\[340\]](#) [\[341\]](#) [\[342\]](#) [\[343\]](#) [\[344\]](#) [\[345\]](#) [\[346\]](#) [\[347\]](#) [\[348\]](#) [\[349\]](#) [\[350\]](#) [\[351\]](#) [\[352\]](#) [\[353\]](#) [\[354\]](#) [\[355\]](#) [\[356\]](#) [\[357\]](#) [\[358\]](#) [\[359\]](#) [\[360\]](#) [\[361\]](#) [\[362\]](#) [\[363\]](#) [\[364\]](#) [\[365\]](#) [\[366\]](#) [\[367\]](#) [\[368\]](#) [\[369\]](#) [\[370\]](#) [\[371\]](#) [\[372\]](#) [\[373\]](#) [\[374\]](#) [\[375\]](#) [\[376\]](#) [\[377\]](#) [\[378\]](#) [\[379\]](#) [\[380\]](#) [\[381\]](#) [\[382\]](#) [\[383\]](#) [\[384\]](#) [\[385\]](#) [\[386\]](#) [\[387\]](#) [\[388\]](#) [\[389\]](#) [\[390\]](#) [\[391\]](#) [\[392\]](#) [\[393\]](#) [\[394\]](#) [\[395\]](#) [\[396\]](#) [\[397\]](#) [\[398\]](#) [\[399\]](#) [\[400\]](#) [\[401\]](#) [\[402\]](#) [\[403\]](#) [\[404\]](#) [\[405\]](#) [\[406\]](#) [\[407\]](#) [\[408\]](#) [\[409\]](#) [\[410\]](#) [\[411\]](#) [\[412\]](#) [\[413\]](#) [\[414\]](#) [\[415\]](#) [\[416\]](#) [\[417\]](#) [\[418\]](#) [\[419\]](#) [\[420\]](#) [\[421\]](#) [\[422\]](#) [\[423\]](#) [\[424\]](#) [\[425\]](#) [\[426\]](#) [\[427\]](#) [\[428\]](#) [\[429\]](#) [\[430\]](#) [\[431\]](#) [\[432\]](#) [\[433\]](#) [\[434\]](#) [\[435\]](#) [\[436\]](#) [\[437\]](#) [\[438\]](#) [\[439\]](#) [\[440\]](#) [\[441\]](#) [\[442\]](#) [\[443\]](#) [\[444\]](#) [\[445\]](#) [\[446\]](#) [\[447\]](#) [\[448\]](#) [\[449\]](#) [\[450\]](#) [\[451\]](#) [\[452\]](#) [\[453\]](#) [\[454\]](#) [\[455\]](#) [\[456\]](#) [\[457\]](#) [\[458\]](#) [\[459\]](#) [\[460\]](#) [\[461\]](#) [\[462\]](#) [\[463\]](#) [\[464\]](#) [\[465\]](#) [\[466\]](#) [\[467\]](#) [\[468\]](#) [\[469\]](#) [\[470\]](#) [\[471\]](#) [\[472\]](#) [\[473\]](#) [\[474\]](#) [\[475\]](#) [\[476\]](#) [\[477\]](#) [\[478\]](#) [\[479\]](#) [\[480\]](#) [\[481\]](#) [\[482\]](#) [\[483\]](#) [\[484\]](#) [\[485\]](#) [\[486\]](#) [\[487\]](#) [\[488\]](#) [\[489\]](#) [\[490\]](#) [\[491\]](#) [\[492\]](#) [\[493\]](#) [\[494\]](#) [\[495\]](#) [\[496\]](#) [\[497\]](#) [\[498\]](#) [\[499\]](#) [\[500\]](#) [\[501\]](#) [\[502\]](#) [\[503\]](#) [\[504\]](#) [\[505\]](#) [\[506\]](#) [\[507\]](#) [\[508\]](#) [\[509\]](#) [\[510\]](#) [\[511\]](#) [\[512\]](#) [\[513\]](#) [\[514\]](#) [\[515\]](#) [\[516\]](#) [\[517\]](#) [\[518\]](#) [\[519\]](#) [\[520\]](#) [\[521\]](#) [\[522\]](#) [\[523\]](#) [\[524\]](#) [\[525\]](#) [\[526\]](#) [\[527\]](#) [\[528\]](#) [\[529\]](#) [\[530\]](#) [\[531\]](#) [\[532\]](#) [\[533\]](#) [\[534\]](#) [\[535\]](#) [\[536\]](#) [\[537\]](#) [\[538\]](#) [\[539\]](#) [\[540\]](#) [\[541\]](#) [\[542\]](#) [\[543\]](#) [\[544\]](#) [\[545\]](#) [\[546\]](#) [\[547\]](#) [\[548\]](#) [\[549\]](#) [\[550\]](#) [\[551\]](#) [\[552\]](#) [\[553\]](#) [\[554\]](#) [\[555\]](#) [\[556\]](#) [\[557\]](#) [\[558\]](#) [\[559\]](#) [\[560\]](#) [\[561\]](#) [\[562\]](#) [\[563\]](#) [\[564\]](#) [\[565\]](#) [\[566\]](#) [\[567\]](#) [\[568\]](#) [\[569\]](#) [\[570\]](#) [\[571\]](#) [\[572\]](#) [\[573\]](#) [\[574\]](#) [\[575\]](#) [\[576\]](#) [\[577\]](#) [\[578\]](#) [\[579\]](#) [\[580\]](#) [\[581\]](#) [\[582\]](#) [\[583\]](#) [\[584\]](#) [\[585\]](#) [\[586\]](#) [\[587\]](#) [\[588\]](#) [\[589\]](#) [\[590\]](#) [\[591\]](#) [\[592\]](#) [\[593\]](#) [\[594\]](#) [\[595\]](#) [\[596\]](#) [\[597\]](#) [\[598\]](#) [\[599\]](#) [\[600\]](#) [\[601\]](#) [\[602\]](#) [\[603\]](#) [\[604\]](#) [\[605\]](#) [\[606\]](#) [\[607\]](#) [\[608\]](#) [\[609\]](#) [\[610\]](#) [\[611\]](#) [\[612\]](#) [\[613\]](#) [\[614\]](#) [\[615\]](#) [\[616\]](#) [\[617\]](#) [\[618\]](#) [\[619\]](#) [\[620\]](#) [\[621\]](#) [\[622\]](#) [\[623\]](#) [\[624\]](#) [\[625\]](#) [\[626\]](#) [\[627\]](#) [\[628\]](#) [\[629\]](#) [\[630\]](#) [\[631\]](#) [\[632\]](#) [\[633\]](#) [\[634\]](#) [\[635\]](#) [\[636\]](#) [\[637\]](#) [\[638\]](#) [\[639\]](#) [\[640\]](#) [\[641\]](#) [\[642\]](#) [\[643\]](#) [\[644\]](#) [\[645\]](#) [\[646\]](#) [\[647\]](#) [\[648\]](#) [\[649\]](#) [\[650\]](#) [\[651\]](#) [\[652\]](#) [\[653\]](#) [\[654\]](#) [\[655\]](#) [\[656\]](#) [\[657\]](#) [\[658\]](#) [\[659\]](#) [\[660\]](#) [\[661\]](#) [\[662\]](#) [\[663\]](#) [\[664\]](#) [\[665\]](#) [\[666\]](#) [\[667\]](#) [\[668\]](#) [\[669\]](#) [\[670\]](#) [\[671\]](#) [\[672\]](#) [\[673\]](#) [\[674\]](#) [\[675\]](#) [\[676\]](#) [\[677\]](#) [\[678\]](#) [\[679\]](#) [\[680\]](#) [\[681\]](#) [\[682\]](#) [\[683\]](#) [\[684\]](#) [\[685\]](#) [\[686\]](#) [\[687\]](#) [\[688\]](#) [\[689\]](#) [\[690\]](#) [\[691\]](#) [\[692\]](#) [\[693\]](#) [\[694\]](#) [\[695\]](#) [\[696\]](#) [\[697\]](#) [\[698\]](#) [\[699\]](#) [\[700\]](#) [\[701\]](#) [\[702\]](#) [\[703\]](#) [\[704\]](#) [\[705\]](#) [\[706\]](#) [\[707\]](#) [\[708\]](#) [\[709\]](#) [\[710\]](#) [\[711\]](#) [\[712\]](#) [\[713\]](#) [\[714\]](#) [\[715\]](#) [\[716\]](#) [\[717\]](#) [\[718\]](#) [\[719\]](#) [\[720\]](#) [\[721\]](#) [\[722\]](#) [\[723\]](#) [\[724\]](#) [\[725\]](#) [\[726\]](#) [\[727\]](#) [\[728\]](#) [\[729\]](#) [\[730\]](#) [\[731\]](#) [\[732\]](#) [\[733\]](#) [\[734\]](#) [\[735\]](#) [\[736\]](#) [\[737\]](#) [\[738\]](#) [\[739\]](#) [\[740\]](#) [\[741\]](#) [\[742\]](#) [\[743\]](#) [\[744\]](#) [\[745\]](#) [\[746\]](#) [\[747\]](#) [\[748\]](#) [\[749\]](#) [\[750\]](#) [\[751\]](#) [\[752\]](#) [\[753\]](#) [\[754\]](#) [\[755\]](#) [\[756\]](#) [\[757\]](#) [\[758\]](#) [\[759\]](#) [\[760\]](#) [\[761\]](#) [\[762\]](#) [\[763\]](#) [\[764\]](#) [\[765\]](#) [\[766\]](#) [\[767\]](#) [\[768\]](#) [\[769\]](#) [\[770\]](#) [\[771\]](#) [\[772\]](#) [\[773\]](#) [\[774\]](#) [\[775\]](#) [\[776\]](#) [\[777\]](#) [\[778\]](#) [\[779\]](#) [\[780\]](#) [\[781\]](#) [\[782\]](#) [\[783\]](#) [\[784\]](#) [\[785\]](#) [\[786\]](#) [\[787\]](#) [\[788\]](#) [\[789\]](#) [\[790\]](#) [\[791\]](#) [\[792\]](#) [\[793\]](#) [\[794\]](#) [\[795\]](#) [\[796\]](#) [\[797\]](#) [\[798\]](#) [\[799\]](#) [\[800\]](#) [\[801\]](#) [\[802\]](#) [\[803\]](#) [\[804\]](#) [\[805\]](#) [\[806\]](#) [\[807\]](#) [\[808\]](#) [\[809\]](#) [\[810\]](#) [\[811\]](#) [\[812\]](#) [\[813\]](#) [\[814\]](#) [\[815\]](#) [\[816\]](#) [\[817\]](#) [\[818\]](#) [\[819\]](#) [\[820\]](#) [\[821\]](#) [\[822\]](#) [\[823\]](#) [\[824\]](#) [\[825\]](#) [\[826\]](#) [\[827\]](#) [\[828\]](#) [\[829\]](#) [\[830\]](#) [\[831\]](#) [\[832\]](#) [\[833\]](#) [\[834\]](#) [\[835\]](#) [\[836\]](#) [\[837\]](#) [\[838\]](#) [\[839\]](#) [\[840\]](#) [\[841\]](#) [\[842\]](#) [\[843\]](#) [\[844\]](#) [\[845\]](#) [\[846\]](#) [\[847\]](#) [\[848\]](#) [\[849\]](#) [\[850\]](#) [\[851\]](#) [\[852\]](#) [\[853\]](#) [\[854\]](#) [\[855\]](#) [\[856\]](#) [\[857\]](#) [\[858\]](#) [\[859\]](#) [\[860\]](#) [\[861\]](#) [\[862\]](#) [\[863\]](#) [\[864\]](#) [\[865\]](#) [\[866\]](#) [\[867\]](#) [\[868\]](#) [\[869\]](#) [\[870\]](#) [\[871\]](#) [\[872\]](#) [\[873\]](#) [\[874\]](#) [\[875\]](#) [\[876\]](#) [\[877\]](#) [\[878\]](#) [\[879\]](#) [\[880\]](#) [\[881\]](#) [\[882\]](#) [\[883\]](#) [\[884\]](#) [\[885\]](#) [\[886\]](#) [\[887\]](#) [\[888\]](#) [\[889\]](#) [\[890\]](#) [\[891\]](#) [\[892\]](#) [\[893\]](#) [\[894\]](#) [\[895\]](#) [\[896\]](#) [\[897\]](#) [\[898\]](#) [\[899\]](#) [\[900\]](#) [\[901\]](#) [\[902\]](#) [\[903\]](#) [\[904\]](#) [\[905\]](#) [\[906\]](#) [\[907\]](#) [\[908\]](#) [\[909\]](#) [\[910\]](#) [\[911\]](#) [\[912\]](#) [\[913\]](#) [\[914\]](#) [\[915\]](#) [\[916\]](#) [\[917\]](#) [\[918\]](#) [\[919\]](#) [\[920\]](#) [\[921\]](#) [\[922\]](#) [\[923\]](#) [\[924\]](#) [\[925\]](#) [\[926\]](#) [\[927\]](#) [\[928\]](#) [\[929\]](#) [\[930\]](#) [\[931\]](#) [\[932\]](#) [\[933\]](#) [\[934\]](#) [\[935\]](#) [\[936\]](#) [\[937\]](#) [\[938\]](#) [\[939\]](#) [\[940\]](#) [\[941\]](#) [\[942\]](#) [\[943\]](#) [\[944\]](#) [\[945\]](#) [\[946\]](#) [\[947\]](#) [\[948\]](#) [\[949\]](#) [\[950\]](#) [\[951\]](#) [\[952\]](#) [\[953\]](#) [\[954\]](#) [\[955\]](#) [\[956\]](#) [\[957\]](#) [\[958\]](#) [\[959\]](#) [\[960\]](#) [\[961\]](#) [\[962\]](#) [\[963\]](#) [\[964\]](#) [\[965\]](#) [\[966\]](#) [\[967\]](#) [\[968\]](#) [\[969\]](#) [\[970\]](#) [\[971\]](#) [\[972\]](#) [\[973\]](#) [\[974\]](#) [\[975\]](#) [\[976\]](#) [\[977\]](#) [\[978\]](#) [\[979\]](#) [\[980\]](#) [\[981\]](#) [\[982\]](#) [\[983\]](#) [\[984\]](#) [\[985\]](#) [\[986\]](#) [\[987\]](#) [\[988\]](#) [\[989\]](#) [\[990\]](#) [\[991\]](#) [\[992\]](#) [\[993\]](#) [\[994\]](#) [\[995\]](#) [\[996\]](#) [\[997\]](#) [\[998\]](#) [\[999\]](#) [\[1000\]](#) [\[1001\]](#) [\[1002\]](#) [\[1003\]](#) [\[1004\]](#) [\[1005\]](#) [\[1006\]](#) [\[1007\]](#) [\[1008\]](#) [\[1009\]](#) [\[1010\]](#) [\[1011\]](#) [\[1012\]](#) [\[1013\]](#) [\[1014\]](#) [\[1015\]](#) [\[1016\]](#) [\[1017\]](#) [\[1018\]](#) [\[1019\]](#) [\[1020\]](#) [\[1021\]](#) [\[1022\]](#) [\[1023\]](#) [\[1024\]](#) [\[1025\]](#) [\[1026\]](#) [\[1027\]](#) [\[1028\]](#) [\[1029\]](#) [\[1030\]](#) [\[1031\]](#) [\[1032\]](#) [\[1033\]](#) [\[1034\]](#) [\[1035\]](#) [\[1036\]](#) [\[1037\]](#) [\[1038\]](#) [\[1039\]](#) [\[1040\]](#) [\[1041\]](#) [\[1042\]](#) [\[1043\]](#) [\[1044\]](#) [\[1045\]](#) [\[1046\]](#) [\[1047\]](#) [\[1048\]](#) [\[1049\]](#) [\[1050\]](#) [\[1051\]](#) [\[1052\]](#) [\[1053\]](#) [\[1054\]](#) [\[1055\]](#) [\[1056\]](#) [\[1057\]](#) [\[105](#)

生し、コストまたはレート制限に影響を与える可能性があります。これらの場合、より長い間隔を検討します。

```
SELECT aidb.update_pipeline(
  my_s3_pipeline,
  background_sync_interval => 1 day
);
```

background_sync_interval は、Postgres 間隔値 '5 minutes'、'1 hour'、'1 day' などを受け入れます。

モニタリングとオブザーバビリティ

aidb.pipeline_metrics ビュー aidb.pipem としてもアクセスを使用して、バックグラウンドワーカーのステータスと状態を追跡できます。主要なメトリックには次のものが含まれます。

- **Table- unprocessed rows** テーブルソースの場合、まだ処理されていない行の数。
- **ボリューム-スキャン完了** ボリュームソースの場合、完了したフルスキャンの数。
- **Count (source records)** ソース内のレコードの合計数。
- **Count (destination records)** 宛先のレコードの合計数。
- **Status** 現在のパイプラインのステータス。

3.3.3 Pipeline error log

各パイプラインにはエラーログテーブルがあります。レコードが任意のステップで失敗すると、ログは、ソースレコード ID、失敗したステップ、操作タイプ、およびエラーメッセージをキャプチャします。バッチの残りの部分は処理を継続します。

その仕組み

- レコードに失敗すると、ソース ID、失敗したステップ番号、操作タイプ、エラーメッセージとともに {source_schema}.pipeline_{pipeline_name}_errors ログに記録されます。
- 単一の不良レコードはパイプラインをブロックしません。バッチ内の他のレコードは、残りのすべてのステップを続行します。
- エラーログテーブルはパイプラインで作成され、パイプラインが削除されるときに削除されます。
- エラーは 4 つの error_category 値のいずれかを取得します。

Category	Scope	Blocks pipeline?	Description
RecordTemporary	Record	No	単一レコードの一時的な障害。再試行で成功する場合があります。
RecordPermanent	Record	No	単一のレコードの永続的な障害たとえば、破損したデータ。

次のページに続く

表 5 – 前のページからの続き

Category	Scope	Blocks pipeline?	Description
<i>PipelineTemporary</i>	Pipeline	Yes	ステップ全体を停止した一時的な障害たとえば、サービスの停止。再試行で解決される場合があります。
<i>PipelinePermanent</i>	Pipeline	Yes	ステップ全体を停止した永続的な障害たとえば、無効なモデル構成。

RecordTemporary および RecordPermanent エラーはパイプラインをブロックしません。他のレコードは処理を継続します。PipelineTemporary および PipelinePermanent エラーは、ステップ全体をブロックします。一時的なエラーは、再試行すると解決される場合があります。永続的なエラーは、不正なデータまたは構成ミスを示しています。

エラーのクエリー

aidb.get_error_logs() を使用して、パイプラインのエラーログを検査します。

```
SELECT * FROM aidb.get_error_logs(my_pipeline);
```

__OUTPUT__

id	source_id	part_ids	pipeline_step	step_operation	error_message	error_category	failed_at	retry_count	last_retry_at
3	doc_99	{0,2}	2	ChunkText	text segment exceeded maximum length	RecordPermanent	2026-04-10 14:32:01.123+00	0	
2	doc_42	{0}	1	ParsePdf	failed to extract text from page 0	RecordPermanent	2026-04-10 14:31:58.456+00	0	
1	doc_7		1	ParsePdf	corrupt PDF header	RecordPermanent	2026-04-10 14:31:55.789+00	0	

(3 rows)

ソースレコード、ステップ番号、またはエラーカテゴリでフィルタリングします。

```
SELECT * FROM aidb.get_error_logs(
  my_pipeline,
  p_source_id => doc_42
);
```

```
SELECT * FROM aidb.get_error_logs(
  my_pipeline,
  p_pipeline_step => 1::smallint,
```

(次のページに続く)

(前のページからの続き)

```
p_error_category => RecordPermanent
);
```

p_limit および p_offset を使用したページ付け

```
SELECT * FROM aidb.get_error_logs(my_pipeline, p_limit => 20, p_offset => 40);
```

完全なパラメーターと戻り型の詳細については、[Operations reference](#) を参照してください。

エラーの概要

1つのパイプラインのステップ、オペレーション、およびカテゴリ別にグループ化されたエラー数を取得します。

```
SELECT * FROM aidb.get_error_log_summary(my_pipeline);
```

```
__OUTPUT__
pipeline_step | step_operation | error_category | error_count | latest_failed_at
-----+-----+-----+-----+-----
1 | ParsePdf | RecordPermanent | 2 | 2026-04-10 14:31:58.456+00
2 | ChunkText | RecordPermanent | 1 | 2026-04-10 14:32:01.123+00
(2 rows)
```

または、すべてのパイプラインで同じロールアップを一度に取得します。

```
SELECT * FROM aidb.get_all_error_summaries();
```

```
__OUTPUT__
pipeline_name | pipeline_step | step_operation | error_category | error_count | latest_failed_at
-----+-----+-----+-----+-----+-----
my_pipeline | 1 | ParsePdf | RecordPermanent | 2 | 2026-04-10 14:31:58.456+00
my_pipeline | 2 | ChunkText | RecordPermanent | 1 | 2026-04-10 14:32:01.123+00
pdf_ingester | 1 | ParsePdf | PipelinePermanent | 1 | 2026-04-10 15:00:12.000+00
(3 rows)
```

エラーのクリア

ID で特定のエラーログエントリを削除します。

```
SELECT aidb.clear_error_logs(my_pipeline, ARRAY[1, 3]);
```

```
__OUTPUT__
clear_error_logs
-----
                2
(1 row)
```

削除された行数を返します。クリアは ID ベースです。エントリを削除する前に `get_error_logs()` で確認してください。

失敗したレコードの再試行

!!!注 `aidb.retry_pipeline_errors()` はまだ利用できません。将来のリリースに含まれる予定です。
[Known issues](#) を参照してください。

`aidb.retry_pipeline_errors()` は、利用可能な場合、失敗したステップから特定のエラーログエントリを再処理します。

```
SELECT * FROM aidb.retry_pipeline_errors(my_pipeline, ARRAY[1, 2]);
```

```
__OUTPUT__
error_id | status   | error_message
-----+-----+-----
        1 | resolved |
        2 | failed   | failed to extract text from page 0
(2 rows)
```

解決されたエントリはエラーログから削除されます。失敗した再試行は、インクリメントされた `retry_count` および更新された `last_retry_at` タイムスタンプとともにログに残ります。

パイプラインステータスの統合

エラー数は `aidb.pipeline_metrics` ビューにも表示されます。パイプラインステータスはエラーの重大度を反映します。

- **PartialErrors** —一部のレコードは失敗しました。残りは通常に処理されました。
- **BlockingErrors** —パイプラインレベルのエラーにより、ステップが完全に停止しました。

パイプラインの状態のモニタリングの詳細については、`PIPELINE_STATUS` を参照してください。

3.3.4 Observability

すべての自動処理モードの詳細なステータスと進行状況の出力を確認します。

2つのビューが可観測性のニーズのほとんどをカバーします。

- `aidb.pipeline_metrics` エイリアス `aidb.pipem` — バックログ、ソース/宛先カウント、ステータス、エラーカウントを含むパイプラインごとに 1 行。これは、処理の進行状況を追跡するためのプライマリビューです。
- `aidb.knowledge_base_metrics` エイリアス `aidb.kbm` — パイプライン数、埋め込み数、および接続されているすべてのパイプラインの集計ステータスを含むナレッジベースごとに 1 行。

パイプラインごとの進行状況

```
SELECT * FROM aidb.pipem;
__OUTPUT__
  pipeline      | auto processing | table: unprocessed rows | volume: scans
→ completed | count(source records) | count(destination records) | Status
→ count(record errors) | count(blocking errors)
-----+-----+-----+-----+
→ -----+-----+-----+-----+
→ -----+-----+-----+-----+
feedback_na_pipe | Live           |                          | 0 |
  |              | 412 |              | 412 | UpToDate |
  0 |              | 0 |
product_docs_pipe | Background     |                          |   |
  6 |              | 1842 |          | 1842 | UpToDate |
  0 |              | 0 |
(2 rows)
```

メカニズムはボリュームソースとテーブルソースでは異なるため、ビューは次のメトリックを公開します。

- `table: unprocessed rows` 変更イベントのバックログにリストされる一意の行の数。Disabled の場合、自動処理は新しい変更イベントをキャプチャしません。
- `volume: scans completed` 完了したソースの完全なリストの数。
- `count(source records)` ソースに存在するレコード数。テーブルソースに関しては常に正確です。ボリュームソースの各フルスキャンの後に更新されます。
- `count(destination records)` パイプラインがその宛先エンベディング、チャンクなどに書き込んだ行数。

ナレッジベースの概要

```
SELECT * FROM aidb.kbm;
__OUTPUT__
```

(次のページに続く)

(前のページからの続き)

name	pipelines	embeddings	status
<code>public.product_catalog_kb</code>	1	1842	UpToDate
<code>public.customer_feedback_kb</code>	2	915	UpToDate

(2 rows)

`status` 列は、その KB に接続されているすべてのパイプラインにわたって最悪のパイプラインステータスをロールアップします。`aidb.pipeline_metrics` をドリルして、どのパイプラインが原因であるかを見つけてみます。

3.4 Pipelines overview

Pipelines は、生データのベクトル検索可能な洞察への変換を自動化する EDB Postgres AI コンポーネントです。`pgvector` 検索機能をビルトインデータ準備ツールと統合し、手動介入なしでナレッジベースとソースデータの同期を維持します。

3.4.1 統合されたデータのライフサイクル

Pipelines は、生データから検索可能なインサイトまでを自動化します。データの取り込み、埋め込み生成、インデックス作成を管理することにより、ベクターストアがソースデータとの同期を維持します。

- **柔軟なモデル統合** HuggingFace などのサポートされているローカルモデルから選択するか、OpenAI 互換の外部 API に接続します。
- **前処理の自動化** ビルトインのデータ準備ステップは、AI パイプライン内でデータのクリーニングと変換を直接処理します。
- **古いデータの削除** 自動処理により、埋め込みが最新の状態に保たれ、古いベクターデータが原因で発生することが多い幻覚を大幅に削減します。
- **統合セマンティック検索** インテリジェントなナレッジベース機能を使用すると、データが Postgres テーブルまたは S3 互換オブジェクトストレージに存在するかどうかに関係なく、単一のファンクション呼び出しを使用してテキストと画像を照会できます。

3.4.2 パイプラインアーキテクチャ

パイプラインは `aidb.create_pipeline()` ファンクションを使用して構成され、3 つの主な側面で構成されます。

- **Creating pipelines** — Postgres テーブルまたは外部クラウドストレージ S3、GCS、Azure から Postgres File System (PGFS) を介して読み取られるパイプライン。
- **Pipeline steps** — 最大 10 の連続した操作がデータを変換します。手順には、チャンク、解析、OCR、要約、ナレッジベースへの埋め込みが含まれます。各ステップは、専用のヘルパー機能を使用して構成されます。
- **Orchestration** — 自動処理モード `Live`、`Background`、または `Disabled` は、手動介入なしで、ナレッジベースとソースデータの同期を自動的に維持します。

3.4.3 サポートされているモデル

AIDB 拡張機能は、HuggingFace Postgres ノードでローカルに実行されているさまざまな Open Encoder LLM と OpenAI エンコーダクラウド API 経由をサポートしています。aidb.model_providers テーブルで利用可能なオプションをすべて表示できます。

3.5 Pipeline steps

AI パイプラインの各ステップは、その操作タイプとオプションの構成オブジェクトによって定義されます。オペレーション名を文字列として aidb.create_pipeline() の step_N パラメーターに渡し、構成を JSONB として対応する step_N_options パラメーターに渡します。

```
SELECT aidb.create_pipeline(
  name          => my_pipeline,
  source        => my_source_table,
  source_key_column => id,
  source_data_column => content,
  step_1        => ChunkText,
  step_1_options => aidb.chunk_text_config(desired_length => 120, max_length => 150),
  step_2        => KnowledgeBase,
  step_2_options => aidb.knowledge_base_config(model => bert, data_format => Text)
);
```

以下で説明するヘルパーファンクションを使用して、各操作タイプの step_N_options 値を構築します。すべてのヘルパーファンクションは、JSONB 構成オブジェクトを結果ます。

3.5.1 ChunkText

LLM コンテキストウィンドウ内に収まるように、テキストを小さなセグメントに分割します。

ヘルパーファンクション aidb.chunk_text_config()

Parameter	Type	Default	Description
<i>desired_length</i>	<i>INTEGER</i>	Required	ターゲットチャンクサイズ。単位は `strategy` に依存します。
<i>max_length</i>	<i>INTEGER</i>	<i>NULL</i>	最大チャンクサイズ。省略した場合、`desired_length` は厳密な上限です。
<i>overlap_length</i>	<i>INTEGER</i>	<i>NULL</i>	連続するチャンク間でオーバーラップするコンテンツの量。デフォルト 0 オーバーラップなし。

次のページに続く

表 6 – 前のページからの続き

Parameter	Type	Default	Description
<i>strategy</i>	<i>TEXT</i>	<i>NULL</i>	文字ベースの場合は`chars`デフォルト、ワードベースのチャンクの場合は`words` ¹ 。

基本的なチャンクの例

```
SELECT aidb.create_pipeline(
  name          => chunk_pipeline,
  source        => source_table,
  source_key_column => id,
  source_data_column => content,
  step_1        => ChunkText,
  step_1_options => aidb.chunk_text_config(
    desired_length => 100,
    max_length     => 150,
    overlap_length => 20,
    strategy       => words
  )
);
```

結果

ChunkText は、part_id 列を導入することにより、データの形状を変換します。各ソース行は、チャンクごとに1つずつ、複数の出力行を生成する場合があります。

3.5.2 ParseHtml

HTML 文字列から可読テキストを抽出し、論理構造を保持しながらタグを削除します。

ヘルパー関数 `aidb.html_parse_config()`

Parameter	Type	Default	Description
<i>method</i>	<i>TEXT</i>	<i>NULL</i>	プレーンテキスト抽出の場合は`StructuredPlaintext`デフォルト、または階層を保持する場合は`StructuredMarkdown` ¹ 。

```
SELECT aidb.create_pipeline(
  name          => html_pipeline,
  source        => web_data_table,
  source_key_column => id,
  source_data_column => html_content,
```

(次のページに続く)

(前のページからの続き)

```

step_1          => ParseHtml,
step_1_options  => aidb.html_parse_config(method => StructuredMarkdown),
step_2          => ChunkText,
step_2_options  => aidb.chunk_text_config(desired_length => 120)
);

```

3.5.3 ParsePdf

非準拠または複雑なファイル进行处理するオプションを使用して、バイナリ PDF データからテキストを抽出します。

ヘルパー関数 `aidb.pdf_parse_config()`

Parameter	Type	Default	Description
<i>method</i>	<i>TEXT</i>	<i>NULL</i>	'Structured' デフォルト — PDF 仕様を使用してテキストブロックを識別します。
<i>allow_partial_parsing</i>	<i>BOOLEAN</i>	<i>NULL</i>	'true' デフォルトの場合、個々のページでエラーが発生した場合に解析を続行します。

結果の `part_id` 列は、各テキストブロックが抽出されたページインデックスにマップします。

```

SELECT aidb.create_pipeline(
  name          => pdf_pipeline,
  source         => pdf_files_table,
  source_key_column => id,
  source_data_column => pdf_data,
  step_1         => ParsePdf,
  step_1_options => aidb.pdf_parse_config(
    method          => Structured,
    allow_partial_parsing => true
  ),
  step_2         => KnowledgeBase,
  step_2_options => aidb.knowledge_base_config(model => bert, data_format =>
    →Text)
);

```

結果

`ParsePdf` は結果のネストを解除します。マルチページ PDF はページごとに 1 行を生成し、それぞれにページインデックスに対応する `part_id` があります。

3.5.4 PdfToImage

PDF ドキュメントの各ページを画像に変換します。ソース PDF がスキャンされた文書、画像の多いレイアウト、またはテキストレイヤーが存在しないまたは信頼できない PDF である場合、この手順を使用します。レンダリングされたページ画像は、テキスト抽出のために **PerformOcr** ステップに直接渡すことができます。

ネイティブテキストレイヤーを含むデジタル生成 PDF の場合、代わりに **ParsePdf** を使用します。より速く、OCR モデルは必要ありません。

オプション形式 オプションを JSON リテラルとして渡します。

Parameter	Type	Default	Description
<i>dpi</i>	<i>INTEGER</i>	<i>300</i>	各ページのレンダリングに使用される 1 インチあたりのドット数の解像度。値が高いほど、より鮮明な画像が生成されますが、ファイルサイズが大きくなります。
<i>first_page</i>	<i>INTEGER</i>	<i>NULL</i>	レンダリングする最初のページ 1 ベース。省略すると最初からレンダリングされます。
<i>last_page</i>	<i>INTEGER</i>	<i>NULL</i>	レンダリングする最後のページ。1 ベースを含みます。省略すると最後までレンダリングされます。
<i>max_pages</i>	<i>INTEGER</i>	<i>NULL</i>	レンダリングするページの最大数。`first_page` および `last_page` に関係なく、安全キャップとして機能します。
<i>render_annotations</i>	<i>BOOLEAN</i>	<i>true</i>	レンダリングされた出力に PDF の注釈コメントやフォームフィールドなどを含めるかどうか。
<i>format</i>	<i>JSONB</i>	<i>{"type": "png"}</i>	出力画像フォーマット。受け入れられる値 <code>{"type": "png"}</code> 、 <code>{"type": "jpeg"}</code> 。出力を `PerformOcr` に渡す場合は、 <code>{"type": "png"}</code> を使用します。

出力画像は、**format** オプションに応じて、PNG または JPEG 形式です。出力を **PerformOcr** ステップに渡すときは、**"png"** デフォルトを使用します — NVIDIA NIM PaddleOCR には、PNG または JPEG 入力が必要です。

PerformOcr を含むパイプラインでこのステップを使用する前に、OCR 対応モデルを登録します。

```
SELECT aidb.create_model(
  my_paddle_ocr_model,
  nim_paddle_ocr,
```

(次のページに続く)

```
credentials => {"api_key": "<NVIDIA_NIM_API_KEY>"}::JSONB
);
```

次に、パイプラインを作成します。

```
SELECT aidb.create_pipeline(
  name          => pdf_ocr_pipeline,
  source        => pdf_files_table,
  source_key_column => id,
  source_data_column => pdf_data,
  step_1        => PdfToImage,
  step_1_options => {"dpi": 150}::json,
  step_2        => PerformOcr,
  step_2_options => aidb.ocr_config(model => my_paddle_ocr_model),
  step_3        => KnowledgeBase,
  step_3_options => aidb.knowledge_base_config(model => bert, data_format =>
  ↳Text)
);
```

結果

PdfToImage は結果のネストを解除します。ソース PDF の各ページは、format オプションに応じて、PNG または JPEG 画像としてレンダリングされたページを含む 1 つの出力行を生成します。part_id 列は、ページインデックス 0 ベースにマップします。この出力形状は、入力としてイメージバイトを想定する PerformOcr ステップと直接互換性があります。

3.5.5 PerformOcr

NVIDIA NIM PaddleOCR などの OCR 対応 AI モデルを使用して、画像からテキストを抽出します。

ヘルパーファンクション aidb.ocr_config()

Parameter	Type	Default	Description
<i>model</i>	<i>TEXT</i>	Required	使用する登録済みの OCR モデルの名前。

この手順を使用する前に、OCR 対応モデルを登録します。

```
SELECT aidb.create_model(
  my_paddle_ocr_model,
  nim_paddle_ocr,
  credentials => {"api_key": "<NVIDIA_NIM_API_KEY>"}::JSONB
);
```

次に、パイプラインでそれを参照します。

```

SELECT aidb.create_pipeline(
  name           => ocr_pipeline,
  source         => images_table,
  source_key_column => id,
  source_data_column => image_data,
  step_1         => PerformOcr,
  step_1_options  => aidb.ocr_config(model => my_paddle_ocr_model),
  step_2         => KnowledgeBase,
  step_2_options  => aidb.knowledge_base_config(model => bert, data_format =>
  ↳Text)
);

```

結果

PerformOcr は結果のネストを解除します。単一の画像から、検出されたテキストブロックごとに 1 つずつ、複数の行が生成される場合があります。NVIDIA NIM プロバイダーは、現在 png および jpeg 形式のみをサポートしています。

3.5.6 SummarizeText

AI 言語モデルを使用して、長いテキストの一節の簡潔な概要を生成します。

ヘルパー関数 `aidb.summarize_text_config()`

Parameter	Type	Default	Description
<i>model</i>	<i>TEXT</i>	Required	要約に使用する登録モデルの名前。
<i>chunk_config</i>	<i>JSONB</i>	<i>NULL</i>	要約前に適用されるオプションのチャンク構成 <code>`aidb.chunk_text_config()`</code> から。
<i>prompt</i>	<i>TEXT</i>	<i>NULL</i>	概要をガイドするカスタムプロンプト。省略すると標準のプロンプトを使用します。
<i>strategy</i>	<i>TEXT</i>	<i>NULL</i>	<code>'append'</code> デフォルトはチャンクごとのサマリーを連結し、 <code>'reduce'</code> は反復圧縮します。
<i>reduction_factor</i>	<i>INTEGER</i>	<i>NULL</i>	<code>'reduce'</code> ストラテジーで使用されます。反復ごとにテキストをどの程度積極的に削減するかを制御しますデフォルトは 3 です。
<i>inference_config</i>	<i>JSONB</i>	<i>NULL</i>	オプションの実行時推論設定 <code>`aidb.inference_config()`</code> から。

```

SELECT aidb.create_pipeline(
  name          => summary_pipeline,
  source         => articles_table,
  source_key_column => id,
  source_data_column => body,
  step_1         => SummarizeText,
  step_1_options => aidb.summarize_text_config(
    model          => my_t5_model,
    chunk_config   => aidb.chunk_text_config(100, 120, 10, words),
    prompt         => Summarize the key points concisely,
    strategy       => reduce,
    reduction_factor => 3
  ),
  step_2         => KnowledgeBase,
  step_2_options => aidb.knowledge_base_config(model => bert, data_format =>
→Text)
);

```

3.5.7 ナレッジベース

処理されたテキストまたは画像データをベクター埋め込みに変換し、検索可能なナレッジベースに保存します。このステップは、その出力が後続のステップで入力として使用できない **VECTOR** タイプであるため、常にパイプラインの **最後のステップ** である必要があります。セマンティック検索またはハイブリッド検索を使用したナレッジベースの照会については、[Knowledge bases reference](#) を参照してください。

ヘルパーファンクション `aidb.knowledge_base_config()`

Parameter	Type	Default	Description
<i>model</i>	<i>TEXT</i>	Required	埋め込みモデルの名前。
<i>data_format</i>	<i>aidb.PipelineDataFor</i>	Required	<i>'Text'</i> または <i>'Image'</i> 。
<i>distance_operator</i>	<i>aidb.DistanceOperator</i>	<i>NULL</i>	類似度メトリック <i>L2</i> デフォルト <i>'Cosine'</i> 、または <i>'InnerProduct'</i>
<i>vector_index</i>	<i>JSONB</i>	<i>NULL</i>	<i>'aidb.vector_index_hnsw_config()'</i> などのベクターインデックスヘルパーでビルドされたベクターインデックス構成。

```

SELECT aidb.create_pipeline(
  name          => kb_pipeline,
  source         => source_table,
  source_key_column => id,
  source_data_column => content,
  step_1         => KnowledgeBase,
  step_1_options => aidb.knowledge_base_config(

```

(次のページに続く)

(前のページからの続き)

```

    model          => bert,
    data_format     => Text,
    distance_operator => Cosine,
    vector_index    => aidb.vector_index_hnsw_config(m => 16, ef_construction =>
↪ 64)
  )
);

```

`aidb.knowledge_base_config_from_kb(data_format)` を使用して、パイプラインを既存のナレッジベースに接続します。これにより、複数のパイプラインが1つのベクターテーブルを共有でき、ナレッジベースからモデル、距離演算子、およびベクターインデックスを継承します。[Multi-pipeline knowledge bases](#) を参照してください。

宛先テーブル

KnowledgeBase ステップは、`destination` または省略した場合は `pipeline_<pipeline_name>` で渡された名前で宛先テーブルを作成します。複数のパイプラインが KB を共有する場合、それらはすべて同じテーブルに書き込みます。

KnowledgeBase ステップは、`destination` パラメーターで名前付けの宛先テーブルを作成します。`destination` が設定されていない場合、テーブルはデフォルトの `pipeline_<pipeline_name>` に設定されます。複数のパイプラインがナレッジベースを共有する場合、それらはすべて同じテーブルに書き込みます。

Column	Type	Description
<code>id</code>	BIGSERIAL	主キー。
<code>pipeline_id</code>	INT	元のパイプラインへの参照。
<code>source_id</code>	TEXT	元のソースレコードの ID。
<code>part_ids</code>	BIGINT[]	データがチャンクまたは解析された場合、セグメントを追跡します。
<code>value</code>	VECTOR	pgvector 埋め込み。

ナレッジベースビュー、取得機能、およびライフサイクル動作については、[Knowledge bases reference](#) および [Knowledge bases reference](#) を参照してください。

完全なエンドツーエンドワークフローで一緒に使用されるパイプラインステップを確認するには、[Examples](#) を参照してください。

3.6 Pipelines API reference

このページでは、AI パイプラインの完全なパブリック API について説明します。API 全体で使用される型と列挙型、パイプラインの状態を検査するためのビュー、コア CRUD および実行機能、パイプラインステップとベクターインデックスの構成ヘルパー機能。モデル構成ヘルパーについては、[Models reference](#) を参照してください。

3.6.1 タイプ

helpb.PipelineAutoProcessing モード

パイプラインが新しいデータまたは変更されたデータを自動的に処理する方法を制御します。

```
CREATE TYPE PipelineAutoProcessingMode AS ENUM (
    Live,
    Background,
    Disabled
);
```

Value	Description
<i>Live</i>	Postgres トリガーを使用して、新しいデータが到着するとすぐに処理します。
<i>Background</i>	Postgres ワーカーを使用して、バックグラウンドでデータを継続的に処理します。
<i>Disabled</i>	自動処理はありません。`aidb.run_pipeline()`を使用して手動でトリガーします。

helpb.PipelineDataFormat

パイプラインが処理するソースデータの形式を指定します。

```
CREATE TYPE PipelineDataFormat AS ENUM (
    Text,
    Image,
    Pdf
);
```

Value	Description
<i>Text</i>	プレーンテキストデータ。
<i>Image</i>	画像データバイト。
<i>Pdf</i>	PDF ドキュメント。

promoteb.PipelineSourceType

パイプラインが読み取るデータソースの種類を示します。

```
CREATE TYPE PipelineSourceType AS ENUM (
    Table,
    Volume,
    Empty
);
```

Value	Description
<i>Table</i>	Postgres テーブルまたはビュー。
<i>Volume</i>	PGFS ストレージボリューム。
<i>Empty</i>	ソースなし。パイプラインは独自のデータを生成します。

promoteb.PipelineDestinationType

パイプラインの書き込み先の種類を示します。

```
CREATE TYPE PipelineDestinationType AS ENUM (
    Table,
    Volume,
    Empty
);
```

Value	Description
<i>Table</i>	Postgres テーブル。
<i>Volume</i>	PGFS ストレージボリューム。
<i>Empty</i>	目的地はありません。出力は破棄されます。

helpb.PipelineStepOperation

パイプラインステップによって実行されるオペレーションを定義します。

```
CREATE TYPE PipelineStepOperation AS ENUM (
    ChunkText,
    SummarizeText,
    ParseHtml,
    ParsePdf,
    PerformOcr,
    KnowledgeBase,
    PdfToImage,
    SemanticKB
);
```

Value	Description
<i>ChunkText</i>	テキストをより小さなチャンクに分割します。
<i>SummarizeText</i>	言語モデルを使用してテキストを要約します。
<i>ParseHtml</i>	HTML からテキストコンテンツを抽出します。
<i>ParsePdf</i>	PDF からテキストまたは画像を抽出します。
<i>PerformOcr</i>	画像で光学式文字認識を実行します。
<i>KnowledgeBase</i>	エンベディングを計算してナレッジベースに保存します。

[次のページに続く](#)

表 18 – 前のページからの続き

Value	Description
<i>PdfToImage</i>	PDF ページを画像に変換します。
<i>SemanticKB</i>	スキーマメタデータをセマンティックナレッジベースにインデックス付けします。

helpb.PipelineStatus

パイプラインの現在の処理状態を表します。

```
CREATE TYPE PipelineStatus AS ENUM (
  Stale
  Processing
  UpToDate
  NoResults
  Failed
  Unknown
  PartialErrors
  BlockingErrors
);
```

Value	Description
<i>Stale</i>	ソースデータが変更されたため、パイプラインを実行する必要があります。
<i>Processing</i>	パイプラインは現在実行中です。
<i>UpToDate</i>	すべてのソースデータは正常に処理されました。
<i>NoResults</i>	処理は完了しましたが、出力は生成されません。
<i>Failed</i>	最後の実行は失敗しました。
<i>PartialErrors</i>	一部のレコードの処理に失敗しました。他の人は成功しました。
<i>BlockingErrors</i>	パイプラインの実行に失敗します。
<i>Unknown</i>	ステータスを判断できません。

helpb.DistanceOperator

ベクトル類似性検索に使用される距離メトリックを指定します。

```
CREATE TYPE DistanceOperator AS ENUM (
  L2,
  InnerProduct,
  Cosine,
  L1,
  Hamming,
  Jaccard
);
```

Value	Description
<i>L2</i>	ユークリッド距離。
<i>InnerProduct</i>	内積。
<i>Cosine</i>	コサイン類似度。
<i>L1</i>	L1 マンハッタン距離。
<i>Hamming</i>	ハミング距離。
<i>Jaccard</i>	ジャッカード距離。

3.6.2 ドメイン

promoteb.pipeline_name_50

パイプライン名が 50 文字以下であることを適用する TEXT ドメイン。

promoteb.background_sync_interval

バックグラウンド同期間隔が 1 second と 2 days の間であることを強制する INTERVAL ドメイン。

3.6.3 ビュー

helpb.pipelines

`aidb.pipes` としてもアクセスできます。登録されたすべてのパイプラインとその構成、ソース、宛先、処理モード、ステップ定義をリストします。

Column	Type	Description
<i>id</i>	integer	内部パイプライン識別子。
<i>name</i>	text	パイプラインの名前。
<i>source_type</i>	<i>promoteb.PipelineSourceType</i>	ソースがテーブルかボリュームかどうか。
<i>source_schema</i>	text	ソーステーブルのスキーマ。
<i>source</i>	text	ソーステーブルまたはボリュームの名前。
<i>source_key_column</i>	text	ソース内の一意キーとして使用される列。
<i>source_data_column</i>	text	処理するデータを含む列。
<i>destination_type</i>	<i>promoteb.PipelineDestinationType</i>	デスティネーションがテーブルかボリュームかどうか。
<i>destination_schema</i>	text	宛先テーブルのスキーマ。
<i>destination</i>	text	宛先テーブルまたはボリュームの名前。
<i>destination_key_column</i>	text	宛先テーブルのキー列。

次のページに続く

表 21 – 前のページからの続き

Column	Type	Description
<i>destination_data_column</i>	text	処理されたデータが書き込まれる宛先の列。
<i>steps</i>	jsonb	パイプラインステップ定義の順序付けられた配列。
<i>auto_processing</i>	<i>helpb.PipelineAutoProcessing</i> モード	自動処理モード。
<i>batch_size</i>	integer	バッチごとに処理されたレコードの数。
<i>background_sync_interval</i>	interval	バックグラウンドモードでの実行間の間隔。
<i>owner_role</i>	text	このパイプラインを所有する Postgres ロール。

例

```
SELECT name, source, destination, auto_processing FROM aidb.pipelines;
```

promoteb.pipeline_metrics

aidb.pipem としてもアクセスできます。各パイプラインの現在の処理統計を表示します。

Column	Type	Description
<i>pipeline</i>	text	パイプラインの名前。
<i>auto_processing</i>	text	現在の自動処理モード。
<i>table: unprocessed rows</i>	bigint	テーブルソースの場合 まだ処理されていない行数。
<i>volume: scans completed</i>	bigint	ボリュームソースの場合 完了したフルスキャンの数。
<i>count(source records)</i>	bigint	ソース内のレコードの合計数。
<i>count(destination records)</i>	bigint	宛先のレコードの合計数。
<i>Status</i>	text	現在の パイプラインステータスの統合 。
<i>count(record errors)</i>	bigint	処理に失敗したレコードの合計数。
<i>count(blocking errors)</i>	bigint	パイプラインの実行を妨げたエラーの合計数。

例

```
SELECT * FROM aidb.pipeline_metrics;
```

__OUTPUT__

```

      pipeline      | auto_processing | table: unprocessed rows | volume: scans_
→ completed | count(source records) | count(destination records) | Status |
→ count(record errors) | count(blocking errors)
```

(次のページに続く)

3.6.4 ファンクション

ソース、最大 10 の連続した処理ステップ、およびオプションの宛先を使用して新しいパイプラインを作成します。

パラメーター

次のページに続く

表 23 – 前のページからの続き

Parameter	Type	Default	Description
<i>destination</i>	TEXT	NULL	宛先テーブルまたはボリュームの名前。
<i>auto_processing</i>	helpb.PipelineAutoProcessingMode	NULL	自動処理モード。
<i>batch_size</i>	INT	NULL	バッチごとに処理するレコードの数。
<i>back-ground_sync_interval</i>	INTERVAL	NULL	バックグラウンドでの実行間の間隔。1 秒から 2 日の間である必要があります。
<i>owner_role</i>	TEXT	NULL	このパイプラインを所有および実行するロール。
<i>step_1_options</i>	JSONB	NULL	ステップ 1 の構成適切な パイプラインステップ構成ヘルパー を使用します。
<i>step_2 ... step_10</i>	helpb.PipelineStepOptions	NULL	ステップ 2～10 のオペレーション。
<i>step_2_options ... step_10_options</i>	JSONB	NULL	手順 2～10 の構成。

返品

Column	Type	Description
<i>name</i>	text	作成されたパイプラインの名前。
<i>destination_type</i>	text	パイプラインの宛先のタイプ。
<i>destination_schema</i>	text	宛先のスキーマ。
<i>destination</i>	text	宛先の名前。
<i>destination_key_column</i>	text	宛先のキー列。
<i>destination_data_column</i>	text	宛先のデータ列。

例

```
-- Single-step pipeline: chunk text from a table into a destination table
SELECT aidb.create_pipeline(
  name          => my_chunker,
  source        => source_docs,
  source_key_column => id,
  source_data_column => body,
  destination   => chunked_docs,
  step_1        => ChunkText,
```

(次のページに続く)

(前のページからの続き)

```

    step_1_options      => aidb.chunk_text_config(200, 250, 25),
    auto_processing     => Live
);

-- Multi-step pipeline: parse PDF, then embed into a knowledge base
SELECT aidb.create_pipeline(
    name                => pdf_to_kb,
    source              => pdf_volume,
    destination         => my_kb,
    step_1              => ParsePdf,
    step_1_options      => aidb.pdf_parse_config(),
    step_2              => KnowledgeBase,
    step_2_options      => aidb.knowledge_base_config(my_model, Text),
    auto_processing     => Background,
    background_sync_interval => 60 seconds
);

```

promoteb.update_pipeline

既存のパイプラインの自動処理設定を更新します。

パラメーター

Parameter	Type	Default	Description
<i>name</i>	TEXT	Required	更新するパイプラインの名前。
<i>auto_processing</i>	helpb.PipelineAutoProcessingMode	NULL	新しい自動処理モード。
<i>batch_size</i>	INT	NULL	新しいバッチサイズ。
<i>background_sync_interval</i>	INTERVAL	NULL	新しいバックグラウンド同期間隔。

例

```

SELECT aidb.update_pipeline(my_chunker, auto_processing => Background, background_
    sync_interval => 5 minutes);

```

promoteb.delete_pipeline

パイプラインとその構成を削除します。ソースまたは宛先テーブルは削除しません。

パラメーター

Parameter	Type	Default	Description
<i>name</i>	TEXT	Required	削除するパイプラインの名前。

例

```
SELECT aidb.delete_pipeline(my_chunker);
```

promoteb.run_pipeline

auto_processing モードに関係なく、パイプラインをすぐに実行するように手動でトリガーします。

パラメーター

Parameter	Type	Default	Description
<i>pipeline_name</i>	TEXT	Required	実行するパイプラインの名前。

例

```
SELECT aidb.run_pipeline(my_chunker);
```

3.6.5 パイプラインステップ構成ヘルパー

これらのファンクションは、aidb.create_pipeline の step_N_options パラメーターで使用する JSONB 構成オブジェクトを結果ます。

promoteb.chunk_text_config

テキストをより小さなセグメントに分割するように ChunkText ステップを構成します。

パラメーター

Parameter	Type	Default	Description
<i>desired_length</i>	INTEGER	Required	ターゲットチャンクサイズ。
<i>max_length</i>	INTEGER	NULL	最大許可チャンクサイズ。
<i>overlap_length</i>	INTEGER	NULL	連続するチャンク間でオーバーラップするユニットの数。

次のページに続く

表 28 – 前のページからの続き

Parameter	Type	Default	Description
<i>strategy</i>	TEXT	NULL	チャンク単位 ' <i>chars</i> ' デフォルトまたは ' <i>words</i> '。

例

```
-- Chunk into ~200 character segments, max 250, with 25-character overlap
SELECT aidb.chunk_text_config(200, 250, 25, chars);
```

helpb.summarize_text_config

言語モデルを使用してテキストを要約するように SummarizeText ステップを構成します。

パラメーター

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	要約に使用するモデルの名前。
<i>chunk_config</i>	JSONB	NULL	要約する前に適用されるオプションのチャンク構成 ' <i>aidb.chunk_text_config</i> ' から。
<i>prompt</i>	TEXT	NULL	概要をガイドするカスタムプロンプト。
<i>strategy</i>	TEXT	NULL	' <i>append</i> ' デフォルトまたは ' <i>reduce</i> '。
<i>reduction_factor</i>	INTEGER	NULL	' <i>reduce</i> ' ストラテジーの場合各リダクションパスの積極性デフォルト 3。
<i>inference_config</i>	JSONB	NULL	オプションの推論設定 ' <i>aidb.inference_config</i> ' から。

例

```
SELECT aidb.summarize_text_config(
  my_llm,
  chunk_config => aidb.chunk_text_config(100, 100, 10, words),
  prompt      => Summarize concisely,
  strategy    => reduce,
  reduction_factor => 4
);
```

promoteb.ocr_config

モデルを使用して画像からテキストを抽出するように **PerformOcr** ステップを構成します。

パラメーター

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	使用する OCR モデルの名前。

例

```
SELECT aidb.ocr_config(my_ocr_model);
```

promoteb.html_parse_config

HTML コンテンツからテキストを抽出するように **ParseHtml** ステップを構成します。

パラメーター

Parameter	Type	Default	Description
<i>method</i>	TEXT	NULL	使用する解析方法。`NULL` の場合、デフォルトの方法を使用します。

例

```
SELECT aidb.html_parse_config();
```

promoteb.pdf_parse_config

PDF ドキュメントからコンテンツを抽出するように **ParsePdf** ステップを構成します。

パラメーター

Parameter	Type	Default	Description
<i>method</i>	TEXT	NULL	使用する解析方法。`NULL` の場合、デフォルトの方法を使用します。

次のページに続く

表 32 – 前のページからの続き

Parameter	Type	Default	Description
<i>allow_partial_parsing</i>	BOOLEAN	NULL	`true` の場合、一部のページを解析できない場合に部分的な結果を返します。

例

```
SELECT aidb.pdf_parse_config(allow_partial_parsing => true);
```

helpb.knowledge_base_config

エンベディングを計算および保存するように KnowledgeBase ステップを構成します。

パラメーター

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	使用する埋め込みモデルの名前。
<i>data_format</i>	<i>helpb.PipelineDataFormat</i>	Required	埋め込まれるデータのフォーマット。
<i>distance_operator</i>	<i>helpb.DistanceOperator</i>	NULL	類似検索のための距離機能。デフォルト `L2`
<i>vector_index</i>	JSONB	NULL	ベクトルインデックス構成 <i>Vector インデックス構成ヘルパー</i> から。

例

```
SELECT aidb.knowledge_base_config(
  my_embedding_model,
  Text,
  distance_operator => Cosine,
  vector_index      => aidb.vector_index_hnsw_config(m => 16, ef_construction => 64)
);
```

fence.knowledge_base_config_from_kb

新しいものを作成するのではなく、既存のナレッジベースにパイプラインを接続するように KnowledgeBase ステップを構成します。

パラメーター

Parameter	Type	Default	Description
<i>data_format</i>	<i>helpb.PipelineDataFo</i>	Required	埋め込まれるデータのフォーマット。

例

```
SELECT aidb.knowledge_base_config_from_kb(Text);
```

helpb.inference_config

SummarizeText などの言語モデルステップで使用する推論構成オブジェクトをビルドします。すべてのパラメーターはオプション。

パラメーター

Parameter	Type	Default	Description
<i>system_prompt</i>	TEXT	NULL	各要求の先頭に追加されるシステムプロンプト。
<i>temperature</i>	DOUBLE PRECISION	NULL	サンプリング温度より高い=よりランダム。
<i>max_tokens</i>	INTEGER	NULL	生成するトークンの最大数。
<i>top_p</i>	DOUBLE PRECISION	NULL	nucleus サンプリングしきい値。
<i>seed</i>	BIGINT	NULL	再現可能な出力のランダムシード。
<i>repeat_penalty</i>	REAL	NULL	繰り返されたトークンのペナルティ。
<i>repeat_last_n</i>	INTEGER	NULL	リピートペナルティを適用する最近のトークンの数。
<i>thinking</i>	BOOLEAN	NULL	拡張推論を有効にしますサポートされているモデルのみ。
<i>extra_args</i>	JSONB	NULL	追加のプロバイダー固有の推論引数。

例

```
SELECT aidb.inference_config(
  system_prompt => You are a technical summarizer.,
  temperature   => 0.3,
  max_tokens    => 512
);
```

3.6.6 Vector インデックス構成ヘルパー

これらのファンクションは、`aidb.knowledge_base_config` の `vector_index` パラメーターの JSONB 構成を結果ます。

promoteb.vector_index_hnsw_config

HNSW インデックス pgvector を構成します。

パラメーター

Parameter	Type	Default	Description
<i>vector_data_type</i>	TEXT	NULL	ベクタストレージタイプ。
<i>m</i>	INTEGER	NULL	ノードごとの最大接続数デフォルト 16。
<i>ef_construction</i>	INTEGER	NULL	ビルド時の検索の深さデフォルト 64。
<i>ef_search</i>	INTEGER	NULL	クエリ時の検索の深さ。

 注釈

HNSW 100 2000 100000. 100 100 1000 1000 100000
aidb.vector_index_disabled_config() 100000000. !!!

次の表は、各 `distance_operator` 値が `pgvector ops` クラスにどのようにマッピングされるかを示しています。

<i>distance_operator</i>	Index ops class
<i>L2</i>	<i>vector_l2_ops</i>
<i>InnerProduct</i>	<i>vector_ip_ops</i>
<i>Cosine</i>	<i>vector_cosine_ops</i>
<i>L1</i>	<i>vector_l1_ops</i>

例

```
SELECT aidb.vector_index_hnsw_config(m => 16, ef_construction => 64);
```

promoteb.vector_index_ivflat_config

pgvector **IVFFlat** インデックスを構成します。

パラメーター

Parameter	Type	Default	Description
<i>vector_data_type</i>	TEXT	NULL	ベクタストレージタイプ。
<i>lists</i>	INTEGER	NULL	クラスター逆引きリストの数。
<i>probes</i>	INTEGER	NULL	クエリー時に検索するクラスターの数。

例

```
SELECT aidb.vector_index_ivfflat_config(lists => 100);
```

promoteb.vector_index_hsphere_optimized_config

HSphere 最適化インデックスを構成します。

パラメーター

Parameter	Type	Default	Description
<i>clusters</i>	INTEGER	Required	クラスターの数。
<i>precision_val</i>	DOUBLE PRECISION	Required	インデックス精度の値。
<i>vector_data_type</i>	TEXT	NULL	ベクタストレージタイプ。

例

```
SELECT aidb.vector_index_hsphere_optimized_config(clusters => 256, precision_val => 0.95);
```

promoteb.vector_index_disabled_config

ベクターインデックスの自動作成を無効にします。埋め込みディメンションが 2000 を超える場合、またはインデックスを手動で管理する場合にこれを使用します。

例

```
SELECT aldb.vector_index_disabled_config();
```

3.6.7 モデル構成ヘルパー

モデル構成ヘルパーは *Models reference* ページに移動しました。

第4章 Configuring external storage

AIDB パイプラインは、2 種類のデータソースから読み取ることができます。

- **Postgres テーブル** — `aidb.create_pipeline()` の `source` パラメーターを使用して、名前で直接テーブルを参照します。 [Creating pipelines](#) を参照してください。
- **外部ストレージボリューム** — S3 互換オブジェクトストア、Google Cloud Storage、Azure、またはローカルファイルシステムを PGFS を介して接続します。このページの残りの部分では、これを設定する方法について説明します。

外部ストレージは、Postgres ファイルシステム PGFS 拡張機能を介してアクセスされます。これは、外部ストレージを保存場所として Postgres にマップします。AIDB は、各保存場所を名前でパイプライン参照する **ボリューム**でラップします。ボリュームは **foreign table** であるため、標準の SQL でオブジェクトをリスト、読み取り、書き込み、および削除できます。

4.1 その仕組み

外部ストレージを AIDB パイプラインに接続することには、2つのオブジェクトが関係します。

- **PGFS ストレージ場所** — 外部ストレージプロバイダーの URI、資格情報、および接続オプションを定義します。 `pgfs.create_storage_location()` で作成されました。
- **AIDB ボリューム** — PGFS ストレージ場所のコンテンツを行として公開する外部テーブル。データフォーマット Text、Image、Pdf とオプションのサブパスを保存します。 `aidb.create_volume()` で作成されました。

ボリュームが存在したら、Postgres テーブルとまったく同じように、`aidb.create_pipeline()` の `source` として参照するか、SQL で直接照会します。

!!!注ストレージの場所を作成する前に、PGFS 拡張機能をインストールする必要があります。 [Configuring AIDB](#) を参照してください。

4.2 ステップ1 保存場所を作成する

`pgfs.create_storage_location()` を使用して、外部ストレージへの接続を定義します。 `uri` パラメーターは、ストレージバックエンドとパスを識別します。 `options` JSONB オブジェクトは、リージョンや資格情報などのプロバイダー固有の設定を保持します。

4.2.1 S3 互換オブジェクトストア

```

- - Private S3 bucket with credentials
SELECT pgfs.create_storage_location(
    name    => my_s3_location,
    uri     => s3://my-bucket/my-folder,
    options => {"region": "us-east-1", "access_key_id": "<key>", "secret_access_key":
→ "<secret>"}
);

- - Public S3 bucket (no credentials required)
SELECT pgfs.create_storage_location(
    name    => my_public_bucket,
    uri     => s3://aidb-rag-app,
    options => {"region": "eu-central-1", "skip_signature": "true"}
);

```

4.2.2 ローカルファイルシステム

ローカルファイルシステムアクセスの場合、保存場所を作成する前に `postgresql.conf` で許可されるパスパスを宣言します。PGFS は、セキュリティのためにこれらのパスへのアクセスを制限します。

```

# postgresql.conf

pgfs.allowed_local_fs_paths = /tmp/pgfs

```

Postgres を再起動した後、`file://` URI を使用して保存場所を作成します。

```

SELECT pgfs.create_storage_location(
    name => local_tmp_pgfs,
    uri  => file:///tmp/pgfs/
);

```

S3、GCS、および Azure のストレージ場所オプションの詳細については、PGFS documentation を参照してください。

!!!注 オブジェクトストアへの外向きの制限 PGFS は、ストレージの場所へのアクセス時に PGFS が到達できるホストを制限するオプションのアウトバウンドホワイトリスト `pgfs.egress_allowlist`、共有フォルバックとして `edb.egress_allowlist` をサポートします。これは、AIDB のモデルプロバイダートラフィックの許可リストとは無関係です。両方を一緒に構成できます。PGFS — Network egress および [AIDB — Network egress](#) を参照してください。

4.3 ステップ 2 ボリュームを作成する

`aidb.create_volume()` を使用して、PGFS 記憶場所を AIDB に公開する外部テーブルを作成します。ボリューム名は、パイプラインと SQL クエリーで参照する外部テーブルの名前です。

```
SELECT aidb.create_volume(
  name      => my_volume,
  server_name => my_s3_location,
  path      => /,
  data_format => Text
);
```

パラメーター

Parameter	Description
<i>name</i>	ボリュームの一意の名前。有効な引用符なしの PostgreSQL 識別子である必要があります小文字、文字またはアンダースコアで始まり、文字、数字、アンダースコアのみが含まれます-ダッシュやスペースは含まれません。外部テーブル名として使用されます。
<i>server_name</i>	アタッチする PGFS ストレージ場所の名前。
<i>path</i>	保存場所内のオプションのサブパス。以下の パス処理 を参照してください。
<i>data_format</i>	このボリュームのデータの種類。 <i>Text</i> 、 <i>Image</i> 、または <i>Pdf</i> のいずれか。パイプラインはこれを使用して、正しい解析ステップを選択します。

!!!注 `data_format` はメタデータです。オブジェクトを処理する方法を AIDB に指示しますが、フィルタリングはしません。ボリュームには、宣言されたフォーマットのオブジェクトのみが含まれていることを確認します。

4.3.1 パス処理

`path` 引数は、作成時に正規化されます。先頭と後続のスラッシュが削除され、後続のスラッシュが 1 つだけ追加されます。これは、次のすべてが `foo/` として保存されることを意味します。

- `foo`
- `/foo`
- `foo/`
- `/foo/`

NULL または `/` を渡すと、`/` として保存されます。これは、ストレージの場所のルートをターゲットとします。正規値とは、`aidb.volumes` で表示されるもの、およびオブジェクトキーの計算に使用されるものです。

4.3.2 例 S3 の PDF

```
SELECT pgfs.create_storage_location(
  pdf_bucket,
  s3://my-docs-bucket,
  options => {"region": "us-east-1", "access_key_id": "<key>", "secret_access_key":
  ↳ "<secret>"}
);

SELECT aidb.create_volume(pdf_volume, pdf_bucket, /, Pdf);
```

4.3.3 例 ローカルディレクトリ内の画像

```
SELECT pgfs.create_storage_location(local_tmp_pgfs, file:///tmp/pgfs/);

SELECT aidb.create_volume(ocr_input_volume, local_tmp_pgfs, ocr_input/, Image);
```

4.4 ステップ 3 ボリュームをパイプラインソースとして使用する

`aidb.create_pipeline()` の `source` パラメーターをボリューム名に設定します。ボリュームソースは、パイプラインの観点からテーブルソースと同じように動作します。

```
SELECT aidb.create_pipeline(
  name      => my_pdf_pipeline,
  source     => pdf_volume,
  step_1     => ParsePdf,
  step_2     => ChunkText,
  step_3     => KnowledgeBase,
  step_3_options => aidb.knowledge_base_config(bert_local, Text)
);
```

完全なワークスルーについては、[Create a pipeline](#) を参照してください。

4.5 ボリュームの管理

4.5.1 ボリュームのリスト

`aidb.volumes` ビューを介してすべてのボリュームをリストします。

```
SELECT * FROM aidb.volumes;

__OUTPUT__
schema | volume      | storage_location | path
-----+-----+-----+-----
public | pdf_volume  | pdf_bucket      | /
```

(次のページに続く)

(前のページからの続き)

```
public | text_bucket | text_bucket | aidb-rag-app/
(2 rows)
```

ビューは、schema、volume、storage_location、および path 正規化された 4 つの列を結果ます。

4.5.2 ボリュームの削除

```
SELECT aidb.delete_volume(my_volume);
```

!!!注 PGFS ストレージ場所を削除すると、その上に構築されたすべてのボリュームも削除されます。

4.6 SQL を使用してボリュームコンテンツにアクセスする

ボリュームは PGFS によってバッキングされる外部テーブルであるため、ストレージ場所のオブジェクトは行として公開されます。列は次のとおりです。

Column	Type	Description
<i>key</i>	TEXT	ボリュームの `path` を基準としたオブジェクトキーファイルパス。
<i>size</i>	BIGINT	バイト単位のオブジェクトサイズ。
<i>last_modified</i>	TIMESTAMPTZ	ストレージバックエンドによって報告された最終変更タイムスタンプ。
<i>e_tag</i>	TEXT	ストレージバックエンドによって報告されるエンティティタグ ETag 利用可能な場合。
<i>body</i>	BYTEA	オブジェクトの内容。クエリーに `WHERE key = '<exact-key>'` が含まれる場合にのみ入力されます。以下の注を参照してください。

!!!注 body 列は、WHERE key = '...' で単一のキーでフィルタリングを選択した場合にのみ取得されます。キーフィルタなしで body を選択すると、NULL が返されます。これは、ボリューム内のすべてのオブジェクトを誤ってダウンロードすることに対する安全対策です。

4.6.1 ボリューム内のオブジェクトのリスト

```
SELECT key, size, last_modified
FROM my_volume
ORDER BY key;
__OUTPUT__
      key          | size | last_modified
-----+-----+-----
Test 1.html        | 1093 | 2026-05-14 15:49:32+00
Test 2.html        | 1563 | 2026-05-14 15:50:02+00
```

(次のページに続く)

(前のページからの続き)

```
Test 3.html. | 1395 | 2026-05-14 15:49:32+00
(3 rows)
```

LIMIT はストレージバックエンドにプッシュダウンされるため、大きなバケットを安全に覗き見できます。

```
SELECT key, size FROM my_volume LIMIT 5;
```

4.6.2 単一オブジェクトの読み取り

正確なキーをフィルタリングしてオブジェクト本体を取得します。

```
- - Read a binary file as BYTEA
SELECT body FROM my_volume WHERE key = report.pdf;

- - Read a text file as text
SELECT convert_from(body, utf8)
FROM my_volume
WHERE key = notes.txt;
```

4.6.3 オブジェクトの書き込み、上書き、および削除

ボリュームは、標準の INSERT、UPDATE、および DELETE をサポートしています。body は BYTEA です。必要に応じてテキストを bytea にキャストします。

```
- - Upload a new object
INSERT INTO my_volume (key, body)
VALUES (hello.txt, Hello, world::bytea);

- - Overwrite an existing object
UPDATE my_volume
SET body = Hello again::bytea
WHERE key = hello.txt;

- - Delete an object
DELETE FROM my_volume WHERE key = hello.txt;
```

!!!注オブジェクトストアの書き込みはトランザクションではありません。成功した INSERT、UPDATE、または DELETE は、周囲の PostgreSQL トランザクションが中止された場合、ロールバックされません。

キーの名前の変更 UPDATE my_volume SET key = ... はサポートされていません。

第5章 Installing and configuring AIDB

AIDB は、インストールと構成が簡単にできるように設計されています。プロジェクトで AIDB の使用を開始するには、以下の手順に従ってください。

- [Installing AIDB](#)
- [Configuring AIDB](#)

5.1 次のステップ

AIDB のインストールと構成が完了したら、パイプラインと SQL ファンクションを強化するモデルを登録します。[Integrating models](#) を参照してください。

5.1.1 Configuring AIDB

AIDB を使用するには、`shared_preload_libraries` に追加し、データベースに拡張機能を作成し、インストールを確認します。

`shared_preload_libraries` の構成

1. `postgresql.conf` ファイルで、`aidb` を `shared_preload_libraries` パラメーターに追加します。

```
shared_preload_libraries = aidb
```

!!! 注: `shared_preload_libraries` に `aidb` を追加すると、データベースの起動時に `aidb` が自動的にインストールされます。この操作は、データベースの起動時にのみ行われます。

1. Postgres の `postgres` ユーザーで、

拡張機能を作成します

1. データベースに AIDB 拡張機能を作成します。

```
CREATE EXTENSION aidb CASCADE;
```

`CASCADE` オプションは、必要な `vector` 拡張機能がまだ存在していない場合は、自動的にインストールします。

!!!注 PGD ユーザー EDB Postgres 分散 PGD を実行している場合、拡張機能を作成した後に次を実行して、レプリケーションセットに AIDB カタログテーブルを登録します。

```
SELECT aidb.bdr_setup();
```

```
CREATE EXTENSION pgfs;
```

詳細については、PGFS documentation を参照してください。

プロキシ設定

ご使用の環境がプロキシを介してアウトバウンド HTTP トラフィックをルーティングする場合、Postgres 環境ファイルで HTTP_PROXY および HTTPS_PROXY 環境変数を設定します。コミュニティ Postgres を使用する Ubuntu では、このファイルは/etc/postgresql/<version>/main/environment にあります。

```
echo "HTTP_PROXY = http://<your-proxy-settings>/" | sudo tee -a /etc/postgresql/16/  
→main/environment  
echo "HTTPS_PROXY = http://<your-proxy-settings>/" | sudo tee -a /etc/postgresql/16/  
→main/environment
```

次に、Postgres を再起動して変更を有効にします。

```
sudo systemctl restart postgresql@16-main
```

<your-proxy-settings> をプロキシアドレスに置き換え、16 を Postgres バージョンに置き換えます。環境ファイルの正確な場所については、Postgres ディストリビューションのドキュメントを参照してください。

!!!注 エアギャップ環境は現在サポートされていません。

構成パラメーター

AIDB は、aidb.* 名前空間で Postgres 構成パラメーター GUC のセットを公開します。postgresql.conf で設定し、SET とセッションごとに、または-c と接続ごとに設定します。以下のすべてのパラメーターは USERSET です。ロールは自分のセッションでそれらを変更できます。

モデルのダウンロード

次のパラメーターは、AIDB が HuggingFace からローカルモデルファイルをダウンロードする方法を制御します該当するプロバイダーについては *Local models* を参照してください。

Parameter	Type	Default	Range	Description
<code>aidb.download_log_level</code>	string	<i>notice</i>	—	ルーチンのダウンロード進行状況メッセージダウンロードの開始/完了、バイト進行状況ハートビート、アダプターのロードラインなどの出力チャネルを制御します。有効な値 <i>notice</i> 、 <i>log</i> 、 <i>off</i> 。以下を参照してください。
<code>aidb.download_max_attempts</code>	integer	30	1-500	AIDB が異常終了するまでのファイルごとの最大合計ダウンロード試行初期要求と再試行。再試行動作については、 ローカルモデルのダウンロード を参照してください。

`helpb.download_log_level`

ルーチンのダウンロード進行状況メッセージの送信先を制御します。警告とエラーは影響を受けません。

Value	Behavior
<i>notice</i>	‘NOTICE’レベルでメッセージをクライアントに送信します。‘psql’のインタラクティブセッションに役立ちます。
<i>log</i>	メッセージをサーバーログのみに送信します。‘log_min_messages’の対象。これは、クライアント出力がコールドおよびウォームキャッシュにわたって安定する必要があるバッチジョブとテストに使用します。
<i>off</i>	ルーチンの進行状況メッセージを完全に抑制します。

例

```
SET aidb.download_log_level = log;
```

`quietb.download_max_attempts`

ファイルごとのダウンロード試行の合計数を制限します。ダウンローダーは、連続した試行で進行状況がゼロの場合、早期に中断するため、このパラメーターは主に過剰な再試行ループに対する安全上限です。

CI またはスクリプト化されたバッチジョブでのフェイルファスト動作の場合、これを 3 または 5 などに下げます。損失の多いリンク上の非常に大きなファイルの場合にのみこの値を上げます。

例

```
SET aidb.download_max_attempts = 5;
```

ネットワーク下り

次のパラメーターは、AIDB からのアウトバウンド HTTP トラフィック、モデルプロバイダー API の呼び出しと HuggingFace からのモデルファイルのダウンロードの両方を制限します。設定しない場合、制限は適用されず、AIDB は任意のホストに到達できます。

Parameter	Type	Default	Description
<code>aidb.egress_allowlist</code>	string	unset	AIDB が到達できるホストおよび/または CIDR レンジのコンマ区切りリスト。設定すると、リストにない宛先への送信要求は、接続が確立される前に拒否されます。
<code>edb.egress_allowlist</code>	string	unset	拡張機能ごとの <code>`aidb.egress_allowlist`</code> が設定されていない場合に、すべての EDB 拡張機能で使用される共有フォールバックホワイトリスト。AIDB と他の EDB 拡張機能 PGFS などが同じネットワークポリシーを共有する場合に役立ちます。

エン트리フォーマット

コンマ区切りの各エントリは、次のいずれかです。

Form	Matches
<code>host.example.com</code>	Exact hostname match (case-insensitive).
<code>.example.com</code>	Any subdomain of <code>example.com</code> (does not match <code>example.com</code> itself).
<code>203.0.113.10</code>	Exact IPv4 or IPv6 literal in the destination URL.
<code>203.0.113.0/24</code>	IPv4 CIDR range; matches IP literals in that range. Does not match hostnames.
<code>2001:db8::/32</code>	IPv6 CIDR range.

エントリの周囲の空白は無視されます。CIDR エントリは IP リテラルに対してのみチェックされます。ホスト名はマッチングのために解決されません。

解決順序

アウトバウンド要求ごとに、AIDB は最初に `aidb.egress_allowlist` をチェックします。設定されていない場合、AIDB は `edb.egress_allowlist` にフォールバックします。どちらも設定されていない場合、チェックは適用されません。リダイレクトたとえば、CDN ホストへの HuggingFace リダイレクトなどは、同じ許可リストに対して再チェックされます。リストにないホストへのリダイレクトは拒否されます。

例

OpenAI および HuggingFace とその CDN への直接呼び出しのみを許可します。

```
aidb.egress_allowlist = api.openai.com, huggingface.co, .huggingface.co, cdn-lfs.
→huggingface.co
```

要求が拒否されると、AIDB はアクティブなパラメーターを名前付けるエラーを発生させて、オペレーターがリストを調整できるようにします。

!!!注 PGFS は、`pgfs.egress_allowlist` の下で同じ許可リストメカニズムを実装し、PGFS からクラウドオブジェクトストアエンドポイント S3、Google Cloud Storage、Azure Blob へのトラフィックに適用されます。`edb.egress_allowlist` は、両方の拡張機能の共有フォールバックです。1つのリストを AIDB と PGFS に一緒に適用するように設定するか、それぞれが異なるポリシーが必要な場合に拡張ごとのパラメーターを使用します。PGFS — Network egress を参照してください。

AIDB パイプラインが外部ストレージから読み取る場合、両方の許可リストに関連します。AIDB はモデルプロバイダーと HuggingFace トラフィックを制御し、PGFS はオブジェクトストアへの接続を制御します。AIDB と PGFS がどのように連携するかについては、[Configuring external storage](#) を参照してください。

`psql` で `\dx` を実行し、出力で `aidb`、`vector`、`pgfs` インストールされている場合を探すことにより、拡張機能がインストールされていることを確認します。

```
aidb=# \dx
__OUTPUT__
```

List of installed extensions			
Name	Version	Schema	Description
aidb	7.4.0	aidb	aidb: makes it easy to build AI applications with postgres
pgfs	1.0.6	pgfs	pgfs: enables access to filesystem-like storage locations
vector	0.8.0	public	vector data type and ivfflat and hnsw access methods

5.1.2 Installing AIDB

AIDB は、スタンドアロンの PostgreSQL 拡張機能です。エージェントファクトリーまたはハイブリッドマネージャーに依存せずに、Postgres インスタンスに直接インストールします。以下の手順では、スタンドアロンパッケージとしてインストールする手順を説明します。

!!!note Agent Factory を介して展開している場合、AIDB はそのセットアップの一部としてバンドルされ、構成されます。代わりに Agent Factory インストールガイドを参照してください。

AIDB は、使用している Postgres ディストリビューションと同じプラットフォームでサポートされています。詳細については、以下を参照してください。

- [PostgreSQL product compatibility](#)

- EDB Postgres Advanced Server product compatibility
- EDB Postgres Extended Server product compatibility

前提条件

インストールプロセスを開始する前に

1. リポジトリのセットアップリポジトリのセットアップは1回限りのタスクです。すでにリポジトリを設定している場合、この手順を実行する必要はありません。リポジトリを設定するには、[EDB repositories](#) に移動し、そこに提供されている指示に従ってください。
2. Postgres のフレーバーがまだインストールされていない場合は、インストールします。

- *Installing PostgreSQL*

Installing EDB Postgres Advanced Server

Installing EDB Postgres Extended Server

3. pgvector 拡張機能をインストールします。

AIDB は、ベクターデータ管理に pgvector 拡張機能に依存しています。

インストール手順については、*Installing pgvector* を参照してください。

パッケージをインストールする

Package install コマンドの構文は次のとおりです。

```
# For Debian/Ubuntu:

sudo apt-get install edb-<postgres><postgres_version>-aidb

# For Linux:

sudo dnf install edb-<postgres><postgres_version>-aidb
```

どこで

- <postgres> は、使用している Postgres のディストリビューションです。

Postgres distribution	Value
PostgreSQL	pg
EDB Postgres Advanced Server (EPAS)	as
EDB Postgres Extended	pgextended

- <postgres_version> は、使用している Postgres のバージョンです。

たとえば、Debian/Ubuntu に EDB Postgres Advanced Server 17 の AIDB をインストールするには

```
sudo apt-get install edb-as17-aidb
```

例、RHEL8 に AIDB for PostgreSQL 16 をインストールするには

```
sudo dnf install edb-pg16-aidb
```

AIDB パッケージをインストールした後、次の手順については [Configuring AIDB](#) を参照してください。

第6章 Integrating models

モデルは、AIDB のコア操作-埋め込み生成、テキストチャンク、要約、OCR、補完を強化する AI エンジンです。AI 操作を実行するすべてのパイプラインステップと SQL ファンクションでは、実行前にモデルを登録して到達可能にする必要があります。

モデルを追加する場合 AIDB のインストールと構成の後、それらを参照するパイプラインまたはナレッジベースを作成する前に、モデルを登録します。AIDB のビルトインローカルモデル BERT や CLIP などを使用している場合、登録は必要ありません。すぐに利用できます。外部 API OpenAI、NVIDIA NIM、Gemini またはカスタム HuggingFace バリエーションの場合、最初に `aidb.create_model()` を使用してモデルを登録します。

Page	What it covers
<i>Local models</i>	Built-in local models (BERT, CLIP, T5, Llama) and how to register custom variants
<i>Support matrix</i>	Pre-vetted HuggingFace model variants confirmed to work with AIDB's local providers
<i>External model connections</i>	Connecting to OpenAI-compatible APIs, NVIDIA NIM, Google Gemini, and OpenRouter

6.1 External model connections

外部モデルは、Postgres インスタンスの外部でホストされる API ベースのモデルです。AIDB は、`aidb.create_model()` にモデルを登録し、プロバイダー名、構成オブジェクト、および API キーなどの資格情報を必要に応じて提供することにより、それらに接続します。

登録されると、外部モデルはビルトインローカルモデルとまったく同様に、AIDB SQL ファンクションまたはパイプラインステップで名前によって使用されます。

6.1.1 OpenAI 互換エンドポイント

AIDB は、OpenAI API を実装する埋め込みまたはチャット/補完サービスをサポートしています。これには、OpenAI 自分自身、Ollama のようなセルフホストサーバー、およびその他の準拠エンドポイントが含まれます。

エンベディング

`aidb.embeddings_config()` とともに `openai_embeddings` プロバイダーを使用します。url パラメーターをオーバーライドして、要求を OpenAI 互換サーバーにリダイレクトします。

```

- - OpenAI
SELECT aidb.create_model(
  my_openai_embedder,
  openai_embeddings,
  config => aidb.embeddings_config(
    model    => text-embedding-3-small,
    api_key  => sk-...,
    url      => https://api.openai.com/v1
  )
);

- - Local Ollama server
SELECT aidb.create_model(
  my_ollama_embedder,
  openai_embeddings,
  config => aidb.embeddings_config(
    model => llama3.2,
    url   => http://llama.local:11434/v1/embeddings
  )
);

```

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	API が想定するモデル識別子。
<i>api_key</i>	TEXT	NULL	認証用の API キー。
<i>url</i>	TEXT	NULL	API エンドポイント URL。デフォルトでは OpenAI のエンドポイントです。
<i>basic_auth</i>	TEXT	NULL	基本的な認証資格情報 <i>user:password</i> 。
<i>max_concurrent_requests</i>	INTEGER	NULL	エンドポイントへの最大同時要求。
<i>max_batch_size</i>	INTEGER	NULL	バッチ要求ごとの入力の最大数。
<i>input_type</i>	TEXT	NULL	エンコードの入力タイプヒントプロバイダー固有。
<i>input_type_query</i>	TEXT	NULL	クエリエンコーディングの入力タイプヒントプロバイダー固有。
<i>is_hcp_model</i>	BOOLEAN	NULL	HCP で実行されているモデルを参照する場合、`true` に設定します。

チャット完了

`aidb.completions_config()` とともに `openai_completions` プロバイダーを使用します。 `url` パラメーターは同じように機能します。OpenAI の場合は省略するか、他の準拠サーバーの場合はオーバーライドします。

```
SELECT aidb.create_model(
  my_openai_llm,
  openai_completions,
  config => aidb.completions_config(
    model      => gpt-4o,
    api_key     => sk-...,
    temperature => 0.2
  )
);
```

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	モデル識別子。
<i>api_key</i>	TEXT	NULL	認証用の API キー。
<i>url</i>	TEXT	NULL	API エンドポイント URL。デフォルトでは OpenAI のエンドポイントです。
<i>basic_auth</i>	TEXT	NULL	基本的な認証資格情報 <i>user:password</i> 。
<i>temperature</i>	DOUBLE PRECISION	NULL	サンプリング温度。
<i>top_p</i>	DOUBLE PRECISION	NULL	nucleus サンプリングしきい値。
<i>seed</i>	BIGINT	NULL	再現可能な出力のランダムシード。
<i>system_prompt</i>	TEXT	NULL	すべての要求の前に追加されるデフォルトのシステムプロンプト。
<i>max_tokens</i>	JSONB	NULL	最大トークン構成 <code>`aidb.max_tokens_config()`</code> から。
<i>thinking</i>	BOOLEAN	NULL	拡張推論を有効にしますサポートされているモデルのみ。
<i>max_concurrent_requests</i>	INTEGER	NULL	エンドポイントへの最大同時要求。
<i>extra_args</i>	JSONB	NULL	追加のプロバイダー固有の引数。
<i>is_hcp_model</i>	BOOLEAN	NULL	HCP で実行されているモデルを参照する場合、 <code>`true`</code> に設定します。

!!!注 pgvector インデックスを使用する場合、[pgvector limits indexed vectors to 2000 dimensions](#) に注意してください。選択したモデルがより多くのディメンションを生成する場合、パイプラインステップで `aidb.vector_index_disabled_config()` を使用し、インデックスを手動で管理します。

6.1.2 NVIDIA NIM

AIDB は、build.nvidia.com でホストされる NVIDIA NIM マイクロサービスと、独自の環境で実行されている NIM インスタンスをサポートしています。

テキスト生成 NIM 補完

モデル識別子と NIM API キーで `nim_completions` プロバイダーを使用します。

```
SELECT aidb.create_model(
  my_nim_llm,
  nim_completions,
  {"model": "meta/llama-3.3-70b-instruct"}::JSONB,
  {"api_key": "<NIM API KEY>"}::JSONB
);
```

独自の環境で実行されている NIM インスタンスを使用するには、NIM エンドポイントを指す構成に `url` キーを追加します。

登録後、`aidb.decode_text()` でモデルを実行します。

```
SELECT aidb.decode_text(my_nim_llm, Tell me a short, one sentence story);
```

__OUTPUT__
decode_text As the clock struck midnight, a single tear fell from the porcelain dolls eye. (1 row)

マルチモーダルエンベディング NIM CLIP

共同テキストと画像の埋め込みには、`aidb.nim_clip_config()` とともに `nim_clip` プロバイダーを使用します。

```
SELECT aidb.create_model(
  my_nim_clip,
  nim_clip,
  config => aidb.nim_clip_config(
    api_key => nvapi-...,
    model   => nvidia/nvclip
  )
);
```

`aidb.nim_clip_config()` パラメーター

Parameter	Type	Default	Description
<i>api_key</i>	TEXT	NULL	NIM 認証用の API キー。
<i>model</i>	TEXT	NULL	NIM CLIP モデル識別子。
<i>url</i>	TEXT	NULL	NIM エンドポイント URL のオーバーライド。
<i>basic_auth</i>	TEXT	NULL	基本的な認証資格情報。
<i>is_hcp_model</i>	BOOLEAN	NULL	モデルが HCP で実行されている場合、`true` に設定します。

OCR NIM OCR

`nim_ocr` プロバイダーを `aidb.nim_ocr_config()` とともに使用して、画像からテキストを抽出します。

```
SELECT aidb.create_model(
  my_nim_ocr,
  nim_paddle_ocr.,
  config => aidb.nim_ocr_config(api_key => nvapi-...)
);
```

`aidb.nim_ocr_config()` パラメーター

Parameter	Type	Default	Description
<i>api_key</i>	TEXT	NULL	NIM 認証用の API キー。
<i>model</i>	TEXT	NULL	NIM OCR モデル識別子。
<i>url</i>	TEXT	NULL	NIM エンドポイント URL のオーバーライド。
<i>basic_auth</i>	TEXT	NULL	基本的な認証資格情報。
<i>is_hcp_model</i>	BOOLEAN	NULL	モデルが HCP で実行されている場合、`true` に設定します。

リランキング NIM リランキング

`nim_reranking` プロバイダーを `aidb.nim_reranking_config()` とともに使用して、関連性によって検索結果を再スコアします。`aidb.rerank_text()` での使用については、[Text reranking](#) を参照してください。

```
SELECT aidb.create_model(
  my_reranker,
  nim_reranking,
  config => aidb.nim_reranking_config(
    api_key => nvapi-...,
    model   => nvidia/nv-rerankqa-mistral-4b-v3
  )
);
```

`aidb.nim_reranking_config()` パラメーター

Parameter	Type	Default	Description
<i>api_key</i>	TEXT	NULL	NIM 認証用の API キー。
<i>model</i>	TEXT	NULL	NIM リランキングモデル識別子。
<i>url</i>	TEXT	NULL	NIM エンドポイント URL のオーバーライド。
<i>basic_auth</i>	TEXT	NULL	基本的な認証資格情報。
<i>is_hcp_model</i>	BOOLEAN	NULL	モデルが HCP で実行されている場合、`true` に設定します。

NIM API キーを取得するには、build.nvidia.com でアカウントを作成し、モデルを選択して、モデルのページからキーを生成します。

6.1.3 Google Gemini

`aidb.gemini_config()` とともに `gemini` プロバイダーを使用します。Google API キーを最初の引数として渡します。

```
SELECT aidb.create_model(
  my_gemini,
  gemini,
  config => aidb.gemini_config(
    AIza...,
    model => gemini-2.0-flash
  )
);
```

`aidb.gemini_config()` パラメーター

Parameter	Type	Default	Description
<i>api_key</i>	TEXT	Required	Google API キー。
<i>model</i>	TEXT	NULL	Gemini モデル識別子たとえば <i>gemini-2.0-flash</i> 。
<i>url</i>	TEXT	NULL	API エンドポイント URL オーバーライド。
<i>max_concurrent_requests</i>	INTEGER	NULL	API への最大同時要求。
<i>thinking_budget</i>	INTEGER	NULL	拡張思考トークンの予算 Gemini 2.x モデルのみ。

6.1.4 OpenRouter

`OpenRouter` は、複数のプロバイダーの 200 以上のモデルに統合 API ゲートウェイを提供します。AIDB は、`OpenRouter` を介したチャット補完と埋め込みの両方をサポートしています。

チャット完了

```
SELECT aidb.create_model(
  my_or_chat,
  openrouter_chat,
  config => aidb.openrouter_chat_config(
    anthropic/claude-3-5-haiku,
    api_key => sk-or-...
  )
);
```

aidb.openrouter_chat_config() パラメーター

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	OpenRouter モデル識別子。
<i>api_key</i>	TEXT	NULL	OpenRouter API キー。
<i>url</i>	TEXT	NULL	API エンドポイント URL オーバーライド。
<i>max_concurrent_requests</i>	INTEGER	NULL	最大同時要求。
<i>max_tokens</i>	JSONB	NULL	最大トークン構成 `aidb.max_tokens_config()`から。

エンベディング

```
SELECT aidb.create_model(
  my_or_embedder,
  openrouter_embeddings,
  config => aidb.openrouter_embeddings_config(
    mistral/mistral-embed,
    api_key => sk-or-...
  )
);
```

aidb.openrouter_embeddings_config() パラメーター

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	OpenRouter エンベディングモデル識別子。
<i>api_key</i>	TEXT	NULL	OpenRouter API キー。
<i>url</i>	TEXT	NULL	API エンドポイント URL オーバーライド。
<i>max_concurrent_requests</i>	INTEGER	NULL	最大同時要求。

次のページに続く

表 9 – 前のページからの続き

Parameter	Type	Default	Description
<i>max_batch_size</i>	INTEGER	NULL	バッチ要求ごとの最大入力。

すべての構成ヘルパー機能の完全なパラメーターの詳細については、[Pipelines API reference](#) を参照してください。

6.2 Local models

AIDB は 2 種類のローカルモデルをサポートしています。どちらも外部 API を必要とせずに Postgres ホストで直接実行されます。

- **ビルトインモデル** – AIDB は、サポートされている各モデルタイプのデフォルトバリエーションをバンドルし、自動的に登録します。使用を開始するために設定は必要ありません。
- **カスタムローカルモデル** – 別のモデルバリエーション、微調整モデル、または特定のローカルパスが必要な場合は、`aidb.create_model()` を使用してカスタムバージョンを登録できます。

6.2.1 ビルトインモデル

次のモデルは、すべての AIDB インストールで構成なしで使用できます。

Model name	Provider	Use
<i>bert</i>	<i>bert_local</i>	Text embeddings
<i>clip</i>	<i>clip_local</i>	Multimodal embeddings (text and image)
<i>t5</i>	<i>t5_local</i>	Text-to-text generation
<i>llama</i>	<i>llama_instruct_local</i>	Text-to-text generation
<i>dummy</i>	<i>dummy</i>	Testing and development (returns zeros)

AIDB SQL ファンクションまたはパイプラインステップ構成で、名前で直接使用します。

```
- - Text embedding with the built-in BERT model
SELECT aidb.encode_text(bert, The quick brown fox);

- - Image embedding with the built-in CLIP model
SELECT aidb.encode_image(clip, pg_read_binary_file('/tmp/photo.jpg')::BYTEA);

- - Text generation with the built-in T5 model
SELECT aidb.decode_text(t5, Translate to French: Hello, world.);
```

6.2.2 カスタムローカルモデル

別のモデルバージョン、微調整されたモデル、または特定のローカルパスが必要な場合、`aidb.create_model()` を使用してカスタムバリエーションを登録します。

```
SELECT aidb.create_model(
  name      => my_model,
  provider => <provider>,
  config    => <config_helper>(...)
);
```

Parameter	Type	Description
<i>name</i>	TEXT	このモデルの一意の名前。パイプラインおよび SQL ファンクションで参照するために使用されます。
<i>provider</i>	TEXT	モデルプロバイダー。AIDB が使用する推論エンジンを決定します。
<i>config</i>	JSONB	プロバイダー固有の構成、構成ヘルパー機能で構築されています。

BERT

`aidb.bert_config()` を使用して、HuggingFace モデル識別子、オプションのレビュー、およびオプションのローカルキャッシュディレクトリを指定します。

```
SELECT aidb.create_model(
  my_bert,
  bert_local,
  config => aidb.bert_config(sentence-transformers/all-MiniLM-L6-v2)
);
```

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	モデル名またはパス HuggingFace 識別子。
<i>revision</i>	TEXT	NULL	モデルのレビューまたはブランチ。
<i>cache_dir</i>	TEXT	NULL	モデルファイルをキャッシュするためのローカルディレクトリ。

クリップ

`aidb.clip_config()` を使用して、ジョイントテキストと画像埋め込みの CLIP ベースのマルチモーダルモデルを構成します。

```
SELECT aidb.create_model(
    my_clip,
    clip_local,
    config => aidb.clip_config(openai/clip-vit-base-patch32)
);
```

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	モデル名またはパス HuggingFace 識別子。
<i>revision</i>	TEXT	NULL	モデルのリビジョンまたはブランチ。
<i>cache_dir</i>	TEXT	NULL	モデルファイルをキャッシュするためのローカルディレクトリ。

T5

`aidb.t5_config()` を使用して、T5 テキスト対テキストモデルを構成します。T5 モデルは、バリエーションに応じて、翻訳、要約、質問応答などのタスクをサポートします。

```
SELECT aidb.create_model(
    my_t5,
    t5_local,
    config => aidb.t5_config(google/flan-t5-base)
);
```

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	モデル名またはパス。
<i>model_path</i>	TEXT	NULL	モデルの重みへの明示的なローカルパス、 <code>model</code> をオーバーライドします。
<i>cache_dir</i>	TEXT	NULL	モデルファイルをキャッシュするためのローカルディレクトリ。

ラマ

`aidb.llama_config()` を使用して、テキスト生成用の Llama ファミリー言語モデルを構成します。Llama モデルは、温度、top-p サンプルング、コンテキスト長、およびその他の生成パラメーターをサポートしています。

```
SELECT aidb.create_model(
  my_llama,
  llama_instruct_local.,
  config => aidb.llama_config(
    meta-llama/Llama-3.2-3B-Instruct,
    temperature => 0.5
  )
);
```

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	モデル名またはパス。
<i>cache_dir</i>	TEXT	NULL	モデルファイルをキャッシュするためのローカルディレクトリ。
<i>model_path</i>	TEXT	NULL	モデルの重みへの明示的なローカルパス、 <code>model</code> をオーバーライドします。
<i>temperature</i>	DOUBLE PRECISION	NULL	サンプリング温度。
<i>top_p</i>	DOUBLE PRECISION	NULL	nucleus サンプリングしきい値。
<i>seed</i>	BIGINT	NULL	再現可能な出力のランダムシード。

!!!注ローカルモデルファイルは HuggingFace からダウンロードされ、最初に使用するときにはキャッシュされます。Postgres プロセスには、ネットワークアクセスとキャッシュディレクトリへの書き込みアクセスが必要です。エアギャップ環境の場合、モデルファイルを事前ダウンロードし、`model_path` または `cache_dir` を使用してローカルコピーをポイントします。

6.2.3 ローカルモデルのダウンロード

ローカルモデルが初めて使用されるとき、またはそのキャッシュが欠落または破損しているとき、AIDB は HuggingFace から構成済みの `cache_dir` にモデルファイルをダウンロードします。ダウンローダーは進行状況メッセージを発行し、一時的なネットワーク障害を自動的に再試行します。

進行状況メッセージ

AIDB は、ファイルごとに、要求の開始時に `downloading` 行を生成し、終了時に `complete` 行を生成します。マルチギガバイトのファイルの場合、ハートビートラインは、累積バイト数と完了パーセントを約 30 秒ごとにレポートするため、大規模なダウンロード中にセッションがフリーズしたように見えません。

```
SELECT aidb.decode_text(t5, test);
__OUTPUT__
NOTICE: [t5_local] loading t5-small (rev refs/pr/15) -> /var/.../snapshots/refs/pr/15
NOTICE: [hf] downloading https://huggingface.co/t5-small/resolve/refs%2Fpr%2F15/
→config.json (expected: 1.2 KB)
NOTICE: [hf] complete: https://huggingface.co/t5-small/resolve/refs%2Fpr%2F15/config.
→json (1.2 KB) in 0.9s
NOTICE: [hf] downloading https://huggingface.co/t5-small/resolve/refs%2Fpr%2F15/
→model.safetensors (size unknown)
NOTICE: [hf] downloading t5-small/model.safetensors: 115.4 MB / 230.8 MB (50.0%)
NOTICE: [hf] downloading t5-small/model.safetensors: 216.4 MB / 230.8 MB (93.7%)
NOTICE: [hf] complete: https://huggingface.co/t5-small/resolve/refs%2Fpr%2F15/model.
→safetensors (230.8 MB) in 83.5s
```

デフォルトでは、これらの行は NOTICE レベルでクライアントに送信されます。aidb.download_log_level パラメーターを使用して、代わりにサーバーログにルーティングするか、完全に抑制します。

再試行と再開

マルチギガバイトのダウンロードの途中で接続をドロップする CDN は、要求を完全に失敗しません。

- 再試行ごとに、以前の試行が停止した場所から Range ヘッダーが送信されるため、進行状況はゼロからリスタートするのではなく、試行にわたって蓄積されます。
- AIDB は、「まだ新しいバイトを受信している」と「進行状況がゼロ」を区別し、aidb.download_max_attempts に対するストールした試行のみをカウントし、試行間の指数関数的なバックオフを使用します。
- psql からの Ctrl-C は、現在の要求がタイムアウトするのを待機する代わりに、再試行の間に着陸します。

ファイルごとの試行の合計数は、aidb.download_max_attempts デフォルト 30 によって制限されます。

検証と自己修復

AIDB は、HuggingFace によって宣伝されるサイズと SHA-256 ハッシュを使用してダウンロードされたファイルを検証し、これらの値を記録する完了した各ダウンロードの横に小さなサイドカーファイルを書き込みます。キャッシュヒットするたびに、AIDB はサイドカーに対してファイルサイズを再チェックするため、元のダウンロードの後に切り詰められたファイルは、使用される前にキャッチされます。

モデルのロード時に、キャッシュされたファイルが破損していると検出されると、AIDB は影響を受けるスナップショットをパージし、自動的に 1 回再ダウンロードします。

6.2.4 登録されたモデルを使用する

登録されると、モデルは AIDB ファンクションで名前参照されます。カスタムモデルとビルトインモデルは互換性があります。

```
-- Use a custom BERT model for embedding
SELECT aidb.encode_text(my_bert, Hello world);

-- Use a custom BERT model in a pipeline step
SELECT aidb.create_pipeline(
  name => my_pipeline,
  source => my_source_table,
  source_key_column => id,
  source_data_column => content,
  step_1 => KnowledgeBase,
  step_1_options => aidb.knowledge_base_config(model => my_bert, data_format =>
  ↳Text)
);
```

すべてのモデル構成ヘルパーファンクションの完全なパラメーターの詳細については、[Models reference](#) を参照してください。

AIDB のローカルプロバイダーで動作するようにテストおよび検証された HuggingFace モデルバリエーションのリストについては、[Support matrix](#) を参照してください。

6.3 Support matrix

6.3.1 ローカル組み込みモデル

次の HuggingFace モデルバリエーションは、AIDB のローカルモデルプロバイダーで動作するようにテストおよび検証されています。aidb.create_model() を呼び出すときに、これらの識別子は、対応する構成ヘルパーで使用できます。

BERT

Model name	HuggingFace identifier	Config string
bert_all-MiniLM-L12-v1	sentence-transformers/all-MiniLM-L12-v1	aidb.bert_config('sentence-transformers/all-MiniLM-L12-v1', revision => 'main')
bert_all-MiniLM-L6-v1	sentence-transformers/all-MiniLM-L6-v1	aidb.bert_config('sentence-transformers/all-MiniLM-L6-v1', revision => 'main')
bert_all-MiniLM-L6-v2	sentence-transformers/all-MiniLM-L6-v2	aidb.bert_config('sentence-transformers/all-MiniLM-L6-v2', revision => 'main')

次のページに続く

表 16 – 前のページからの続き

Model name	HuggingFace identifier	Config string
<i>bert_all-distilroberta-v1</i>	<i>sentence-transformers/all-distilroberta-v1</i>	<i>aidb.bert_config('sentence-transformers/all-distilroberta-v1', revision => 'main')</i>
<i>bert_msmarco-bert-base-dot-v5</i>	<i>sentence-transformers/msmarco-bert-base-dot-v5</i>	<i>aidb.bert_config('sentence-transformers/msmarco-bert-base-dot-v5', revision => 'main')</i>
<i>bert_multi-qa-MiniLM-L6-cos-v1</i>	<i>sentence-transformers/multi-qa-MiniLM-L6-cos-v1</i>	<i>aidb.bert_config('sentence-transformers/multi-qa-MiniLM-L6-cos-v1', revision => 'main')</i>
<i>bert_multi-qa-MiniLM-L6-dot-v1</i>	<i>sentence-transformers/multi-qa-MiniLM-L6-dot-v1</i>	<i>aidb.bert_config('sentence-transformers/multi-qa-MiniLM-L6-dot-v1', revision => 'main')</i>
<i>bert_paraphrase-TinyBERT-L6-v2</i>	<i>sentence-transformers/paraphrase-TinyBERT-L6-v2</i>	<i>aidb.bert_config('sentence-transformers/paraphrase-TinyBERT-L6-v2', revision => 'main')</i>
<i>bert_paraphrase-multilingual-MiniLM-L12-v2</i>	<i>sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2</i>	<i>aidb.bert_config('sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2', revision => 'main')</i>
<i>bert_paraphrase-multilingual-mpnet-base-v2</i>	<i>sentence-transformers/paraphrase-multilingual-mpnet-base-v2</i>	<i>aidb.bert_config('sentence-transformers/paraphrase-multilingual-mpnet-base-v2', revision => 'main')</i>

すべての BERT バリエントは、テキスト埋め込み `aidb.encode_text()` をサポートしています。

クリップ

Model name	HuggingFace identifier	Config string
<i>openai/clip-vit-base-patch32</i>	<i>openai/clip-vit-base-patch32</i>	<i>aidb.clip_config('openai/clip-vit-base-patch32')</i>

CLIP は、テキスト埋め込み `aidb.encode_text()` と画像埋め込み `aidb.encode_image()` の両方をサポートしています。

ラマ

Model name	HuggingFace identifier	Config string
<i>HuggingFaceTB/SmolLM-1.7B-Instruct</i>	<i>HuggingFaceTB/SmolLM2-1.7B-Instruct</i>	<i>aidb.llama_config('HuggingFaceTB/SmolLM2-1.7B-Instruct', revision => 'main')</i>

次のページに続く

表 18 – 前のページからの続き

Model name	HuggingFace identifier	Config string
<i>HuggingFaceTB/SmolLM-135M-Instruct</i>	<i>HuggingFaceTB/SmolLM2-135M-Instruct</i>	<i>aidb.llama_config('HuggingFaceTB/SmolLM2-135M-Instruct', revision => 'main')</i>
<i>HuggingFaceTB/SmolLM-360M-Instruct</i>	<i>HuggingFaceTB/SmolLM2-360M-Instruct</i>	<i>aidb.llama_config('HuggingFaceTB/SmolLM2-360M-Instruct', revision => 'main')</i>
<i>TinyLlama/TinyLlama-1.1B-Chat-v1.0</i>	<i>TinyLlama/TinyLlama-1.1B-Chat-v1.0</i>	<i>aidb.llama_config('TinyLlama/TinyLlama-1.1B-Chat-v1.0', revision => 'main')</i>

すべての Llama バリエントは、テキスト生成 `aidb.decode_text()` をサポートしています。

T5

Model name	HuggingFace identifier	Config string
<i>t5-small</i>	<i>t5-small</i>	<i>aidb.t5_config('t5-small', revision => 'main')</i>
<i>t5-base</i>	<i>t5-base</i>	<i>aidb.t5_config('t5-base', revision => 'main')</i>
<i>t5-large</i>	<i>t5-large</i>	<i>aidb.t5_config('t5-large', revision => 'main')</i>

T5 バリエントは、テキストエンコード `aidb.encode_text()` とテキスト生成 `aidb.decode_text()` の両方をサポートしています。

!!!注これらは、AIDB で動作することが確認された事前審査されたバリエントです。これらに制限されません。互換性のある HuggingFace モデルは、適切な構成ヘルパーで使用できますが、ここにリストされているバリエントのみが EDB によってテストされています。

6.3.2 NIM モデル

AIDB は、複数のプロバイダーを介して NVIDIA NIM マイクロサービスをサポートしています。以下の表は、各プロバイダーの互換性のある NIM モデルをリストしています。**Default on HM** とマークされたモデルは、EDB Hybrid Manager のモデルサービングとともに出荷され、HM 展開ですぐに利用できます。別の展開は必要ありません。

NIM 補完 `nim_completions`

`aidb.decode_text()`、`aidb.decode_text_batch()` をサポート

NIM model identifier	Default on HM
<i>nvidia/nemotron-3-nano</i>	OK

次のページに続く

表 20 – 前のページからの続き

NIM model identifier	Default on HM
<i>nvidia/llama-3.3-nemotron-super-49b-v1</i>	OK
<i>ibm/granite-guardian-3.0-8b</i>	—
<i>ibm/granite-3.0-8b-instruct</i>	—
<i>ibm/granite-3.0-3b-a800m-instruct</i>	—
<i>meta/llama-3.3-70b-instruct</i>	—
<i>meta/llama-3.2-3b-instruct</i>	—
<i>meta/llama-3.2-1b-instruct</i>	—
<i>meta/llama-3.1-405b-instruct</i>	—
<i>meta/llama-3.1-70b-instruct</i>	—
<i>meta/llama-3.1-8b-instruct</i>	—
<i>meta/llama3-70b-instruct</i>	—
<i>meta/llama3-8b-instruct</i>	—
<i>nvidia/llama-3.1-nemotron-70b-instruct</i>	—
<i>nvidia/llama-3.1-nemotron-51b-instruct</i>	—
<i>nvidia/nemotron-mini-4b-instruct</i>	—
<i>nvidia/nemotron-4-340b-instruct</i>	—
<i>google/shieldgemma-9b</i>	—
<i>google/gemma-7b</i>	—
<i>google/codegemma-7b</i>	—

NIM エンベディング `nim_embeddings`

`aidb.encode_text()`、`aidb.encode_text_batch()` をサポート

NIM model identifier	Default on HM
<i>nvidia/llama-3.2-nemoretriever-300m-embed-v1</i>	OK
<i>nvidia/nv-embedqa-e5-v5</i>	—

NIM CLIP `nim_clip`

`aidb.encode_text()`、`aidb.encode_text_batch()`、`aidb.encode_image()` をサポート

NIM model identifier	Default on HM
<i>nvidia/nvclip</i>	OK

NIM OCR `nim_paddle_ocr`

`aidb.perform_ocr()` をサポート

NIM model identifier	Default on HM
<i>baidu/paddleocr</i>	OK

NIM リランキング `nim_reranking`

`aidb.rerank_text()` をサポート

NIM model identifier	Default on HM
<i>nvidia/llama-3.2-nv-rerankqa-1b-v2</i>	OK

!!!注 **Default on HM** としてマークされたモデルは、EDB Hybrid Manager のモデルサービングによって事前展開されます。他のすべてのモデルは、KServe を使用してセルフホストし、`aidb.create_model()` を使用して AIDB に登録できます。セットアップ手順については、[NVIDIA NIM](#) を参照してください。

第7章 Knowledge bases

ナレッジベース KB には、エンベディングとその構成モデル、距離演算子、ベクトルインデックスが保存されます。パイプラインはそれらに書き込み、`aidb.retrieve_text()` または `aidb.retrieve_key()` を使用して名前で照会します。

ナレッジベースは、単一のパイプラインと `KB` から始まります。同じナレッジベースに追加のパイプラインを接続できます。さまざまなソースまたは処理チェーンをすべて、1つの照会可能なベクトルテーブルへのエンベディングに貢献させます。

Page	What it covers
<i>Multi-pipeline knowledge bases</i>	Attach multiple pipelines to one KB; compatibility rules and deletion semantics
<i>Hybrid search</i>	Combining semantic search with relational filters and full-text search
<i>Examples</i>	End-to-end worked examples for table and volume sources

7.1 取得ファンクション

パイプラインが実行されたら、`aidb.retrieve_text()` または `aidb.retrieve_key()` を使用してナレッジベースを照会します。どちらもベクトルの類似度を使用して、正確なキーワードではなく意味に基づいて結果を検索し、クエリ入力として `TEXT` と `BYTEA` の両方をサポートしています。

7.2 検索ファンクションの流れ

取得ファンクションが呼び出されると、システムは内部で次の手順を実行します。

- **埋め込み** 入力クエリテキストまたは画像は、そのナレッジベース用に構成された特定の埋め込みモデルを使用してベクトルに変換されます。
- **類似性検索** ベクトル類似性検索は、ナレッジベースの内部ベクトルテーブルに対して実行され、上位 `K` の最近傍を見つけます。
- **ソースルックアップテキストのみ** `retrieve_text` の場合、システムはソーステーブルを識別し、一致したキーに対応する生のコンテンツを取得します。

7.2.1 `promoteb.retrieve_text()`

このファンクションは、最も近いベクトル一致に関連付けられた実際のソーステキストを取得する必要がある場合に使用します。

- **プロセス** ファンクションはクエリーを埋め込み、類似性検索を実行して、第 2 フェーズを実行して、`pipeline_id` を使用して元のテーブルからソーステキストをルックアップします。
- **戻り値** 以下を含む列のセット
- **key** ソーステーブルの識別子
- **value** 実際のソーステキスト
- **distance** 類似性スコア。通常、低いほど、より近い一致を示します。
- **part_ids** 一致した特定のチャンクまたはパーツを示す ID の配列。
- **pipeline_name** データを提供したパイプラインの名前。
- ****intermediate_steps**** ナレッジベースの前に発生するステップからのデータを含む JSONB 列。例、`ChunkText`。

7.2.2 promoteb.retrieve_key()

このファンクションは、完全なソースコンテンツではなく、一致の一意の識別子のみが必要な高パフォーマンスの検索に使用します。

- **戻り値** 以下を含む列のセット
- **key** ソーステーブルからの識別子
- **distance** 類似度スコア。通常、低い値は、より近い一致を示します。
- **part_ids** 一致した特定のチャンクまたはパーツを示す ID の配列。
- **pipeline_name** データを提供したパイプラインの名前。

7.3 高度なクエリー 中間ステップの結合

`ChunkText` や `ParseHtml` などの中間変換を含むパイプラインの場合、`part_ids` 列を使用して取得結果を中間パイプラインテーブルと結合することにより、特定の変換済みセグメントにアクセスできます。

7.3.1 構文の例

次のクエリは、取得結果を中間ステップテーブルと結合して、特定のチャンク値にアクセスします。

```
SELECT
  r.key,
  r.value,
  r.distance,
  r.part_ids,
  int_step.value AS chunked_content
FROM aidb.retrieve_text(my_kb, search query, 5) AS r
JOIN pipeline_my_pipeline_step_1 AS int_step
ON int_step.source_id = r.key
AND int_step.part_ids = (r.part_ids)[:1];
```

Examples

次の例は、AI パイプラインを使用して大量のナレッジベース KB 埋め込みを実行する方法を示しています。各ステージで予想される出力を含みます。これらの例は、標準の Postgres テーブルと外部 S3 バケットボリュームからのデータの取り込みをカバーしています。

例 1 Postgres テーブルからの大量埋め込み

このワークフローは、ソースデータがデータベース内に既に保存されている場合に使用されます。

ステップ 1 AI モデルを初期化する

テキストエンコード操作をサポートするモデルを登録します。

```
SELECT aidb.create_model(model__6545, bert_local);
__OUTPUT__
create_model
-----
model__6545
(1 row)
```

ステップ 2 ソースデータを準備する

テーブルを作成し、埋め込むレコードを挿入します。

```
CREATE TABLE source_table__6545 (
    id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    content TEXT NOT NULL
);

INSERT INTO source_table__6545 (id, content)
VALUES (43941, Catwalk Women Brown Heels),
       (55018, Lakme 3 in 1 Orchid Aqua Shine Lip Color),
       (19337, United Colors of Benetton Men Stripes Black Jacket);
```

ステップ 3 パイプラインを定義する

ソーステーブルをナレッジベースの宛先に接続するパイプラインを作成します。

```
SELECT aidb.create_pipeline(
    name => pipeline__6545,
    source => source_table__6545,
    source_key_column => id,
    source_data_column => content,
    step_1 => KnowledgeBase,
    step_1_options => aidb.knowledge_base_config(
        model => dummy,
        data_format => Text
    )
);
```

(次のページに続く)

```

    )
);
__OUTPUT__
INFO: Created pipeline "pipeline__6545"
INFO: Created knowledge base "public.pipeline_pipeline__6545" for pipeline "pipeline_
→_6545"

create_pipeline

-----

(pipeline__6545,Table,public,pipeline_pipeline__6545,source_id,value)
(1 row)

```

ステップ 4 実行と確認

処理エンジンをトリガーし、生成されたベクトルテーブルを検査します。

```

SELECT aidb.run_pipeline(pipeline__6545);

-- View the generated embeddings
SELECT * FROM pipeline_pipeline__6545 ORDER BY id;
__OUTPUT__
id | pipeline_id | source_id | part_ids | value
-----+-----+-----+-----+-----
1 | 49 | 43941 | {0} | [0.1,0.2,0.3]
2 | 49 | 55018 | {0} | [0.1,0.2,0.3]
3 | 49 | 19337 | {0} | [0.1,0.2,0.3]
(3 rows)

```

例 2 S3 バケット PGFS からの大量埋め込み

このワークフローは、外部クラウドストレージに保存されている非構造化データに使用されます。

ステップ 1 外部ストレージとボリュームを構成する

S3 バケットの場所を定義し、AIDB ボリュームにマッピングします。

```

SELECT pgfs.create_storage_location(
    bucket__6545,
    s3://noah-html-test,
    options => {"region": "eu-central-1", "skip_signature": "true"}
);

SELECT aidb.create_volume(volume__6545, bucket__6545, /, Text);

```

ステップ 2 ボリュームコンテンツをリストする

ボリュームは外部テーブルです—直接照会して、処理可能なファイルを確認します。

```
SELECT key, size, last_modified FROM volume__6545 ORDER BY key;
```

__OUTPUT__

key	size	last_modified
Bilge Ince.html	8878	2025-02-19 09:58:29+00
all_tags.html	11170	2025-02-19 09:58:29+00
hello_world.html	108	2025-02-19 09:58:30+00
simple.html	662	2025-02-19 09:58:29+00

(4 rows)

ステップ 3 ボリュームパイプラインを作成して実行する

```
SELECT aidb.create_pipeline(
  name => volume_pipeline__6545,
  source => volume__6545,
  step_1 => KnowledgeBase,
  step_1_options => aidb.knowledge_base_config(
    model => dummy,
    data_format => Text
  )
);
```

```
SELECT aidb.run_pipeline(volume_pipeline__6545);
```

ステップ 4 埋め込みを検証する

```
SELECT source_id, part_ids, value FROM pipeline_volume_pipeline__6545 ORDER BY source_
->id;
```

__OUTPUT__

source_id	part_ids	value
Bilge Ince.html	{0}	[0.1,0.2,0.3]
all_tags.html	{0}	[0.1,0.2,0.3]
hello_world.html	{0}	[0.1,0.2,0.3]
simple.html	{0}	[0.1,0.2,0.3]

(4 rows)

Hybrid search

ハイブリッド検索は、ナレッジベースの埋め込みのセマンティックベクター/埋め込み検索と 1 つ以上の追加フィルターメカニズムを組み合わせます。

- **リレーショナル述語フィルタリング** — ソーステーブルの任意の列、たとえば `WHERE category = 'vegetarian'` や `WHERE published_date > '2024-01-01'` など結果をフィルタリングします。
- **全文検索** — ネイティブ PostgreSQL FTS を使用してキーワードの関連性によって結果をランク付けし、特定の単語またはフレーズが結果に表示されるようにします。

これら 2 つのアプローチは、セマンティック検索と並行して、独立してまたは一緒に使用できます。

AI パイプラインは、埋め込み生成を含むナレッジベースの構築を通常のように処理します。ハイブリッド検索は、クエリー時に、ベクターテーブルとソーステーブルを組み合わせたカスタム SQL を構成することにより適用されます。これは別途インストールされる機能ではなく、以下の前提条件を超えて追加のセットアップは必要ありません。

前提条件

pgvector 反復インデックススキャンは、すべてのハイブリッドクエリパターンに必要です。ネイティブ FTS は、クエリでキーワードレグを使用する場合にのみ必要です。

pgvector 反復インデックススキャン

pgvector 0.8.0 以降が必要です。ハイブリッドクエリを実行する前に反復インデックススキャンを有効にします。これにより、ベクトルインデックスは、フィルタリング後でも LIMIT が満足するまで候補を生成し続けます。SET コマンドについては、[Writing Hybrid search queries](#) を参照してください。

ネイティブ PostgreSQL FTS 拡張機能は必要ありません。

標準の PostgreSQL 全文検索 FTS `tsvector` / `ts_rank` は、追加の拡張機能なしですべての Postgres インストールで利用できます。設定手順については、[Setting up native PostgreSQL FTS](#) を参照してください。

ボリュームソースの制限

リレーショナル述語フィルタリングまたは全文検索を使用したハイブリッド検索は、ボリュームソースのナレッジベースではサポートされていません。ボリュームソースには、結合するバッキング Postgres テーブルがないため、フィルタまたはインデックスを作成するソーステーブルがありません。

ボリュームソースのナレッジベースは、`aidb.retrieve_text()` および `aidb.retrieve_key()` を介して標準のセマンティック検索を引き続きサポートします。

Hybrid search

次のステップ

- [Setting up native PostgreSQL FTS](#) — `tsvector` 列、GIN インデックス、およびトリガーをソーステーブルに追加します。

- *Writing Hybrid search queries* —すべての組み合わせの SQL パターンリレーショナルフィルター、逆数ランクフュージョン RRF、重み付けリニアフュージョン。サンプルデータと予想される出力を含む作業例が含まれています。
- *Handling embedding service outages* —埋め込みサービスの停止を正常に処理する方法。
- *Pipelines API reference* — `aidb.kb_query_encode()`、`aidb.rerank_text()`、および `aidb.knowledge_bases` ビューの完全なリファレンス。

Handling embedding service outages

埋め込みサービスが利用できない場合、または最近の行の埋め込みが生成されていない場合、ハイブリッド検索クエリは失敗するか、不完全な結果を返します。3 層フォールバックを実装して、これらの状況でもアプリケーションが結果を返し続けるようにします。

フォールバックチェーンの定義

推奨される劣化チェーン

1. **ハイブリッド** —ベクター+全文検索 FTS プライマリパス、埋め込みサービスが必要です。
2. **FTS-only** — `tsvector` を使用したキーワード検索埋め込みサービスは必要ありません。
3. **Regex** — `ILIKE` パターンマッチ最後の手段、インデックスは必要ありません。

フォールバックの実装

埋め込みステップから接続エラーをキャッチし、適切なフォールバックにルーティングすることにより、アプリケーションコードに、または CASE 式またはラッパーファンクションを使用して SQL にこのストラテジーを実装します。以下の 3 つのクエリは、各層のビルディングブロックです。

プライマリ: ハイブリッドベクター+ネイティブ FTS

```
WITH semantic AS (
  SELECT source_id,
         (2.0 - (value <=> aidb.kb_query_encode(public.pipeline_docs, :query)::vector)) / 2.0 AS vec_score
  FROM public.pipeline_docs
  ORDER BY value <=> aidb.kb_query_encode(public.pipeline_docs, :query)::vector
  LIMIT 50
),
keyword AS (
  SELECT id::TEXT AS source_id,
         ts_rank_cd(search_vector, plainto_tsquery(english, :query)) AS fts_score
  FROM my_docs
  WHERE search_vector @@ plainto_tsquery(english, :query)
  LIMIT 50
)
SELECT COALESCE(s.source_id, k.source_id) AS id,
       0.6 * COALESCE(s.vec_score, 0) + 0.4 * COALESCE(k.fts_score, 0) AS score
```

(次のページに続く)

(前のページからの続き)

```
FROM semantic s
FULL OUTER JOIN keyword k ON s.source_id = k.source_id
ORDER BY score DESC
LIMIT 10;
```

フォールバック 1FTS のみ埋め込みサービスなし

```
SELECT id, title,
       ts_rank_cd(search_vector, plainto_tsquery(english, :query)) AS score
FROM my_docs
WHERE search_vector @@ plainto_tsquery(english, :query)
ORDER BY score DESC
LIMIT 10;
```

フォールバック 2: 正規表現インデックスなし

```
SELECT id, title
FROM my_docs
WHERE title ILIKE % || :query || %
      OR body ILIKE % || :query || %
LIMIT 10;
```

正規表現フォールバックは、完全なシークエンシャルスキャンを実行し、大規模なテーブルにスケーリングしません。緊急使用または小規模なデータセット用に予約し、アプリケーションにタイムアウトガードを追加して、長時間実行されるクエリを防止します。

!!!注 *Setting up native PostgreSQL FTS* で説明されている `tsvector` トリガーパターンにより、エンベディングが生成される前に到着した行であっても、FTS カバレッジが常に最新であることが保証されます。これにより、新しいコンテンツの信頼できるフォールバックになります。

Writing Hybrid search queries

以下のパターンは、ベクトル検索とキーワードおよび関係フィルターとのサポートされているすべての組み合わせをカバーしています。ネイティブ全文検索 FTS を必要とするパターンを使用する前に、関連する設定ページを完了します。

- *Setting up native PostgreSQL FTS*

ベクトル+関係フィルター

ソーステーブルの任意の列でセマンティック検索結果をフィルタリングします。フィルタはベクトル `ORDER BY` と同じクエリレベルで適用されるため、反復インデックススキャンで正確に `LIMIT` の結果を生成できます。

```
SET hnsw.iterative_scan = relaxed_order;

SELECT s.id, s.description,
```

(次のページに続く)

(前のページからの続き)

```

        kb.value <=> aidb.kb_query_encode(public.my_kb_vector, summer fashion)::vector
→AS distance
FROM my_kb_vector kb
JOIN my_source_table s ON s.id::TEXT = kb.source_id
WHERE s.category_id = 2 -- relational filter on source table
ORDER BY kb.value <=> aidb.kb_query_encode(public.my_kb_vector, summer
→fashion)::vector
LIMIT 10;

```

Vector + ネイティブ PostgreSQL FTS

標準の PostgreSQL `tsvector` / `ts_rank` をキーワードレグとして使用します。この例では、ソーステーブルに `search_vector tsvector` 列を想定しており、*Setting up native PostgreSQL FTS* の説明に従って移入およびインデックスが作成されます。

```

SET hnsw.iterative_scan = relaxed_order;

SELECT s.id, s.title,
       kb.value <=> aidb.kb_query_encode(public.pipeline_docs, connection
→pooling)::vector AS distance,
       ts_rank_cd(s.search_vector, plainto_tsquery(english, connection pooling))
       AS fts_score
FROM public.pipeline_docs AS kb
JOIN my_docs AS s ON s.id::TEXT = kb.source_id
WHERE s.search_vector @@ plainto_tsquery(english, connection pooling) -- keyword
→filter
ORDER BY kb.value <=> aidb.kb_query_encode(public.pipeline_docs, connection
→pooling)::vector
LIMIT 10;

```

WHERE 句は、ハードキーワードフィルタとして機能します。検索用語を含む行のみが検討されます。結果は意味的な類似性によって並べられます。両方だけでなくいずれかの信号と一致する結果を表示するには、代わりに `||| ||| |||` を使用します。

重み付けリニアフュージョン

加重線形フュージョンは、正規化されたベクトルの類似性とフルテキストスコアを、2つの明示的な重み 1 つがセマンティック関連性、もう 1 つがキーワード関連性を使用して単一の複合スコアに組み合わせます。合計が 1.0 になる必要があります。ランク位置のみを使用する RRF とは異なり、重み付けリニアフュージョンは実際のスコアの大きさを尊重するため、各信号が最終的なランキングにどの程度寄与するかを正確に制御したい場合に役立ちます。

重み付けリニアフュージョンと RRF を使用する場合

	Weighted linear fusion	RRF
Uses actual score values	Yes	No (rank positions only)
Requires score normalization	Yes	No
Tunable per use case	Yes (explicit weights)	Partially (smoothing constant k)
Robust to score scale differences	No (normalize carefully)	Yes (rank-based, scale-independent)
Best for	Workloads where score magnitude carries information	General-purpose merging of heterogeneous result sets

ワークロードに基づいて重みを調整します。

Use case	Suggested weights
Concept/meaning is most important	0.7 semantic, 0.3 keyword
Exact terminology matters as much as meaning	0.5 semantic, 0.5 keyword
Keyword precision is critical (for example, code or rule search)	0.3 semantic, 0.7 keyword

ネイティブ PostgreSQL FTS の場合

ベクトル距離と `ts_rank` 値は異なるスケールを使用するため、両方のレッグは結合する前に `[0, 1]` に正規化されます。コサイン距離レンジ `[0, 2]` は、 $(2.0 - \text{distance}) / 2.0$ を使用して類似度に変換されます。`ts_rank_cd` は既におよそ `[0, 1]` の値を返しています。

```
WITH semantic AS (
    SELECT source_id,
           (2.0 - (value <=> aidb.kb_query_encode(public.pipeline_docs, connection_
→pooling)::vector)) / 2.0
           AS vec_score
    FROM public.pipeline_docs
    ORDER BY value <=> aidb.kb_query_encode(public.pipeline_docs, connection_
→pooling)::vector
    LIMIT 50
),
keyword AS (
    SELECT id::TEXT AS source_id,
           ts_rank_cd(search_vector, plainto_tsquery(english, connection pooling)) AS_
→fts_score
    FROM my_docs
    WHERE search_vector @@ plainto_tsquery(english, connection pooling)
    ORDER BY fts_score DESC
```

(次のページに続く)

(前のページからの続き)

```

LIMIT 50
)
SELECT
    COALESCE(s.source_id, k.source_id) AS id,
    0.6 * COALESCE(s.vec_score, 0) + 0.4 * COALESCE(k.fts_score, 0) AS fusion_score
FROM semantic s
FULL OUTER JOIN keyword k ON s.source_id = k.source_id
ORDER BY fusion_score DESC
LIMIT 10;

```

実用的な例リレーショナル述語フィルタリング

この例は、セマンティックベクトル検索とリレーショナル WHERE フィルターを組み合わせ、結果を特定のカテゴリに絞り込む方法を示しています。パイプラインは製品名をナレッジベースに埋め込み、クエリによってカテゴリでフィルターされた意味的に同様の結果を取得します。

ソーステーブルとパイプラインの設定

1. ソーステーブルを作成します。

```

CREATE TABLE products (
    id SERIAL PRIMARY KEY,
    product_name TEXT NOT NULL,
    category TEXT NOT NULL
);

```

1. サンプルデータを入力します。

```

INSERT INTO products (product_name, category) VALUES
    (Hamburger, meat),
    (Cheeseburger, meat),
    (Fish n Chips, fish),
    (Fries, vegetarian),
    (Burrito, meat),
    (Pizza, meat),
    (Sandwich, meat),
    (Veggie Burger, vegetarian),
    (Kebab, meat);

```

1. product_name をナレッジベースに埋め込むパイプラインを作成します。

```

SELECT aidb.create_pipeline(
    name => food_products,
    source => products,
    source_key_column => id,

```

(次のページに続く)

(前のページからの続き)

```

source_data_column => product_name,
step_1             => KnowledgeBase,
step_1_options     => aidb.knowledge_base_config(
                    model      => bert,
                    data_format => Text
                    )
);

```

1. パイプラインを実行してエンベディングを生成します。

```
SELECT aidb.run_pipeline(food_products);
```

標準のセマンティッククエリーフィルタなし

このベースラインクエリーは、制約なしで上位のセマンティック一致を返すため、結果セットを後続のフィルタリングされたクエリーと比較できます。

```
SELECT * FROM aidb.retrieve_text(public.pipeline_food_products, fast food, topk => 4);
```

__OUTPUT__

key	value	distance	part_ids	pipeline_name	intermediate_steps
8	Veggie Burger	0.9955745765192575	{0}	food_products	[]
6	Pizza	1.0280680840689795	{0}	food_products	[]
4	Fries	1.0630385212203115	{0}	food_products	[]
7	Sandwich	1.0903594734917037	{0}	food_products	[]

リレーショナルフィルターを使用したカスタムクエリー

このクエリーは、ソーステーブルに WHERE 句を追加して、結果を単一のカテゴリに制限し、ベクトル ORDER BY を同じクエリーレベルに保つため、反復インデックススキャンが LIMIT を効率的に満たすことができます。

```

SET hnsf.iterative_scan = relaxed_order;

SELECT vector.source_id, source.product_name, source.category
FROM public.pipeline_food_products AS vector
JOIN products AS source ON source.id::TEXT = vector.source_id
WHERE source.category = vegetarian
      -- relational filter
ORDER BY vector.value <=> aidb.kb_query_encode(public.pipeline_food_products, fast_
->food)::vector
LIMIT 10;
__OUTPUT__
source_id | product_name | category
- - - - -

```

(次のページに続く)

(前のページからの続き)

8	Veggie Burger	vegetarian
4	Fries	vegetarian

ベジタリアン商品のみが返され、「ファーストフード」との意味的な関連性によって順序付けられます。

Hybrid search reference

AIDB は、ハイブリッド検索クエリを構成するためのヘルパー関数とメタデータビューを提供します。これらを使用して、モデル名やテーブルパスをハードコーディングせずに、特定のナレッジベースに対するクエリをエンコードし、取得した結果を再ランク付けし、ナレッジベース構成を検査します。

quietb.kb_query_encode()

モデル名を検索する必要なく、特定のナレッジベース用に構成された埋め込みモデルを使用してテキストクエリをエンコードします。

```
SELECT aidb.kb_query_encode(public.my_kb_vector, test query);
```

カスタムベクトルクエリを構築するときに、`aidb.encode_text_query('<model>', '<query>')` の代わりにこれを使用します。

quietb.rerank_text()

候補を取得した後、最終的な関連性の高い結果を得るために再ランキングモデルに渡します。

```
SELECT aidb.rerank_text(my_reranking_model, query text, ARRAY[result1, result2, ...]);
```

`nim_reranking` や `hf_tei_reranking` などの再ランキングモデルは、取得後の別の後処理ステップとしてサポートされています。

aidb.knowledge_bases view

`aidb.knowledge_bases` ビュー `aidb.kbs` としてもアクセス可能は、ハイブリッドクエリの構成に必要なすべてのメタデータを公開します。

Column	Type	Description
<i>id</i>	integer	内部識別子
<i>name</i>	text	ナレッジベース名
<i>vector_schema</i>	text	エンベディングテーブルのスキーマ
<i>vector_table</i>	text	エンベディングストレージテーブル
<i>model_name</i>	text	この KB で使用される埋め込みモデル
<i>distance_operator</i>	aidb.distanceoperator	距離メトリックの列挙値
<i>distance_operator_sql</i>	text	SQL 演算子構文の距離演算子
<i>vector_data_column</i>	text	エンベディングを格納する列
<i>vector_key_column</i>	text	埋め込みをソース行にリンクする列
<i>vector_index</i>	jsonb	ベクトルインデックス構成

次のページに続く

表 4 – 前のページからの続き

Column	Type	Description
<i>pipeline_ids</i>	integer[]	このナレッジベースに接続されているすべてのパイプラインの ID
<i>pipeline_names</i>	text[]	このナレッジベースに接続されているすべてのパイプラインの名前

```
SELECT name, vector_schema, vector_table, distance_operator_sql
FROM aidb.knowledge_bases;
```

Setting up native PostgreSQL FTS

tsvector 列、GIN インデックス、および自動更新トリガーでソーステーブルを構成して、ハイブリッド検索クエリのキーワードレグとしてネイティブの PostgreSQL 全文検索 FTS を使用します。追加の拡張機能は必要ありません。以下のすべての手順は、データベースで実行される SQL コマンドです。セットアップには、タイトルの一致が本文の一致よりもランクが高いようにフィールドの重み付けと、行が挿入または更新されても検索列が最新の状態に保たれるようにトリガーが含まれます。

1. tsvector 列を追加します

tsvector 列を追加して、各行の前処理されたテキスト表現を保存します。これは、FTS インデックスが動作する列です。

```
ALTER TABLE my_docs ADD COLUMN search_vector tsvector;
```

1. 列に重み付けフィールドを設定します

重み付けフィールドを使用して列にデータを入力して、さまざまなフィールドが関連性に影響を与える方法を制御します。重み A 最高～D 最低を使用する。タイトルや概要などのフィールドに高い重みを割り当て、長い本文フィールドに低い重みを割り当てます。

```
UPDATE my_docs
SET search_vector =
    setweight(to_tsvector(english, coalesce(title, )), A) ||
    setweight(to_tsvector(english, coalesce(summary, )), B) ||
    setweight(to_tsvector(english, coalesce(body, )), C);
```

1. GIN インデックスを作成する

search_vector 列に GIN 一般逆引きインデックスを作成して、高速な全文検索を有効にします。このインデックスがないと、FTS クエリはシークエンシャルスキャンを実行し、スケールしません。

```
CREATE INDEX my_docs_fts_idx ON my_docs USING gin(search_vector);
```

1. 列を最新の状態に保つトリガーを作成します

新しい行および更新された行を自動的にトークン化するトリガーを設定して、手動更新せずに FTS インデックスの同期を維持します。

```

CREATE OR REPLACE FUNCTION my_docs_search_vector_update()
RETURNS trigger AS $$
BEGIN
    NEW.search_vector :=
        setweight(to_tsvector(english, coalesce(NEW.title, )), A) ||
        setweight(to_tsvector(english, coalesce(NEW.summary, )), B) ||
        setweight(to_tsvector(english, coalesce(NEW.body, )), C);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER my_docs_search_vector_trigger
BEFORE INSERT OR UPDATE ON my_docs
FOR EACH ROW EXECUTE FUNCTION my_docs_search_vector_update();

```

1. テストクエリーを実行する

プレーンフレーズマッチングには `plainto_tsquery` を使用し、Google スタイルの構文 `"exact phrase"`、`-exclude`、OR の場合は `websearch_to_tsquery` を使用します。アプリケーションのクエリ入力形式と一致するものを使用します。

```

SELECT id, title,
       ts_rank_cd(search_vector, query) AS score
FROM my_docs,
     plainto_tsquery(english, connection pooling) query
WHERE search_vector @@ query
ORDER BY score DESC
LIMIT 10;

```

ネイティブ FTS インデックスとともに実行される AIDB パイプラインの場合、AIDB はエンベディング生成とベクター検索を処理します。FTS トリガーとパイプラインは独立して動作し、新しい行は両方によって自動的にカバーされます。

!!!注上記のトリガーパターンにより、埋め込みが生成される前に到着する行であっても、FTS カバレッジが常に最新であることが保証されます。これにより、ネイティブ FTS は、新しいコンテンツの信頼できるフォールバックになります。詳細については、[Handling embedding service outages](#) を参照してください。

次のステップ

- [Writing Hybrid search queries](#) — 重み付け線形フュージョンを含むハイブリッドクエリで FTS インデックスを使用します。
- [Handling embedding service outages](#) — 埋め込みサービスが利用できない場合に、フォールバックとしてネイティブ FTS を使用します。

Multi-pipeline knowledge bases

ナレッジベース KB には、複数のパイプラインを書き込むことができます。各パイプラインは、独自のソース、処理チェーン、および自動処理モードを維持しますが、それらはすべて同じベクトルテーブルにエンベディングを保存します。取得は KB を 1 回照会し、接続されているすべてのパイプラインにわたって結果を集計します。

これは、次のような場合に役立ちます。

- 複数のソーステーブルを単一の検索可能なインデックスに結合します。
- 混合データ形式 — たとえば、1 つのパイプラインは **Text** 行をインデックスし、別のパイプラインは *Configuring external storage* を介して Pdf ファイルをインデックスします。
- ソースごとに異なる前処理を適用します — 1 つのパイプラインは埋め込む前に **ChunkText** を実行し、別のパイプラインは短いテキストを直接埋め込みます。

既存の KB へのパイプラインのアタッチ

最初のパイプラインは KB を作成します。後続のパイプラインは、同じ **destination** に書き込むことにより接続します。後続のパイプラインに *aidb.knowledge_base_config_from_kb()* を使用して、KB のモデル、距離演算子、ベクトルインデックスを継承します。

```
- - Pipeline 1  creates the KB from the North America customer feedback table
SELECT aidb.create_pipeline(
  name => feedback_na_pipe,
  source => customers_na,
  source_key_column => id,
  source_data_column => feedback,
  destination => customer_feedback_kb,
  step_1 => KnowledgeBase,
  step_1_options => aidb.knowledge_base_config(model => bert, data_format => Text)
);

- - Pipeline 2  attaches the Europe customer feedback table to the same KB
SELECT aidb.create_pipeline(
  name => feedback_eu_pipe,
  source => customers_eu,
  source_key_column => id,
  source_data_column => feedback,
  destination => customer_feedback_kb,
  step_1 => KnowledgeBase,
  step_1_options => aidb.knowledge_base_config_from_kb(data_format => Text)
);

SELECT aidb.run_pipeline(feedback_na_pipe);
SELECT aidb.run_pipeline(feedback_eu_pipe);
```

クエリーは両方のパイプラインから行を返し、**pipeline_name** がソースを識別します。

```

SELECT key, value, pipeline_name
FROM aidb.retrieve_text(public.customer_feedback_kb, shipping was slow, topk => 3);
__OUTPUT__
key | value | pipeline_name
-----+-----+-----
1042 | Delivery was held up in customs for over a week | feedback_eu_pipe
318 | Order took twice as long to arrive as estimated | feedback_na_pipe
1057 | Tracking number never updated until it arrived | feedback_eu_pipe

```

さまざまなステップチェーンが1つのKBを共有できます。たとえば、あるパイプラインはKnowledgeBase ステップの前に `ChunkText` で長いドキュメントをチャンクし、別のパイプラインは短い行を直接埋め込みます。`aidb.retrieve_text()` の `intermediate_steps` JSONB 列は、パイプラインごとの中間結果を結果ます。

互換性ルール

KB に接続されているすべてのパイプラインは、埋め込み形状について一致する必要があります。既存のKBをターゲットとするパイプラインを作成すると、AIDB は以下を検証します。

Constraint	Error if violated
Model	<i>Model mismatch: existing KB uses '<X>' but pipeline specifies '<Y>'</i>
Distance operator	<i>Distance operator mismatch: ...</i>
Vector index config	<i>Vector index mismatch: ...</i>
Embedding dimensions	<i>Embedding dimension mismatch: existing KB vector column has N dimensions but model '<X>' produces M</i>

`aidb.knowledge_base_config_from_kb()` は、既存のKBからモデル、距離演算子、およびベクトルインデックスを継承するため、最初の3つのチェックに常にパスします。寸法チェックは引き続き実行され、モデルが同じ名前で異なる出力サイズで再作成されるエッジケースをキャッチします。

削除セマンティクス

Action	Effect
<code>aidb.delete_pipeline('one_of_many')</code>	Removes that pipeline and only its rows from the shared vector table. KB and other pipelines remain.
<code>aidb.delete_pipeline('the_last_one')</code>	Drops the KB registry entry and the vector table along with the pipeline.
<code>aidb.delete_knowledge_base('schema.table')</code>	Cascades — deletes every attached pipeline, the KB, and the vector table.

これらの操作は、ソーステーブルまたはソースボリュームに触れることはありません。

接続されたパイプラインの検査

```

SELECT name, model_name, distance_operator, pipeline_names
FROM aidb.knowledge_bases
WHERE name = public.customer_feedback_kb;
__OUTPUT__

```

name	model_name	distance_operator	pipeline_names
public.customer_feedback_kb	bert	L2	{feedback_eu_pipe, feedback_na_pipe}

集計数とステータスについては、aidb.knowledge_base_metrics を照会します。パイプラインごとの進行状況については、aidb.pipeline_metrics を照会してください。

ベクターテーブルスキーマ

共有ベクトルテーブルには pipeline_id 列が含まれるため、各行を元のパイプラインまでトレースできます。主キーは (pipeline_id, source_id, part_ids) です。

```

SELECT pipeline_id, source_id, part_ids
FROM customer_feedback_kb
ORDER BY pipeline_id, source_id;
__OUTPUT__

```

pipeline_id	source_id	part_ids
1	318	{0}
1	412	{0}
2	1042	{0}
2	1057	{0}

第8章 AIDB overview

AIDB は、EDB によって維持される Postgres 拡張機能であり、AI データワークフローを Postgres データベースに直接もたします。これは、AI アプリケーションのデータを準備、埋め込み、インデックス付けするパイプラインを構築する宣言的 SQL ネイティブの方法を提供します。取得拡張生成 RAG、セマンティック検索、大規模言語モデル LLM 統合など。

AIDB は Postgres 内で実行されるため、データがデータベースを離れる必要はありません。SQL で AI ワークフローを定義すると、AIDB がエンベディング生成、ベクトルインデックス、モデル統合、リアルタイム同期の複雑さを処理します。

8.1 AIDB の役割

AIDB には、**パイプライン** と **ナレッジベース** という 2 つの主要なビルディングブロックがあります。

**** Creating pipelines **** 生データを AI 対応ベクターに変換するマルチステップワークフローを定義します。パイプラインの各ステップは、1 つの変換-PDF の解析、テキストのチャンク、OCR の実行、コンテンツの要約、またはベクター埋め込みの生成を処理します。パイプラインは、オンデマンドでバッチで、またはソースデータが変更されるたびに自動的に実行できます。

**** Knowledge bases reference **** は、埋め込みパイプラインステップの出力です。これらは pgvector インデックステーブルにベクトル埋め込みを保存し、`aidb.retrieve_key()` および `aidb.retrieve_text()` ファンクションを介してセマンティック検索を公開します。複数のパイプラインは同じナレッジベースにフィードでき、単一のナレッジベースで複数の取得クエリを提供できます。

すべてのパイプライン操作は、スタンドアロンの **SQL ファンクション** - `aidb.chunk_text()`、`aidb.parse_pdf()`、`aidb.perform_ocr()`、`aidb.summarize_text()` など - として使用できるため、完全なパイプラインを定義せずにクエリで直接呼び出すことができます。

8.2 スタンドアロンの拡張機能としての AIDB

AIDB は Postgres 拡張機能です。完全にデータベース内にあり、Agent Factory または Hybrid Manager への実行時依存はありません。他の Postgres 拡張機能と同じようにインストールし、`postgresql.conf` を介して構成し、SQL を介して対話します。

!!!note Agent Factory を使用している場合、AIDB はそのプラットフォームの一部として既に含まれていますが、Agent Factory は AIDB を使用する必要はありません。このドキュメントでは、スタンドアロン AIDB について説明します。

8.3 主要な機能

モデルの柔軟性 AIDB は、ローカルで実行されているモデル BERT や T5 などに接続するだけでなく、OpenAI 互換 API を介して外部モデルに接続します。サポートされているプロバイダーには、NVIDIA NIM、Google Gemini、および OpenAI 互換のエンドポイントが含まれます。モデルの資格情報は、標準の Postgres ユーザーマッピング機能を介して安全に管理されます。

データソースのサポート—パイプラインは、標準の Postgres テーブルまたはオブジェクトストアとローカルファイルシステムをサポートする Postgres File System (PGFS) 拡張機能を介して外部ストレージから読み取ることができます。パイプライン出力は、Postgres テーブルに書き込むか、PGFS を使用してボリュームに戻すことができます。

自動処理—Postgres トリガーベースのライブモードまたはバックグラウンドワーカーベースのスケジューリングを使用して、自動的に実行するようにパイプラインを構成できます。これにより、手動介入なしでソースデータが変更されると、ナレッジベースが最新の状態に保たれます。

スキーマの柔軟性 すべての AIDB オブジェクトパイプライン、ナレッジベース、ボリュームは任意の Postgres スキーマで作成でき、既存のデータベース構造に沿って AI ワークロードを簡単に編成できます。

ベクトルインデックス制御—ナレッジベースは、pgvector を介して HNSW や IVFFlat などの構成可能なベクトルインデックスをサポートしています。

8.3.1 Known issues

次の既知の問題と制限が AIDB の現在のリリースに適用されます。

`aidb.retry_pipeline_errors()` はまだ利用できません

影響します すべてのバージョン

エラーログから失敗したパイプラインレコードを再試行する機能 `aidb.retry_pipeline_errors()` は、まだ使用できません。将来のリリースの予定です。

回避策 `aidb.get_error_logs()` を使用して、失敗したレコードを検査し、障害の原因を特定します。根本的な問題を解決したら、破損したソースファイルの置換やモデル構成の修正など、`aidb.clear_error_logs()` を使用してエラーエントリを削除し、`aidb.run_pipeline()` でパイプラインを再実行して影響を受けるデータを再処理します。

第9章 Reference

AIDB 拡張機能の完全な API リファレンス: パラメーターテーブル、戻り値の型、およびルックアップのシグネチャ。使用コンテキストと実際の例については、以下の表にリンクされているガイドセクションを参照してください。

Page	What it covers	Related guide
Models reference	Catalog views, model CRUD (<i>create_model</i> , <i>get_model</i> , <i>delete_model</i>), HCP model functions, inference functions (<i>encode_text</i> , <i>decode_text</i> , <i>rerank_text</i>), and all model config helpers	Integrating models
Knowledge bases reference	Knowledge base views and stats, retrieval functions (<i>retrieve_key</i> , <i>retrieve_text</i>), volume functions, and auto-processing	Knowledge bases reference
Operations reference	Pipeline error log functions (<i>get_error_logs</i> , <i>clear_error_logs</i> , <i>retry_pipeline_errors</i>)	Pipeline error log
Functions reference	Standalone SQL functions: <i>chunk_text</i> , <i>parse_html</i> , <i>parse_pdf</i> , <i>perform_ocr</i> , <i>summarize_text</i>	SQL functions
Pipelines reference	Pipeline types, views, CRUD functions, step config helpers, and vector index config helpers	Reference → Functions

9.1 Functions reference

パイプラインを必要とせず、クエリでデータを直接変換する AIDB スタンドアロン SQL ファンクションのリファレンス。ガイドスタイルのドキュメントと使用例については、[SQL functions](#) を参照してください。

AI 推論機能 `encode_text`、`decode_text`、`rerank_text`、および関連に関しては、[\[?\] \[?\] \[?\]](#) を参照してください。

9.1.1 `helpb.chunk_text`

テキスト文字列を、より小さい意味的に一貫したセグメントに分割します。

パラメーター

Parameter	Type	Description
<i>input</i>	TEXT	チャンクへのテキスト。
<i>options</i>	JSONB	チャンク構成 以下を参照してください。

オプション

Key	Type	Default	Description
<i>desired_length</i>	integer	Required	ターゲットセグメントサイズ。 `max_length`が省略された場合、厳密な上限として機能します。
<i>max_length</i>	integer	NULL	チャンクサイズの上限。チャンクは、セマンティック境界を保持するためのみに`desired_length`を超えて拡張されます。
<i>overlap_length</i>	integer	0	境界を越えたコンテキストを保持するために、連続するチャンク間で繰り返すコンテンツの量。
<i>strategy</i>	text	'chars'	チャンク単位 'chars' 文字ベースまたは'words' ワードベース。`desired_length`、`max_length`、および`overlap_length`の単位を決定します。

返品

Column	Type	Description
<i>part_id</i>	integer	ゼロベースのセグメントインデックス。
<i>chunk</i>	text	テキストセグメント。

例

```
SELECT * FROM aidb.chunk_text(
  input    => Long text here...,
  options => {"desired_length": 120, "max_length": 150}
);
```

9.1.2 quietb.parse_html

HTML 文字列から可読テキストを抽出し、構造を維持しながらタグを削除します。

パラメーター

Parameter	Type	Description
<i>html</i>	TEXT	解析する HTML 文字列。
<i>options</i>	JSONB	構成の解析 以下を参照してください。

オプション

Key	Type	Default	Description
<i>method</i>	text	'StructuredPlain-text'	解析方法 'StructuredPlain-text' プレーンテキスト抽出または 'StructuredMarkdown' ヘッダーとリストを保持するマークダウンのような出力。

返品

TEXT — 抽出されたテキストコンテンツ。

例

```
SELECT aidb.parse_html(
  html    => <h1>Hello</h1><p>World</p>,
  options => {"method": "StructuredPlaintext"}
);
```

9.1.3 quietb.parse_pdf

バイナリ PDF データからテキストを抽出します。ページごとに 1 行を返します。

パラメーター

Parameter	Type	Description
<i>bytes</i>	BYTEA	生の PDF バイナリデータ。
<i>options</i>	JSONB	構成の解析 以下を参照してください。

オプション

Key	Type	Default	Description
<i>method</i>	text	'Structured'	解析方法。現在 ``Structured`` 仕様ベースのテキストブロック抽出。
<i>allow_partial_parsing</i>	boolean	true	``true`` の場合、個々のページでエラーが発生したときに解析を継続し、可能な限り多くのデータを結果ます。

返品

Column	Type	Description
<i>part_id</i>	integer	テキストが抽出されたページインデックスゼロベース。
<i>text</i>	text	そのページの抽出されたテキスト。

例

```
SELECT * FROM aidb.parse_pdf(
  bytes => pg_read_binary_file(/path/to/doc.pdf)::BYTEA,
  options => {"allow_partial_parsing": true}
);
```

9.1.4 quietb.perform_ocr

登録された OCR 対応モデルを使用して、画像データからテキストを抽出します。

パラメーター

Parameter	Type	Description
<i>input</i>	BYTEA	生のバイナリ画像データ。
<i>options</i>	JSONB	OCR 構成下記を参照してください。

オプション

Key	Type	Description
<i>model</i>	text	登録された OCR 対応モデルの名前たとえば、 <code>`nim_ocr`</code> プロバイダーを使用するモデル。

返品

Column	Type	Description
<i>part_id</i>	integer	テキストブロックインデックス。プロバイダーが複数のテキストセグメントを返す場合、単一の画像から複数の行が生成される場合があります。
<i>text</i>	text	そのブロックの抽出されたテキスト。

例

```
SELECT * FROM aidb.perform_ocr(
  input    => pg_read_binary_file(/path/to/image.png)::BYTEA,
  options => {"model": "my_ocr_model"}
);
```

9.1.5 quietb.summarize_text

登録された言語モデルを使用して、テキスト文字列の簡潔な概要を生成します。

パラメーター

Parameter	Type	Description
<i>input</i>	TEXT	要約するテキスト。
<i>options</i>	JSONB	要約構成以下を参照してください。

オプション

Key	Type	Default	Description
<i>model</i>	text	Required	<code>`decode_text`</code> をサポートする登録済みモデルの名前。
<i>prompt</i>	text	(standard prompt)	概要スタイルをガイドするカスタム指示たとえば <code>'Summarize for a 5th grader'</code> 。

次のページに続く

表 14 – 前のページからの続き

Key	Type	Default	Description
<i>chunk_config</i>	JSONB	NULL	入力がモデルのコンテキストウィンドウを超える場合、要約の前に適用するチャンク構成。 <code>`aidb.chunk_text`</code> オプションと同じキーを受け入れます。
<i>strategy</i>	text	<i>'append'</i>	要約戦略 <i>'append'</i> 各チャンクを独立して要約および連結する または <i>'reduce'</i> 目的の長さに達するまで反復要約する。
<i>reduction_factor</i>	integer	3	<code>`reduce`</code> ストラテジーで使用されます。各反復でテキストをどの程度積極的に削減するかを制御します。

返品

TEXT — 生成された概要。

例

```
SELECT aidb.summarize_text(
  input    => Long article text here...,
  options => {"model": "my_t5_model"}
);
```

9.1.6 quietb.summarize_text_aggregate

`aidb.summarize_text` の集計バージョン。各グループのすべての行からテキストを蓄積し、結合された結果を要約のために LLM に送信します。GROUP BY グループごとに 1 つの概要を結果ます。空の行と NULL 行はスキップされます。

パラメーター

Parameter	Type	Required	Description
<i>input</i>	TEXT	Yes	各行のテキスト列。空/NULL 行はスキップされます。

次のページに続く

表 15 – 前のページからの続き

Parameter	Type	Required	Description
<i>options</i>	JSON	Yes	<code>`aidb.summarize_text_config()`</code> でビルドされた構成オブジェクト。少なくとも <code>`model`</code> を含める必要があります。

オプション

`aidb.summarize_text` と同じオプションを受け入れます上記を参照してください。 `options` は集計に 必須 であり、オプションではないことに注意してください。

返品

TEXT グループの概要。

使用法

```
SELECT category,
       aidb.summarize_text_aggregate(
           text_column,
           aidb.summarize_text_config(my_t5_model)::json ORDER BY id
       ) AS summary
FROM my_table
GROUP BY category;
```

9.1.7 パラメーターのデフォルトのクイックリファレンス

Function	Parameter	SQL default	Runtime default
<i>summarize_text</i>	<i>options</i>	<code>'{}'</code>	Must include <i>model</i>
<i>summa- rize_text_aggregate</i>	<i>input</i>	— (required)	—
<i>summa- rize_text_aggregate</i>	<i>options</i>	— (required)	—
<i>summarize_text_config</i>	<i>model</i>	— (required)	—
<i>summarize_text_config</i>	<i>chunk_config</i>	<i>NULL</i>	No chunking
<i>summarize_text_config</i>	<i>prompt</i>	<i>NULL</i>	Standard summarize prompt
<i>summarize_text_config</i>	<i>strategy</i>	<i>NULL</i>	<code>'append'</code>
<i>summarize_text_config</i>	<i>reduction_factor</i>	<i>NULL</i>	3
<i>summarize_text_config</i>	<i>inference_config</i>	<i>NULL</i>	Provider defaults
<i>chunk_text_config</i>	<i>desired_length</i>	— (required)	—

次のページに続く

表 16 – 前のページからの続き

Function	Parameter	SQL default	Runtime default
<i>chunk_text_config</i>	<i>max_length</i>	<i>NULL</i>	Same as <i>desired_length</i>
<i>chunk_text_config</i>	<i>overlap_length</i>	<i>NULL</i>	<i>0</i>
<i>chunk_text_config</i>	<i>strategy</i>	<i>NULL</i>	<i>'chars'</i>

9.2 Knowledge bases reference

ナレッジベース KB 機能とビューのリファレンス。ガイドスタイルのドキュメントについては、[Knowledge bases reference](#) を参照してください。

9.2.1 ビュー

`helpb.Knowledge_bases`

`aidb.kbs` としてもアクセスできます。データベース内のすべてのナレッジベースとその完全な構成をリストします。

Column	Type	Description
<i>id</i>	integer	内部識別子
<i>name</i>	text	ナレッジベース名
<i>vector_schema</i>	text	エンベディングテーブルのスキーマ
<i>vector_table</i>	text	エンベディングストレージテーブル
<i>model_name</i>	text	使用される埋め込みモデル
<i>distance_operator</i>	helpb.DistanceOperator	検索に使用されるベクトル距離ファンクション
<i>distance_operator_sql</i>	text	SQL 演算子構文の距離演算子たとえば <code><-></code> 、 <code><=></code>
<i>vector_data_column</i>	text	エンベディング列
<i>vector_key_column</i>	text	エンベディングテーブルのキー列
<i>vector_index</i>	jsonb	ベクトルインデックス構成
<i>pipeline_ids</i>	integer[]	このナレッジベースに接続されているすべてのパイプラインの ID
<i>pipeline_names</i>	text[]	このナレッジベースに接続されているすべてのパイプラインの名前

`helpb.Knowledge_base_metrics`

`aidb.kbm` としてもアクセスできます。各ナレッジベースの集計統計—パイプライン数、埋め込み行数、および最悪のパイプラインステータス。パイプラインごとの進行状況バックログ、エラー、ソース/宛先数については、`aidb.pipeline_metrics` を参照してください。

Column	Type	Description
<i>name</i>	text	ナレッジベース名 <i>schema.vector_table</i> 。
<i>pipelines</i>	integer	この KB に接続されているパイプラインの数。
<i>embeddings</i>	bigint	ベクタテーブルの行数。
<i>status</i>	text	アタッチされているすべてのパイプラインで最悪のステータス Reference → Functions を参照してください。

例

```
SELECT * FROM aidb.kbm;
```

__OUTPUT__

name	pipelines	embeddings	status
public.product_catalog_kb	1	1842	UpToDate
public.customer_feedback_kb	2	915	UpToDate

(2 rows)

古いビュー名 `aidb.knowledge_base_stats` および `aidb.kbstat` は、下位互換性を保つために `aidb.knowledge_base_metrics` のエイリアスとして保持されます。

9.2.2 タイプ

helpb.DistanceOperator

取得中に使用されるベクトル距離ファンクション。

Value	Description
<i>L2</i>	ユークリッド距離
<i>InnerProduct</i>	内積
<i>Cosine</i>	コサイン類似度
<i>L1</i>	L1 距離
<i>Hamming</i>	ハミング距離
<i>Jaccard</i>	Jaccard 距離

helpb.PipelineDataFormat

パイプラインソースまたはボリュームのデータのフォーマット。

Value	Description
<i>Text</i>	プレーンテキスト
<i>Image</i>	バイナリ画像データ
<i>Pdf</i>	PDF ドキュメント

helpb.PipelineAutoProcessing モード

ナレッジベースパイプラインの自動処理モード。

Value	Description
<i>Live</i>	データが変更されるたびに Postgres トリガーを介した即時処理
<i>Background</i>	Postgres バックグラウンドワーカーを介した定期処理
<i>Disabled</i>	自動処理なし。`aidb.run_pipeline()`で手動実行

9.2.3 ナレッジベース機能

パイプラインの自動処理モードを変更するには、`aidb.update_pipeline` を使用します。

promoteb.retrieve_key

ソースデータを取得せずに、上位一致するエンベディングのソースレコードキーと距離を結果ます。`part_ids` は、マルチステップパイプラインの各ステップの一致する部分の ID も結果ます。たとえば、単一のソースレコードが「ChunkText」ステップによって 10 の部分に分割されている場合、`part_ids` は、どのチャンクが一致したかを示します。

パラメーター

Parameter	Type	Default	Description
<i>knowledge_base_name</i>	TEXT	Required	ナレッジベースの名前 <code>schema.vector_table</code> 。
<i>query</i>	TEXT or BYTEA	Required	テキストまたは画像のバイトを照会します。エンベディングモデルのデータ形式と一致する必要があります。
<i>topk</i>	INTEGER	0	返される結果の数。`0`はデフォルトの 1 を使用します。

次のページに続く

表 22 – 前のページからの続き

Parameter	Type	Default	Description
<i>deduplicate</i>	BOOLEAN	<i>true</i>	複数の「部分」チャンク/ページなどが一致する場合でも、各ソースの結果を 1 回だけ返します

返品

Column	Type	Description
<i>key</i>	text	ソースレコードキー
<i>distance</i>	double precision	クエリーからのベクトル距離
<i>part_ids</i>	bigint[]	ソースレコードのどの部分たとえばチャンクまたはページが一致するベクトルを生成した
<i>pipeline_name</i>	text	この行を提供したパイプラインの名前。KB に複数のパイプラインが接続されている場合に役立ちます。

例

```
SELECT * FROM aidb.retrieve_key(public.customer_feedback_kb, late delivery, topk=>3);
```

__OUTPUT__

key	distance	part_ids	pipeline_name
1042	1.1931772046185758	{0,0}	feedback_eu_pipe
318	1.1980633963685476	{2,0}	feedback_na_pipe
1057	1.2150919866080878	{2,0}	feedback_eu_pipe

(3 rows)

helpb.retrieve_text

エンベディングテーブルをソーステーブルに結合することにより、上位に一致するエンベディングのソーステキストと距離を結果ます。part_ids では、マルチステップパイプラインの各ステップに一致するパーツの ID も結果ます。たとえば、単一のソースレコードが「ChunkText」ステップによって 10 の部分に分割されている場合、part_ids は、どのチャンクが一致したかを示します。戻り列 intermediate_steps には、実際の中間ステップ出力が含まれます。

パラメーター

Parameter	Type	Default	Description
<i>knowledge_base_name</i>	TEXT	Required	ナレッジベースの名前 <i>schema.vector_table</i> 。
<i>query</i>	TEXT or BYTEA	Required	テキストまたは画像のバイトを照会します。エンベディングモデルのデータ形式と一致する必要があります。
<i>topk</i>	INTEGER	0	返される結果の数。`0`はデフォルトの1を使用します。
<i>deduplicate</i>	BOOLEAN	true	複数の「部分」チャンク/ページなどが一致する場合でも、各ソースの結果を1回だけ返します

返品

Column	Type	Description
<i>key</i>	text	ソースレコードキー
<i>value</i>	text	ソーステキストのコンテンツ。NULL 元のソースコンテンツが非テキスト PDF または画像などの場合。他の列はまだ表面一致するメタデータです。
<i>distance</i>	double precision	クエリーからのベクトル距離
<i>part_ids</i>	bigint[]	ソースレコードのどの部分が一致したのかチャンクまたはページなど
<i>pipeline_name</i>	text	このソースレコードを処理するパイプラインの名前
<i>intermediate_steps</i>	jsonb	マッチングは中間ステップからの結果です。つまり、「part_ids」に属する値

例

```
SELECT * FROM aidb.retrieve_text(my_kb, waterproof jacket, 3);
```

__OUTPUT__

key	value	distance
19337	Men Stripes Waterproof Shell Jacket	0.2994317672742334
55018	Women All-Weather Hiking Anorak	0.3804609668507203

(2 rows)

promoteb.delete_knowledge_base

カスケードナレッジベースに接続されているすべてのパイプラインを削除し、ベクターテーブルを削除し、レジストリエントリを削除します。基になるソーステーブルは削除しません。

KB と他のパイプラインのベクトルをそのまま維持しながら、マルチパイプライン KB から単一のパイプラインを削除するには、代わりに `aidb.delete_pipeline` を使用します。最後に接続されたパイプラインを削除すると、KB がドロップされます。

パラメーター

Parameter	Type	Description
<i>knowledge_base_name</i>	TEXT	ナレッジベース識別子 <i>schema.vector_table</i> 。

例

```
SELECT aidb.delete_knowledge_base(public.customer_feedback_kb);
```

9.2.4 ボリュームファンクション**promoteb.create_volume**

PGFS ストレージ場所から AIDB ボリュームを作成します。ボリュームは、格納場所のコンテンツを行として公開する外部テーブルであり、ナレッジベースデータソースとして使用したり、SQL で直接照会したりできます。最初に PGFS ストレージの場所を設定するには、[Configuring external storage](#) を参照してください。

パラメーター

Parameter	Type	Description
<i>name</i>	TEXT	ボリュームの名前。有効な引用符のない PostgreSQL 識別子である必要があります。
<i>server_name</i>	TEXT	PGFS 格納場所の名前。
<i>path</i>	TEXT	保存場所内のオプションのサブパス。単一の末尾のスラッシュに正規化されましたたとえば、 <i>foo</i> 、 <i>/foo</i> 、 <i>foo/</i> 、 <i>/foo/</i> はすべて <i>foo/</i> として保存されます。`NULL`または <code>/</code> として保存されます。
<i>data_format</i>	TEXT	データ形式 <i>Text</i> 、 <i>Image</i> 、または <i>Pdf</i>

例

```
SELECT aidb.create_volume(pdf_volume, my_s3_location, /, Pdf);
```

helpb.volumes

データベース内のすべての AIDB ボリュームをリストするビュー。schema、volume、storage_location、および path を返します。

```
SELECT * FROM aidb.volumes;
```

promoteb.delete_volume

AIDB ボリュームを削除します。

!!!note 基になる PGFS ストレージの場所を削除すると、その上に構築されたすべてのボリュームも削除されます。

パラメーター

Parameter	Type	Description
<i>volume_name</i>	TEXT	削除するボリュームの名前。

```
SELECT aidb.delete_volume(pdf_volume);
```

9.3 Models reference

AIDB のすべてのモデル関連のファンクションとビューのリファレンス。ガイドスタイルのドキュメントについては、*Integrating models* を参照してください。

9.3.1 カタログビュー

helpb.model_providers

システムに登録されている使用可能なモデルプロバイダーをすべてリストします。

Column	Type	Description
<i>server_name</i>	name	モデルプロバイダーの名前
<i>server_description</i>	text	プロバイダーの説明

次のページに続く

表 29 – 前のページからの続き

Column	Type	Description
<i>server_options</i>	text[]	使用可能な構成オプション

helpb.models

ビルトインおよびユーザー作成モデルを含む、レジストリ内のすべてのモデルをリストします。

Column	Type	Description
<i>name</i>	text	モデルのユーザー定義の名前
<i>provider</i>	text	モデルプロバイダー名
<i>options</i>	jsonb	モデルの設定済みオプション

例

```
SELECT * FROM aidb.models;
```

__OUTPUT__

```

name | provider | options
-----+-----+-----
bert | bert_local | {"config={}"}
```

```
clip | clip_local | {"config={}"}
```

```
t5 | t5_local | {"config={}"}
```

(3 rows)

9.3.2 モデル管理機能

promoteb.create_model

新しいモデルを AIDB モデルレジストリに登録します。

パラメーター

Parameter	Type	Default	Description
<i>name</i>	TEXT	Required	モデルの一意の名前。
<i>provider</i>	TEXT	Required	プロバイダー名 `aidb.model_providers`を参照してください。
<i>config</i>	JSONB	'{}'	プロバイダー固有の構成。 モデル構成ヘルパー でビルドします。

次のページに続く

表 31 – 前のページからの続き

Parameter	Type	Default	Description
<i>credentials</i>	JSONB	<code>{}</code>	プロバイダーの資格情報たとえば <code>{"api_key": "..."} </code> 。
<i>replace_credentials</i>	BOOLEAN	<i>false</i>	<code>true</code> の場合、モデルを再作成せずに保存されている資格情報を更新します。

TLS 構成

HTTPS モデルエンドポイントに接続するには、`config` 内に `tls_config` フィールドを含めます。

```
"tls_config": {
  "insecure_skip_verify": true,
  "ca_path": "/etc/aidb/myCA.pem"
}
```

例

```
- - Minimal registration
SELECT aidb.create_model(my_t5, t5_local);

- - With config and credentials
SELECT aidb.create_model(
  name      => my_t5,
  provider  => t5_local,
  config    => {"param1": "value1"}::JSONB,
  credentials => {"token": "abcd"}::JSONB
);

- - Rotate credentials without re-creating
SELECT aidb.create_model(
  name      => my_openai,
  provider  => openai_embeddings,
  config    => aidb.embeddings_config(model => text-embedding-3-small),
  credentials => {"api_key": "<new-key>"}::JSONB,
  replace_credentials => true
);
```

promoteb.get_model

登録されたモデルの構成を返します。

パラメーター

Parameter	Type	Description
<i>model_name</i>	TEXT	取得するモデルの名前。

返品

Column	Type	Description
<i>name</i>	text	モデル名
<i>provider</i>	text	プロバイダー名
<i>options</i>	jsonb	設定されたオプション

例

```
SELECT * FROM aidb.get_model(t5);
```

__OUTPUT__

```
name | provider | options
-----+-----+-----
t5   | t5_local | {"config={}"}
```

(1 row)

promoteb.delete_model

レジストリからモデルを削除します。次に実行されるまで、それを参照するパイプラインまたはナレッジベースには影響しません。

パラメーター

Parameter	Type	Description
<i>model_name</i>	TEXT	削除するモデルの名前。

返品

JSONB としての削除されたモデルの名前、プロバイダー、およびオプション。

例

```
SELECT aidb.delete_model(t5);

__OUTPUT__
      delete_model
-----
(t5,t5_local,{"config={}"})
(1 row)
```

9.3.3 HCP モデルファンクション

これらの機能は、EDB Hybrid ManagerHCP で実行されているモデルを管理します。

promoteb.list_hcp_models

ハイブリッドマネージャーで現在実行されているモデルをリストします。

返品

Column	Type	Description
<i>name</i>	text	HCP でのモデルインスタンス名
<i>url</i>	text	API エンドポイント URL
<i>model</i>	text	モデル識別子

例

```
SELECT * FROM aidb.list_hcp_models();

__OUTPUT__
      name          |          url
--|-----
→|      model
-----+-----
→+-----+-----
llama-3-1-8b-instruct-1xgpu | http://llama-3-1-8b-predictor.default.svc.local
→| meta/llama-3.1-8b-instruct
(1 row)
```

promoteb.create_hcp_model

実行中のインスタンス名を参照することにより、HCP ホストモデルを登録します。

パラメーター

Parameter	Type	Description
<i>name</i>	TEXT	AIDB 内のモデルのユーザー定義の名前。
<i>hcp_model_name</i>	TEXT	HCP で実行されているモデルインスタンスの名前。

promoteb.sync_hcp_models

AIDB モデルレジストリを、現在 HCP で実行されているモデルと同期します。新しい HCP モデルのエントリを作成し、そこで実行されなくなったモデルのエントリを削除します。

返品

Column	Type	Description
<i>status</i>	text	<i>created</i> 、 <i>deleted</i> 、 <i>unchanged</i> 、または <i>skipped</i>
<i>model</i>	text	同期モデルの名前

9.3.4 推論ファンクション**helpb.encode_text**

指定されたモデルを使用して、単一のテキスト文字列をベクトルにエンコードします。

パラメーター

Parameter	Type	Description
<i>model_name</i>	TEXT	登録されているモデルの名前。
<i>input</i>	TEXT	エンコードするテキスト。

返品

`real[]` — 入力テキストの埋め込み。 `vector` にキャストして `pgvector` 演算子で使います。

例

```
SELECT aidb.encode_text(bert_local, The quick brown fox);
```

helpb.encode_text_batch

1 回の呼び出しでテキスト文字列の配列をベクトルにエンコードします。

パラメーター

Parameter	Type	Description
<i>model_name</i>	TEXT	登録されているモデルの名前。
<i>input</i>	TEXT[]	エンコードする文字列の配列。

返品

SETOF real[] —入力ごとに 1 行、入力オーダー。結果列の名前は `encode_text_batch` です。

promoteb.encode_image

マルチモーダルモデル CLIP などを使用して、バイナリイメージをベクトルにエンコードします。

パラメーター

Parameter	Type	Description
<i>model_name</i>	TEXT	登録されているモデルの名前。
<i>input</i>	BYTEA	Raw 画像バイト。

返品

real[] —入力画像の埋め込み。 `vector` にキャストして `pgvector` 演算子で使します。

例

```
SELECT aidb.encode_image(clip_local, pg_read_binary_file('/tmp/photo.jpg')::BYTEA);
```

promoteb.decode_text

指定されたモデルを使用して、プロンプトからテキスト応答を生成します。

パラメーター

Parameter	Type	Default	Description
<i>model_name</i>	TEXT	Required	登録されているモデルの名前。
<i>input</i>	TEXT	Required	テキストプロンプトまたは入力。
<i>inference_config</i>	JSON	NULL	実行時推論設定。 :ref:`aidb.inference_config()` <#aidbinference_config>` を使用 します。結果を `json` にキャスト します。

返品

TEXT — 生成されたレスポンス

例

```
- - Basic usage
SELECT aidb.decode_text(t5_local, translate to French: Hello, world.);

- - With inference configuration
SELECT aidb.decode_text(
  my_llama,
  Explain quantum computing in one sentence.,
  aidb.inference_config(
    system_prompt => Be concise and factual.,
    temperature   => 0.7,
    max_tokens    => 100
  )::json
);
```

promoteb.decode_text_batch

1 回の呼び出しで一連のプロンプトに対するテキスト応答を生成します。

パラメーター

Parameter	Type	Default	Description
<i>model_name</i>	TEXT	Required	登録されているモデルの名前。
<i>input</i>	TEXT[]	Required	テキストプロンプトの配列。
<i>inference_config</i>	JSON	NULL	実行時推論設定。 :ref:`aidb.inference_config()` <#aidbinference_config>` を使用 します。結果を `json` にキャスト します。

返品

SETOF TEXT —入力プロンプトごとに 1 行、入力順。結果列の名前は `decode_text_batch` です。

例

```
SELECT aidb.decode_text_batch(t5_local, ARRAY[
    translate to German: hello,
    translate to German: goodbye
]);
```

helpb.rerank_text

リランキングモデルを使用して、クエリに対してテキスト入力セットをスコアリングおよびランク付けします。

パラメーター

Parameter	Type	Default	Description
<i>model_name</i>	TEXT	Required	登録されたリランキングモデルの名前。
<i>query</i>	TEXT	Required	入力をランク付けするクエリー。
<i>input</i>	TEXT[]	[]	ランク付けの候補テキストの配列。

返品

Column	Type	Description
<i>text</i>	text	候補テキスト
<i>logit_score</i>	double precision	関連性スコア高いほど関連性が高い

次のページに続く

表 44 – 前のページからの続き

Column	Type	Description
<i>id</i>	int	入力配列内のテキストの元のインデックス

例

```
SELECT * FROM aidb.rerank_text(
  my_reranker,
  How do I configure AIDB?,
  ARRAY[
    AIDB requires shared_preload_libraries.,
    Postgres supports JSON natively.,
    Run CREATE EXTENSION aidb CASCADE to install.
  ]
) ORDER BY logit_score DESC;
```

9.3.5 推論構成ヘルパー

helpb.inference_config

ランタイム推論設定の JSONB 構成オブジェクトをビルドします。結果 `json` へのキャストを `aidb.decode_text()`、`aidb.decode_text_batch()` に渡すか、`aidb.summarize_text_config()` で使用します。

すべてのパラメーターはオプション - 必要ないものは省略します。

パラメーター

Parameter	Type	Default	Description
<i>system_prompt</i>	TEXT	NULL	要求の前に追加されるシステムプロンプト。T5 モデルではサポートされていません。
<i>temperature</i>	DOUBLE PRECISION	NULL	サンプリング温度。`0.0`は決定的です。値が高いほど多様性が増します。
<i>max_tokens</i>	INTEGER	NULL	生成する最大トークン。
<i>top_p</i>	DOUBLE PRECISION	NULL	nucleus サンプリングしきい値。
<i>seed</i>	BIGINT	NULL	再現可能な出力のランダムシード。

次のページに続く

表 45 – 前のページからの続き

Parameter	Type	Default	Description
<i>repeat_penalty</i>	REAL	NULL	トークンを繰り返す場合のペナルティ。 <i>1.0</i> =ペナルティなし。NIM または OpenAI ではサポートされていません。
<i>repeat_last_n</i>	INTEGER	NULL	`repeat_penalty` と考えられる最近のトークンの数。 NIM または OpenAI ではサポートされていません。
<i>thinking</i>	BOOLEAN	NULL	<think> タグの処理を制御します。 `true`/`NULL` はタグを保持します。 `false` はそれらを削除します。
<i>extra_args</i>	JSONB	NULL	API に直接渡される追加のプロバイダー固有の引数。

返品

JSONB — 推論ファンクションに渡す前に `::json` にキャストします。

例

```
- - With temperature and token limit
SELECT aidb.decode_text(
  my_llama,
  What is machine learning?,
  aidb.inference_config(temperature => 0.5, max_tokens => 150)::json
);

- - Full configuration
SELECT aidb.decode_text(
  my_llama,
  Explain relativity.,
  aidb.inference_config(
    system_prompt => You are a physics professor. Be precise.,
    temperature   => 0.3,
    max_tokens    => 200,
    top_p         => 0.9,
    seed          => 42,
    repeat_penalty => 1.1
  )::json
);

- - Hide reasoning tags from a thinking model
```

(次のページに続く)

(前のページからの続き)

```
SELECT aidb.decode_text(
  my_reasoning_model,
  Solve: 2x + 5 = 15,
  aidb.inference_config(thinking => false)::json
);
```

9.3.6 モデル構成ヘルパー

これらの関数は、`aidb.create_model()` の `config` パラメーターの JSONB 構成オブジェクトを結果ます。各ヘルパーは、特定のモデルプロバイダーに対応します。

`promoteb.embeddings_config`

`openai_embeddings` および OpenAI 互換の埋め込みエンドポイントの場合。

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	モデル識別子。
<i>api_key</i>	TEXT	NULL	API キー。
<i>url</i>	TEXT	NULL	エンドポイント URL のオーバーライド。
<i>basic_auth</i>	TEXT	NULL	基本認証 <i>user:password</i>
<i>max_concurrent_requests</i>	INTEGER	NULL	最大同時要求。
<i>max_batch_size</i>	INTEGER	NULL	バッチごとの最大入力数。
<i>input_type</i>	TEXT	NULL	入力タイプのヒントプロバイダー固有。
<i>input_type_query</i>	TEXT	NULL	クエリ入力タイプのヒントプロバイダー固有。
<i>tls_config</i>	JSONB	NULL	TLS 構成。
<i>is_hcp_model</i>	BOOLEAN	NULL	<i>true</i> モデルが HCP で実行される場合。

```
SELECT aidb.create_model(
  my_embedder,
  openai_embeddings,
  config => aidb.embeddings_config(
    model    => text-embedding-3-small,
    api_key  => sk-...,
    url      => https://api.openai.com/v1
  )
);
```

promoteb.completions_config

openai_completions および OpenAI 互換のコンプリートエンドポイントの場合。

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	モデル識別子。
<i>api_key</i>	TEXT	NULL	API キー。
<i>url</i>	TEXT	NULL	エンドポイント URL のオーバーライド。
<i>basic_auth</i>	TEXT	NULL	基本認証 <i>user:password</i>
<i>system_prompt</i>	TEXT	NULL	デフォルトのシステムプロンプト。
<i>temperature</i>	DOUBLE PRECISION	NULL	サンプリング温度。
<i>top_p</i>	DOUBLE PRECISION	NULL	nucleus サンプリングしきい値。
<i>seed</i>	BIGINT	NULL	ランダムシード。
<i>thinking</i>	BOOLEAN	NULL	`<think>` タグの処理。
<i>max_tokens</i>	JSONB	NULL	最大トークン構成 `aidb.max_tokens_config()` を使用します。
<i>max_concurrent_requests</i>	INTEGER	NULL	最大同時要求。
<i>extra_args</i>	JSONB	NULL	追加のプロバイダー固有の引数。
<i>is_hcp_model</i>	BOOLEAN	NULL	<i>true</i> モデルが HCP で実行される場合。

```
SELECT aidb.create_model(
  my_llm,
  openai_completions,
  config => aidb.completions_config(
    model      => gpt-4o,
    api_key    => sk-...,
    temperature => 0.2
  )
);
```

promoteb.max_tokens_config

aidb.completions_config() で使用する max_tokens オブジェクトをビルドします。

Parameter	Type	Default	Description
<i>size</i>	INTEGER	Required	生成する最大トークン。

次のページに続く

表 48 – 前のページからの続き

Parameter	Type	Default	Description
<i>format</i>	TEXT	NULL	形式 'default'、'legacy'、または 'both'

```
SELECT aidb.completions_config(
  model      => gpt-4o,
  max_tokens => aidb.max_tokens_config(size => 1024, format => default)
);
```

promoteb.bert_config

bert_local プロバイダーの場合。

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	HuggingFace モデル識別子。
<i>revision</i>	TEXT	NULL	モデルのリビジョンまたはブランチ。
<i>cache_dir</i>	TEXT	NULL	ローカルキャッシュディレクトリ。

```
SELECT aidb.create_model(my_bert, bert_local,
  config => aidb.bert_config(sentence-transformers/all-MiniLM-L6-v2));
```

promoteb.clip_config

clip_local プロバイダーの場合。

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	HuggingFace モデル識別子。
<i>revision</i>	TEXT	NULL	モデルのリビジョンまたはブランチ。
<i>cache_dir</i>	TEXT	NULL	ローカルキャッシュディレクトリ。
<i>image_size</i>	INTEGER	NULL	ピクセル単位の入力画像サイズ。

```
SELECT aidb.create_model(my_clip, clip_local,
  config => aidb.clip_config(openai/clip-vit-base-patch32));
```

promoteb.llama_config

llama_local プロバイダーの場合。

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	モデル識別子または HuggingFace パス。
<i>revision</i>	TEXT	NULL	モデルリビジョン。
<i>cache_dir</i>	TEXT	NULL	ローカルキャッシュディレクトリ。
<i>model_path</i>	TEXT	NULL	モデルの重みへの明示的なローカルパス、 <i>model</i> をオーバーライドします。
<i>system_prompt</i>	TEXT	NULL	デフォルトのシステムプロンプト。
<i>temperature</i>	DOUBLE PRECISION	NULL	サンプリング温度。
<i>top_p</i>	DOUBLE PRECISION	NULL	nucleus サンプリングしきい値。
<i>seed</i>	BIGINT	NULL	ランダムシード。
<i>sample_len</i>	INTEGER	NULL	生成する最大トークン。
<i>repeat_penalty</i>	REAL	NULL	リピートペナルティ。
<i>repeat_last_n</i>	INTEGER	NULL	反復ペナルティの対象とされるトークン。
<i>use_flash_attention</i>	BOOLEAN	NULL	フラッシュアテンションを有効にします。
<i>use_kv_cache</i>	BOOLEAN	NULL	KV キャッシュを有効にします。

```
SELECT aidb.create_model(my_llama, llama_local,
  config => aidb.llama_config(
    meta-llama/Llama-3.2-3B-Instruct,
    temperature => 0.5
  )
);
```

promoteb.t5_config

t5_local プロバイダーの場合。

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	モデル識別子または HuggingFace パス。

次のページに続く

表 52 – 前のページからの続き

Parameter	Type	Default	Description
<i>revision</i>	TEXT	NULL	モデルリビジョン。
<i>model_path</i>	TEXT	NULL	モデルの重みへの明示的なローカルパス。
<i>cache_dir</i>	TEXT	NULL	ローカルキャッシュディレクトリ。
<i>temperature</i>	DOUBLE PRECISION	NULL	サンプリング温度。
<i>top_p</i>	DOUBLE PRECISION	NULL	nucleus サンプリングしきい値。
<i>seed</i>	BIGINT	NULL	ランダムシード。
<i>max_tokens</i>	INTEGER	NULL	生成する最大トークン。
<i>repeat_penalty</i>	REAL	NULL	リピートペナルティ。
<i>repeat_last_n</i>	INTEGER	NULL	反復ペナルティの対象とされるトークン。

```
SELECT aidb.create_model(my_t5, t5_local,
    config => aidb.t5_config(google/flan-t5-base));
```

promoteb.gemini_config

gemini プロバイダーの場合。

Parameter	Type	Default	Description
<i>api_key</i>	TEXT	Required	Google API キー。
<i>model</i>	TEXT	NULL	Gemini モデル識別子たとえば <i>gemini-2.0-flash</i> 。
<i>url</i>	TEXT	NULL	API エンドポイントのオーバーライド。
<i>max_concurrent_requests</i>	INTEGER	NULL	最大同時要求。
<i>thinking_budget</i>	INTEGER	NULL	拡張思考トークンの予算 Gemini 2.x のみ。

```
SELECT aidb.create_model(my_gemini, gemini,
    config => aidb.gemini_config(AIza..., model => gemini-2.0-flash));
```

promoteb.nim_clip_config

nim_clip プロバイダー NVIDIA NIM マルチモーダル埋め込みの場合。

Parameter	Type	Default	Description
<i>api_key</i>	TEXT	NULL	NIM API キー。
<i>model</i>	TEXT	NULL	NIM CLIP モデル識別子。
<i>url</i>	TEXT	NULL	NIM エンドポイント URL のオーバーライド。
<i>basic_auth</i>	TEXT	NULL	基本的な認証資格情報。
<i>is_hcp_model</i>	BOOLEAN	NULL	<i>true</i> モデルが HCP で実行される場合。

```
SELECT aidb.create_model(my_nim_clip, nim_clip,
    config => aidb.nim_clip_config(api_key => nvapi-..., model => nvidia/nvclip));
```

promoteb.nim_ocr_config

nim_ocr プロバイダー NVIDIA NIM OCR の場合。

Parameter	Type	Default	Description
<i>api_key</i>	TEXT	NULL	NIM API キー。
<i>model</i>	TEXT	NULL	NIM OCR モデル識別子。
<i>url</i>	TEXT	NULL	NIM エンドポイント URL のオーバーライド。
<i>basic_auth</i>	TEXT	NULL	基本的な認証資格情報。
<i>is_hcp_model</i>	BOOLEAN	NULL	<i>true</i> モデルが HCP で実行される場合。

```
SELECT aidb.create_model(my_nim_ocr, nim_ocr,
    config => aidb.nim_ocr_config(api_key => nvapi-...));
```

promoteb.nim_reranking_config

nim_reranking プロバイダーの場合。

Parameter	Type	Default	Description
<i>api_key</i>	TEXT	NULL	NIM API キー。
<i>model</i>	TEXT	NULL	NIM リランキングモデル識別子。

次のページに続く

表 56 – 前のページからの続き

Parameter	Type	Default	Description
<i>url</i>	TEXT	NULL	NIM エンドポイント URL のオーバーライド。
<i>basic_auth</i>	TEXT	NULL	基本的な認証資格情報。
<i>is_hcp_model</i>	BOOLEAN	NULL	<i>true</i> モデルが HCP で実行される場合。

```
SELECT aidb.create_model(my_reranker, nim_reranking,
  config => aidb.nim_reranking_config(
    api_key => nvapi-...,
    model   => nvidia/nv-rerankqa-mistral-4b-v3
  )
);
```

promoteb.openrouter_chat_config

openrouter_chat プロバイダーの場合。

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	OpenRouter モデル識別子。
<i>api_key</i>	TEXT	NULL	OpenRouter API キー。
<i>url</i>	TEXT	NULL	API エンドポイントのオーバーライド。
<i>max_concurrent_requests</i>	INTEGER	NULL	最大同時要求。
<i>max_tokens</i>	JSONB	NULL	最大トークン構成 `aidb.max_tokens_config()`を使用します。

```
SELECT aidb.create_model(my_or_chat, openrouter_chat,
  config => aidb.openrouter_chat_config(
    anthropic/claude-3-5-haiku,
    api_key => sk-or-...
  )
);
```

promoteb.openrouter_embeddings_config

openrouter_embeddings プロバイダーの場合。

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	OpenRouter エンベディングモデル識別子。
<i>api_key</i>	TEXT	NULL	OpenRouter API キー。
<i>url</i>	TEXT	NULL	API エンドポイントのオーバーライド。
<i>max_concurrent_requests</i>	INTEGER	NULL	最大同時要求。
<i>max_batch_size</i>	INTEGER	NULL	バッチごとの最大入力数。

```
SELECT aidb.create_model(my_or_embedder, openrouter_embeddings,
  config => aidb.openrouter_embeddings_config(
    mistral/mistral-embed,
    api_key => sk-or-...
  )
);
```

9.4 Operations reference

パイプラインエラーログ機能のリファレンス。ガイドスタイルのドキュメントについては、*Pipeline error log* を参照してください。

9.4.1 エラーログファンクション

各パイプラインは、`{source_schema}.pipeline_{pipeline_name}_errors` でエラーログテーブルを維持します。これらの機能は、そのログを照会および管理するための一貫したインターフェイスを提供します。

promoteb.get_error_logs

オプションのフィルターを使用して、パイプラインのエラーログエントリを結果ます。

パラメーター

Parameter	Type	Default	Description
<i>pipeline_name</i>	TEXT	Required	パイプラインの名前。
<i>p_source_id</i>	TEXT	NULL	特定のソースレコード ID にフィルターします。
<i>p_pipeline_step</i>	SMALLINT	NULL	特定のパイプラインステップ番号にフィルターします。

次のページに続く

表 59 – 前のページからの続き

Parameter	Type	Default	Description
<i>p_error_category</i>	TEXT	NULL	カテゴリによるフィルター <i>RecordTemporary</i> 、 <i>RecordPermanent</i> 、 <i>PipelineTemporary</i> 、または `PipelinePermanent`
<i>p_limit</i>	INTEGER	NULL	返される行の最大数。
<i>p_offset</i>	INTEGER	NULL	スキップする行数ページネーションの場合。

返品

Column	Type	Description
<i>id</i>	bigint	エラーログエントリ ID
<i>source_id</i>	text	失敗したソースレコード
<i>part_ids</i>	bigint[]	部分識別子チャンクレコードの場合
<i>pipeline_step</i>	smallint	障害が発生したステップ番号
<i>step_operation</i>	text	オペレーション名 例 <i>ParsePdf</i> 、 <i>ChunkText</i>
<i>error_message</i>	text	エラーの説明
<i>error_category</i>	aidb.ErrorBlocking	エラーカテゴリ
<i>failed_at</i>	timestampz	障害のタイムスタンプ
<i>retry_count</i>	integer	リトライ試行回数
<i>last_retry_at</i>	timestampz	前回の再試行のタイムスタンプ

例

```

-- All errors for a pipeline
SELECT * FROM aidb.get_error_logs(my_pipeline);

-- Filter by source record
SELECT * FROM aidb.get_error_logs(my_pipeline, p_source_id => doc_42);

-- Filter by step and category
SELECT * FROM aidb.get_error_logs(
    my_pipeline,
    p_pipeline_step => 1::smallint,
    p_error_category => RecordPermanent
);

-- Paginate
SELECT * FROM aidb.get_error_logs(my_pipeline, p_limit => 20, p_offset => 40);

```

helpb.get_error_log_summary

単一のパイプラインの集計エラー数をステップ、オペレーション、およびカテゴリごとにグループ化して返します。

パラメーター

Parameter	Type	Description
<i>pipeline_name</i>	TEXT	パイプラインの名前。

返品

Column	Type	Description
<i>pipeline_step</i>	smallint	ステップ番号
<i>step_operation</i>	text	オペレーション名
<i>error_category</i>	text	エラーカテゴリ
<i>error_count</i>	bigint	このグループのエラーの数
<i>latest_failed_at</i>	timestamptz	最新の障害のタイムスタンプ

例

```
SELECT * FROM aidb.get_error_log_summary(my_pipeline);

__OUTPUT__
pipeline_step | step_operation | error_category | error_count | latest_failed_at
-----+-----+-----+-----+-----
1 | ParsePdf      | RecordPermanent | 2 | 2026-04-10 14:31:58.
456+00
2 | ChunkText     | RecordPermanent | 1 | 2026-04-10 14:32:01.
123+00
(2 rows)
```

promoteb.get_all_error_summaries

aidb.get_error_log_summary() と同じロールアップを返しますが、一度にすべてのパイプラインを返します。

返品

Column	Type	Description
<i>pipeline_name</i>	text	パイプライン名
<i>pipeline_step</i>	smallint	ステップ番号
<i>step_operation</i>	text	オペレーション名
<i>error_category</i>	text	エラーカテゴリ
<i>error_count</i>	bigint	このグループのエラーの数
<i>latest_failed_at</i>	timestampz	最新の障害のタイムスタンプ

例

```
SELECT * FROM aidb.get_all_error_summaries();
```

```
__OUTPUT__
```

```

pipeline_name | pipeline_step | step_operation | error_category | error_count |
→latest_failed_at
-----+-----+-----+-----+-----+
→-----
my_pipeline   |              1 | ParsePdf      | RecordPermanent |          2 |
→2026-04-10 14:31:58.456+00
pdf_ingester  |              1 | ParsePdf      | PipelinePermanent |          1 |
→2026-04-10 15:00:12.000+00
(2 rows)
```

promoteb.clear_error_logs

ID に基づいて特定のエラーログエントリを削除します。クリアする前に `aidb.get_error_logs()` でエントリを確認します。

パラメーター

Parameter	Type	Description
<i>pipeline_name</i>	TEXT	パイプラインの名前。
<i>error_ids</i>	BIGINT[]	削除するエラーログエントリ ID の配列。

返品

INTEGER — 削除された行数。

例

```
SELECT aidb.clear_error_logs(my_pipeline, ARRAY[1, 3]);
```

__OUTPUT__

clear_error_logs

```
-----
                2
(1 row)
```

promoteb.retry_pipeline_errors

!!!注 aidb.retry_pipeline_errors() はまだ利用できません。将来のリリースに含まれる予定です。
[Known issues](#) を参照してください。

特定のエラーログエントリを、最初に失敗したステップから再処理します。

パラメーター

Parameter	Type	Description
<i>pipeline_name</i>	TEXT	パイプラインの名前。
<i>error_ids</i>	BIGINT[]	再試行するエラーログエントリ ID の配列。

返品

Column	Type	Description
<i>error_id</i>	bigint	エラーログエントリ ID
<i>status</i>	text	<i>resolved</i> または <i>failed</i>
<i>error_message</i>	text	失敗の理由 解決された場合は空

解決されたエントリはエラーログから削除されます。失敗した再試行は、インクリメントされた `retry_count` および更新された `last_retry_at` とともに残ります。

例

```
SELECT * FROM aidb.retry_pipeline_errors(my_pipeline, ARRAY[1, 2]);
```

__OUTPUT__

error_id | status | error_message

```
-----+-----+-----
      1 | resolved |
      2 | failed   | failed to extract text from page 0
(2 rows)
```

9.4.2 エラーカテゴリ

Category	Scope	Blocks pipeline?	Description
<i>RecordTemporary</i>	Single record	No	1 つのレコードの一時的な障害。再試行で成功する場合があります。
<i>RecordPermanent</i>	Single record	No	1 つのレコードの永続的な障害たとえば、破損したデータ。
<i>PipelineTemporary</i>	Entire step	Yes	ステップを停止した一時的な障害たとえば、サービスの停止。再試行で解決される場合があります。
<i>PipelinePermanent</i>	Entire step	Yes	ステップを停止した永続的な障害たとえば、無効なモデル構成。

9.5 Pipelines reference

完全なパイプライン API リファレンスは、パイプラインドキュメントとともにあります。 [Pipelines API reference](#) を参照してください。

Category	Items
Types and enums	<i>aidb.PipelineAutoProcessingMode</i> , <i>aidb.PipelineDataFormat</i> , <i>aidb.PipelineSourceType</i> , <i>aidb.PipelineDestinationType</i> , <i>aidb.PipelineStepOperation</i> , <i>aidb.PipelineStatus</i> , <i>aidb.DistanceOperator</i>
Domains	<i>aidb.pipeline_name_50</i> , <i>aidb.background_sync_interval</i>
Views	<i>aidb.pipelines</i> , <i>aidb.pipeline_metrics</i>
Functions	<i>aidb.create_pipeline</i> , <i>aidb.update_pipeline</i> , <i>aidb.delete_pipeline</i> , <i>aidb.run_pipeline</i> , <i>aidb.get_pipeline_metrics</i>
Pipeline step config helpers	<i>aidb.chunk_text_config</i> , <i>aidb.summarize_text_config</i> , <i>aidb.ocr_config</i> , <i>aidb.html_parse_config</i> , <i>aidb.pdf_parse_config</i> , <i>aidb.knowledge_base_config</i> , <i>aidb.knowledge_base_config_from_kb</i> , <i>aidb.inference_config</i>
Vector index config helpers	<i>aidb.vector_index_hnsw_config</i> , <i>aidb.vector_index_ivfflat_config</i> , <i>aidb.vector_index_hsphere_optimized_config</i> , <i>aidb.vector_index_disabled_config</i>

第10章 AIDB release notes

AIDB は、AI 搭載アプリケーションを構築するための EDB の PostgreSQL 拡張機能です。ナレッジベース管理、データの準備、および取得拡張生成 RAG のパイプラインと、内部および外部の AI モデルの統合を提供します。

AIDB ドキュメントでは、AIDB の最新バージョンについて説明しています。これらのリリースノートは、各リリースの新機能に関する情報を提供します。

Version	Release date
AIDB 7.4.0 release notes	06 Jul 2026
AIDB 7.3.2 release notes	17 Apr 2026
AIDB 7.3.0 release notes	08 Apr 2026
AIDB 7.2.0 release notes	30 Mar 2026
6.1.0	26 Feb 2026
6.0.0	10 Feb 2026
5.3.0	04 Dec 2025
5.2.0	26 Nov 2025
5.1.0	05 Nov 2025
5.0.0	29 Oct 2025
4.5.0	10 Sep 2025
4.4.0	13 Aug 2025
4.3.0	04 Jul 2025
4.2.0	16 Jun 2025
4.1.1	28 May 2025
4.1.0	19 May 2025
4.0.1	09 May 2025
4.0.0	05 May 2025
3.0.1	03 Apr 2025
2.2.1	13 Mar 2025
2.1.2	24 Feb 2025
2.1.1	03 Feb 2025
2.0.0	13 Jan 2025
1.0.7	10 Dec 2024

10.1 AIDB 7.x release notes

AIDB メジャーバージョン 7 のリリースノート。

Version	Release date
AIDB 7.4.0 release notes	06 Jul 2026
AIDB 7.3.2 release notes	17 Apr 2026
AIDB 7.3.0 release notes	08 Apr 2026
AIDB 7.2.0 release notes	30 Mar 2026

10.1.1 AIDB 7.2.0 release notes

リリース日 2026 年 3 月 30 日

AIDB 7.2.0 には、次の機能強化とバグ修正が含まれています。

Type	Description
Enhancement	マルチパイプラインのナレッジベース ナレッジベースをマルチプルのパイプラインに関連付けできるようになり、より豊富で柔軟なデータの取り込みおよび取得アーキテクチャが可能になります。
Enhancement	ハイブリッド検索 BYOSQL 持込 SQL BYOSQL インターフェイスを使用した階層 1 ハイブリッド検索サポートを導入し、カスタム取得ロジックをセマンティック検索と組み合わせることができます。
Enhancement	マルチステップパイプラインの取得サポート マルチステップパイプライン構成内で取得操作がサポートされるようになり、パイプラインステージ全体でエンドツーエンドの RAG ワークフローが有効になります。
Enhancement	HuggingFace API のサポート エンベディングおよびその他の AI タスクのモデルプロバイダーとして、HuggingFace Inference API のサポートを追加しました。
Enhancement	モデルオプション モデル構成にプロバイダー固有のオプションを含めることができるようになり、温度、最大トークン、その他のパラメーターなどのモデルの動作をより詳細に制御できます。
Enhancement	パイプラインデッドレターキュー DLQ – フェーズ 1: パイプラインのデッドレターキューサポートの最初のフェーズが導入されました。失敗したパイプライン操作の処理エラーがログに記録されるようになり、失敗したレコードの検査と再処理が可能になります。
Enhancement	構成可能な CPU 制限 バックグラウンドパイプラインワーカーの構成可能な CPU 制限を追加して、共有環境でのリソース競合を防ぎます。
Enhancement	HCP モデル同期ヒント HCP 同期が必要なモデルをリストする場合、AIDB は HCP からモデルを同期する方法をユーザーにガイドする役立つヒントを表示します。

[次のページに続く](#)

表 3 – 前のページからの続き

Type	Description
Enhancement	オプションバージョン付けインターフェイス OVI パイプラインビューとナレッジベースビューがオプションバージョン付けインターフェイスをサポートするようになり、クライアントは特定の API バージョンにピン留めできます。
Enhancement	非推奨 UX の向上 非推奨機能は、代替機能と移行手順に関するガイダンスを含む明確なメッセージを提供するようになりました。
Enhancement	`aidb.models` ビューの `functions` タグ `aidb.models` ビューには `functions` タグが含まれ、ファンクション呼び出し機能をサポートするモデルを特定しやすくなりました。
Bug fix	ディレクトリを使用したボリュームリストを修正します ボリューム内のファイルと一緒にディレクトリが存在する場合、ボリュームリストが失敗する問題を修正しました。
Change	引用符で囲まれた識別子を含むボリューム名がブロックされる: SQL 参照のあいまいさを防ぐため 、引用符で囲まれた識別子を必要とする名前たとえば、スペースまたは特殊文字を含む名前で作成することはできなくなりました。

10.1.2 AIDB 7.3.0 release notes

リリース日 2026 年 4 月 8 日

AIDB 7.3.0 には、次の機能強化とバグ修正が含まれています。

Type	Description
Enhancement	パイプライン DLQ – フェーズ 2 ノンブロッキングエラー Dead Letter Queue はノンブロッキングエラーを処理するようになりました。これは、一時的なまたは非クリティカルな処理障害がパイプラインの実行を停止しないことを意味します。失敗したレコードは、検査と再処理のためにキューに入れられます。
Enhancement	一時的なエラー分類 モデルエラーとデータベースエラーが必要に応じて `PipelineTemporary` として分類されるようになり、パイプラインが一時的な障害を永続的なエラーとして扱うのではなく、自動的に再試行できます。
Enhancement	DLQ ユーザ API 失敗したパイプラインレコードをリスト、検査、および再処理する機能を含む、デッドレターキューと対話するためのユーザー対応 SQL ファンクションを導入しました。
Enhancement	セマンティックナレッジベース検索のコメント セマンティックナレッジベース検索には、ドキュメントコンテンツとともに保存されているコメントの検索のサポートが含まれるようになりました。

非推奨のインターフェイス

AIDB 7.3.0 は、メジャーバージョンインターフェイスのクリーンアップの一部として、次の SQL ビューと関クションを削除します。それぞれに置換えが記載されています。

Deprecated objects	Replacement
Views	
<i>aidb.preparers</i>	<i>aidb.pipelines</i> (filter by operation type if needed)
<i>aidb.preps</i>	<i>aidb.pipelines</i>
<i>aidb.retrievers</i>	<i>aidb.knowledge_bases</i> or <i>aidb.kbs</i>
<i>aidb.preparer_registry</i>	<i>aidb.pipeline_registry</i> (internal)
<i>aidb.preparer_registry_source_table</i>	<i>aidb.pipeline_registry</i> (internal)
<i>aidb.preparer_registry_source_volume</i>	<i>aidb.pipeline_registry</i> (internal)
Functions	
<i>aidb.create_table_knowledge_base(...)</i>	<i>aidb.create_pipeline(... step_1 => 'KnowledgeBase')</i>
<i>aidb.create_volume_knowledge_base(...)</i>	<i>aidb.create_pipeline(... step_1 => 'KnowledgeBase')</i>
<i>aidb.set_auto_knowledge_base(...)</i>	<i>aidb.update_pipeline(name, auto_processing => ...)</i>
<i>aidb.assert_knowledge_base_preconditions(...)</i>	Removed — no-op; validation is now handled internally
<i>aidb.create_retriever_for_table(...)</i>	<i>aidb.create_pipeline(... step_1 => 'KnowledgeBase')</i>
<i>aidb.create_retriever_for_volume(...)</i>	<i>aidb.create_pipeline(... step_1 => 'KnowledgeBase')</i>
<i>aidb.delete_retriever(...)</i>	<i>aidb.delete_pipeline(...)</i>
<i>aidb.set_retriever_auto_processing(...)</i>	<i>aidb.update_pipeline(name, auto_processing => ...)</i>
<i>aidb.enable_auto_embedding_for_table(...)</i>	<i>aidb.update_pipeline(name, auto_processing => 'Live')</i>
<i>aidb.disable_auto_embedding_for_table(...)</i>	<i>aidb.update_pipeline(name, auto_processing => 'Disabled')</i>
<i>aidb.create_table_preparer(...)</i>	<i>aidb.create_pipeline(...)</i>
<i>aidb.create_volume_preparer(...)</i>	<i>aidb.create_pipeline(...)</i>
<i>aidb.create_preparer_for_table(...)</i>	<i>aidb.create_pipeline(...)</i>
<i>aidb.create_preparer_for_volume(...)</i>	<i>aidb.create_pipeline(...)</i>
<i>aidb.delete_preparer(...)</i>	<i>aidb.delete_pipeline(...)</i>
<i>aidb.set_preparer_auto_processing(...)</i>	<i>aidb.update_pipeline(name, auto_processing => ...)</i>
<i>aidb.get_preparer_data_source(...)</i>	Query <i>aidb.pipelines</i> view directly
<i>aidb.assert_preparer_preconditions(...)</i>	Removed — validation is now handled internally
<i>aidb.assert_table_preparer_preconditions(...)</i>	Removed
<i>aidb.assert_volume_preparer_preconditions(...)</i>	Removed
<i>aidb.bulk_embedding(name)</i>	<i>aidb.run_pipeline(name)</i>
<i>aidb.bulk_data_preparation(name)</i>	<i>aidb.run_pipeline(name)</i>
<i>aidb.single_embedding(name, key, text)</i>	<i>aidb.run_pipeline(name)</i> or direct model calls
<i>aidb.single_embedding(name, key, bytea)</i>	<i>aidb.run_pipeline(name)</i> or direct model calls
<i>aidb.set_auto_preparer(name, mode)</i>	<i>aidb.update_pipeline(name, auto_processing => mode)</i>

次のページに続く

表 5 – 前のページからの続き

Deprecated objects	Replacement
<code>aidb.upsert_chunk_text(name, key, value)</code>	Use pipeline execution via <code>aidb.run_pipeline()</code>
<code>aidb.upsert_summarize_text(name, key, value)</code>	Use pipeline execution via <code>aidb.run_pipeline()</code>
<code>aidb.upsert_parse_html(name, key, value)</code>	Use pipeline execution via <code>aidb.run_pipeline()</code>
<code>aidb.upsert_parse_pdf(name, key, value)</code>	Use pipeline execution via <code>aidb.run_pipeline()</code>
<code>aidb.upsert_perform_ocr(name, key, value)</code>	Use pipeline execution via <code>aidb.run_pipeline()</code>

10.1.3 AIDB 7.3.2 release notes

リリース日 2026 年 4 月 17 日

AIDB 7.3.2 には、次の機能強化とバグ修正が含まれています。

Type	Description
Enhancement	パイプライン DLQ – フェーズ 3 メトリックとエラーログ API メトリックとエラーログ API でデッドレターキューを拡張し、パイプラインごとのエラー数、エラーの詳細、履歴エラーログへのプログラムによるアクセスを提供します。
Enhancement	マルチユーザーパイプライン PGD トリガーラッパー PGD パイプラインのマルチユーザー対応トリガーラッパーを追加し、パイプライン自動処理トリガーがマルチユーザー PGD 環境で正しく動作することを保証します。
Enhancement	PGD レプリケーションセットアップのナレッジベースステーブル ナレッジベースステーブルは、セットアップ中に自動的に PGD レプリケーション構成に含まれ、ナレッジベースデータが PGD ノードにわたってレプリケートされます。
Enhancement	PDF 解析の向上 <code>lopdf</code> ライブラリを、構造 PDF 解析が向上した新しいバージョンに更新しました。その結果、複雑で不正な形式の PDF ドキュメントからのテキスト抽出が向上しました。
Enhancement	HTTP 429 再試行処理 AIDB は、モデルプロバイダー API から HTTP 429 Too Many Requests 応答を受信した要求を自動的に遅延および再試行するようになり、API レート制限に到達した場合の復元力を向上させます。
Enhancement	ソースキーによる取得結果の重複除去 Retrieve 操作はソースキーに基づいて結果の重複を排除するようになり、同じソースドキュメントが複数回インデックスされている場合、または複数のパイプライン実行にわたって参照されている場合に、取得出力に重複レコードが表示されるのを防ぎます。
Bug fix	間違った入力タイプエラーの処理を修正します。 入力タイプ検証エラーは、予期しないエラーとして下流に伝播するのではなく、入力ステージで正しくキャッチおよび表示されます。
Bug fix	マルチパイプライン ID のバグを修正します。 マルチパイプライン構成でパイプライン ID が誤って割り当てられる問題を修正しました。これにより、パイプラインが誤って識別されるか、不正に実行される可能性があります。
Bug fix	自動処理状態のバグを修正する パイプラインが誤った処理状態になるか、その状態に留まる可能性があるパイプライン自動処理システムの状態管理の問題を解決しました。

10.1.4 AIDB 7.4.0 release notes

リリース日 2026 年 7 月 6 日

AIDB 7.4.0 には、次の機能強化とバグ修正が含まれています。

Type	Description
Enhancement	クエリ可能外部テーブルとしてのボリューム 各 AIDB ボリュームは PGFS バックアップの外部テーブルとして公開されるため、標準の SQL <code>INSERT</code> 、 <code>UPDATE</code> 、 <code>DELETE</code> を含むで含まれるオブジェクトをリスト、読み取り、書き込み、および削除できます。オブジェクトに対して、および <code>SELECT body FROM my_volume WHERE key = '...'</code> を使用して単一のオブジェクトを取得します。:ref: `Configuring external storage<Configuring external storage>` を参照してください。
Enhancement	出力ホワイトリスト モデル呼び出しとモデルのダウンロードのオプションのアウトバウンド HTTP ホワイトリスト。`aidb.egress_allowlist` パラメーター共有フォールバックとして `edb.egress_allowlist` を使用して構成します。ホストおよび CIDR エントリを受け入れます。設定されていない場合、チェックは適用されません。
Enhancement	より信頼性の高い HuggingFace モデルのダウンロード ダウンロードは、不安定な接続を介して再開され、サイズと SHA-256 を検証し、進行状況通知を発行し、再試行の間に Ctrl-C に応答するようになりました。破損したキャッシュされたスナップショットは、次のロード時に自己修復します。
Enhancement	厳密なパイプラインオプションの検証 <code>step_N_options</code> 、ナレッジベース構成、およびスタンドアロンヘルパー <code>chunk_text</code> 、 <code>summarize_text</code> 、 <code>parse_html</code> 、 <code>parse_pdf</code> 、 <code>pdf_to_image</code> 、`perform_ocr` の不明なキーと間違ったタイプは、サイレントドロップされる代わりに、明確なエラーを発生させるようになりました。
Enhancement	明確なモデルエラーメッセージ モデルプロバイダーからの HTTP エラーに要求 URL が含まれるようになりました。大きな入力値は、エラー出力で切り捨てられます。OCR 応答は、HTTP 200 で JSON としてのみ解析されます。
Enhancement	バッチのサマライズ `SummarizeText` パイプラインステップは、レート制限エラーに対して入力ごとにリトライを使用して、バッチでレコードを処理するようになりました。
Enhancement	パイプライン作成時のパーミッションエラーの明確化 `SELECT` 特権なしでソーステーブルに対してパイプラインを作成すると、不透明な内部障害の代わりに明示的な「permission denied」エラーが返されるようになりました。
Bug fix	空の取得は空の結果を返します 空のナレッジベースの `retrieve_text` は、標準の SQL `SELECT` 動作と一致して、エラーを生成する代わりに警告を含む行を結果ずに結果ます。
Bug fix	非テキストソースの `retrieve_text` 値列 元のソースが非テキスト PDF や画像などのパイプラインからの取得は、テキストとしてデコードされた生のバイトの代わりに `NULL` を `value` で返すようになりました。一致するメタデータは引き続き返されます。

次のページに続く

表 7 – 前のページからの続き

Type	Description
Bug fix	メトリックコレクターでのデッドロック バックグラウンドメトリックコレクターは、単一のトランザクションですべてのユーザーテーブルのロックを保持しなくなり、`delete_pipeline` に対するデッドロックを削除します。
Bug fix	BIGINT パイプラインキー `int4` 範囲を超える `BIGINT` 主キーを持つソーステーブルのパイプラインは、「out of range for type integer」エラーで失敗しなくなりました。
Bug fix	ステップ全体で尊重されるパイプラインのバッチサイズ レコードチャンクなどを爆発させるステップは、次のステップに渡す前に出力をサブバッチ化し、構成済みの `batch_size` 内でメモリと並列性を維持します。
Bug fix	OCR 構成の検証 OCR モデル構成は、モデル作成時に検証されるようになりました。

第11章 SQL functions

AIDB ファンクションは、スタンドアロン SQL ファンクションとして利用でき、1 回限りの変換、テスト、探索のためにクエリで直接呼び出すことができ、またはパイプラインフレームワークの外部でカスタムワークフローを構築できます。

ここでの各ページは、使用状況と例に焦点を当てています。簡潔なパラメーターテーブルと戻り型については、API リファレンスを参照してください。

- 変換ファンクション `chunk_text`、`parse_html`、`parse_pdf`、`perform_ocr`、`summarize_text` は [Functions reference](#)
- 推論ファンクション `encode_text`、`decode_text`、`rerank_text` は [Reference → Models](#) にあります。

Page	Functions	What it covers
Text chunking	<code>aidb.chunk_text()</code>	Splitting text into segments for LLM context windows
Data parsing	<code>aidb.parse_html()</code> , <code>aidb.parse_pdf()</code>	Extracting readable text from HTML and PDF inputs
Performing OCR	<code>aidb.perform_ocr()</code>	Extracting text from images using an OCR-capable model
Summarization cookbook	<code>aidb.summarize_text()</code> , <code>aidb.summarize_text_aggregate()</code>	Generating and aggregating AI-powered text summaries, use cases, and cookbook
AI inference functions	<code>aidb.encode_text()</code> , <code>aidb.decode_text()</code> , and variants	Embedding text and images into vectors; generating completions
Text reranking	<code>aidb.rerank_text()</code>	Scoring and reordering candidate passages by relevance

!!!注 これらの操作をパイプライン内のステップとして使用するには、[Pipeline steps](#) を参照してください。

11.1 Text chunking

`aidb.chunk_text()` ファンクションを使用して、長い文字列をより小さい意味的に一貫したセグメントにインテリジェントに分割します。このプロセスは、大規模言語モデル LLM のコンテキストウィンドウ制限内にとどまり、ベクターストレージを最適化するために重要です。

```
SELECT * FROM aidb.chunk_text(
  input => This is a significantly longer text example that might require splitting.
```

(次のページに続く)

(前のページからの続き)

```

→into smaller chunks. The purpose of this function is to partition text data into
→segments of a specified maximum length, for example, this sentence 145 is
→characters. This enables processing or storage of data in manageable parts.,
    options => [{"desired_length": 120, "max_length": 150}]
);

__OUTPUT__
part_id | chunk
-----+-----
→-----
    0 | This is a significantly longer text example that might require splitting
→into smaller chunks.
    1 | The purpose of this function is to partition text data into segments of a
→specified maximum length, for example, this sentence 145 is characters.
    2 | This enables processing or storage of data in manageable parts.
(3 rows)

```

11.1.1 構成オプション

- **desired_length** 必須各セグメントのターゲットサイズ。スプリッターは、セマンティック境界を維持しながら、このサイズに到達しようとします。**max_length** を省略すると、これは厳密な上限値になります。
- **max_length** オプションナルチャンクサイズの上限。指定すると、ファンクションは **desired_length** に近いチャンクを生成しようとしますが、より大きな意味単位完全な文や段落などを保持するために **max_length** まで拡張する場合があります。チャンクは、意味のある境界をまたぐカットを回避する必要がある場合にのみ **desired_length** を超えます。単位は使用する **strategy** によって異なります。
- **desired_length = max_length** を指定すると、固定サイズのチャンクたとえば、埋め込みのためにコンテキストウィンドウを正確に埋める場合。
- ソフトリミット内にとどまりたいが、ある程度の柔軟性を許可する場合は、より大きな **max_length** を使用します RAG、要約、または Q&A アプリケーションで一般的な、より高い意味単位を保持します。
- **overlap_length** オプションナル連続するチャンク間でオーバーラップするコンテンツの量。これは、1つのチャンクの終わりから次のチャンクの先頭までコンテンツの一部を複製することにより、チャンク境界を超えてコンテキストを保持するのに役立ちます。デフォルト 0 オーバーラップなし。単位は使用される **strategy** によって異なります。
- **strategy** オプションナル使用するチャンク戦略。文字数によって厳密に分割する文字ベースのチャンクの場合は "chars" デフォルト、またはワード境界を保持するワードベースのチャンクの場合は "words" のいずれかです。これにより、**desired_length**、**max_length**、および **overlap_length** の単位が決まります。"chars" は文字、"words" はワードです。

アルゴリズムの概要

- テキストは、意味的境界の階層を使用して分割されます文字、書記素、単語、文、およびますます長くなる改行シーケンス段落など。
- スプリッターは、指定されたサイズ範囲内に収まる意味的に有効な最大のチャンクを形成しようとします。
- 次のセマンティックユニットの追加が `max_length` を超える場合、`desired_length` より短いチャンクが返される場合があります。

💡 Tip

part_id 列は、チャンクが属する部分の ID を示します。チャンクは、`Creating pipelines` 列で定義されています。

11.2 AI inference functions

AIDB は、クエリで直接 AI 推論を実行するための SQL ファンクションセットを提供します。これらの機能を使用すると、パイプラインを設定せずにデータをベクトル表現にエンコードし、言語モデルを使用してテキストを生成し、推論動作を制御できます。

11.2.1 quietb.encode_text()

登録された埋め込みモデルを使用して、単一のテキスト文字列をベクトルにエンコードします。

```
SELECT aidb.encode_text(my_embedder,What is pgvector?);
```

__OUTPUT__

encode_text

[0.021,-0.043,0.117, ...]

(1 row)

完全なパラメーターのリファレンスについては、[Models reference](#) を参照してください。

11.2.2 promoteb.encode_text_batch()

単一のバッチ要求で複数のテキスト文字列をベクターにエンコードします。多くの入力を処理する場合、ループで `encode_text()` を呼び出すよりも効率的です。

```
SELECT * FROM aidb.encode_text_batch(
  my_model_name,
  ARRAY[What is pgvector?, How does RAG work?, Explain embeddings.]
);
```

(次のページに続く)

(前のページからの続き)

```
__OUTPUT__
encode_text_batch
-----
[0.021,-0.043,0.117,...]
[0.008, 0.091,-0.034,...]
[0.055,-0.012, 0.076,...]
(3 rows)
```

完全なパラメーターのリファレンスについては、[Models reference](#) を参照してください。

11.2.3 quietb.encode_image()

登録されたマルチモーダル埋め込みモデル CLIP モデルなどを使用して、単一の画像をベクトルにエンコードします。

```
SELECT aidb.encode_image(
    my_model_name,
    pg_read_binary_file(/path/to/image.png)::BYTEA
);

__OUTPUT__
          encode_image
-----
[-0.012, 0.084, 0.031, ...]
(1 row)
```

完全なパラメーターのリファレンスについては、[Models reference](#) を参照してください。

11.2.4 quietb.decode_text()

指定されたプロンプトに応じて言語モデル LLM からテキストを生成します。これは、AIDB の LLM 推論のコア機能です。

```
SELECT aidb.decode_text(
    model_name => my_model_name,
    input      => Explain what a knowledge base is in two sentences.,
    inference_config => NULL
);

__OUTPUT__
          decode_text
```

(次のページに続く)

(前のページからの続き)

```

-----
→ -----
A knowledge base is a structured repository of information used to answer queries or.
→ ...
(1 row)

```

生成動作を制御するには、`aidb.inference_config()` によってビルドされた `inference_config` オブジェクトを渡します。呼び出しが `aidb.decode_text(TEXT, TEXT, json)` 署名と一致するように、JSONB 結果を `json` にキャストします。

```

SELECT aidb.decode_text(
  model_name      => my_model_name,
  input           => Summarize the following text: ...,
  inference_config => aidb.inference_config(
    system_prompt => You are a concise technical writer.,
    temperature   => 0.2,
    max_tokens     => 256
  )::json
);

```

完全なパラメーターのリファレンスについては、[Models reference](#) を参照してください。

11.2.5 quietb.decode_text_batch()

単一のバッチ要求で複数のプロンプトのテキスト応答を生成します。

```

SELECT aidb.decode_text_batch(
  model_name => my_model_name,
  input      => ARRAY[
    What is a vector database?,
    What is retrieval-augmented generation?
  ]
);

```

__OUTPUT__

decode_text_batch

```

-----
A vector database stores high-dimensional vectors ...
Retrieval-augmented generation (RAG) combines a ...
(2 rows)

```

完全なパラメーターのリファレンスについては、[Models reference](#) を参照してください。

11.2.6 quietb.inference_config()

推論構成オブジェクトをビルドして、言語モデル生成動作を制御します。結果を `aidb.decode_text()` または `aidb.decode_text_batch()` の `inference_config` 引数として、またはパイプラインステップ内の `aidb.summarize_text_config()` の一部として渡します。

```
SELECT aidb.decode_text(
  model_name      => my_llm,
  input           => Write a haiku about PostgreSQL.,
  inference_config => aidb.inference_config(
    temperature => 0.9,
    max_tokens  => 64,
    seed        => 42
  )::json
);
```

完全なパラメーターのリファレンスについては、[Models reference](#) を参照してください。

!!!注これらのファンクションをパイプライン内のステップとして使用するには、[Pipeline steps](#) を参照してください。

11.3 Data parsing

EDB Postgres AI は、HTML や PDF を含むさまざまなデータ形式からテキストを解析および抽出するビルトイン機能を提供します。これらの機能は、さまざまな形式の複雑さを処理しながら、後続のパイプラインステップで簡単に処理できるクリーンで構造化されたテキスト出力を提供するように設計されています。

11.3.1 HTML を解析する

`aidb.parse_html()` ファンクションは、HTML 文字列から可読テキストを抽出し、コンテンツの論理構造を保持しながらタグを削除します。

```
SELECT * FROM aidb.parse_html(
  html =>
    <h1>Hello, world!</h1>
    <p>This is my first web page.</p>
    <p>
      It contains some <strong>bold text</strong>, some <em>italic test</em>,
      and a <a href="https://google.com" target="_blank">link</a>.
    </p>

    <ol>
      <li>List item</li>
      <li>List item</li>
      <li>List item</li>
    </ol>
```

(次のページに続く)

OUTPUT

```
parse_html
```

```
Hello, world!
+
+
This is my first web page.
+
+
It contains some bold text, some italic test, and a link.
+
+
Postgres Logo Image
+
List item
+
List item
+
List item
```

(1 row)

- `method` は、HTML を解析する方法を決定します。
 - `StructuredPlaintext` デフォルトアルゴリズム的なアプローチを使用して、HTML から生のテキストを抽出します。単純なデータの取り込みに最適です。
 - `StructuredMarkdown` —HTML を Markdown のようなテキストに変換します。これにより、重要な構文コンテキストヘッダーやリストなどが保持され、LLM がドキュメントの元の階層をより深く理解するのに役立ちます。

`aidb.parse_pdf()` ファンクションは、バイナリ PDF データからテキストを抽出します。PDF は構造的に複雑であることが多いため、このファンクションは堅牢なエラー処理を提供します。

```
SELECT * FROM aidb.parse_pdf(
    bytes =>
decode(255044462d312e340a25b89a929d0a312030206f626a3c3c2f547970652f436174616c6f672f506167657320332030
hex),
    options => {"method": "Structured", "allow_partial_parsing": true} -- Default
);
```

--OUTPUT--

part_id	text
0	Hello World!+

(次のページに続く)

(1 row)

- `method` は、PDF を解析する方法を決定します。
 - `Structured` メソッドデフォルト標準の PDF 仕様に依存して、テキストブロックを識別します。PDF 仕様に依存する構造 PDF 解析は、特に複雑なまたは独自のエンコーディングとさまざまなフォントが含まれる場合、仕様に準拠していない PDF に苦戦する可能性があります。
 - `allow_partial_parsing` : 次の場合に PDF の解析を続行するかどうかをシステムに指示する安全フラグパーサーは、1 つ以上のページでエラーを検出しました。これを `true` デフォルトに設定すると、非準拠または複雑なファイルからできるだけ多くのデータを取得できます。
- ページマッピング 結果の `part_id` 列は、テキストが抽出された特定のページインデックスに直接マップし、テキストを維持できます。AI パイプライン内の元のドキュメント参照。

 Tip

Figure 1: The structure of the proposed model. The input is a sequence of tokens $x_1, x_2, \dots, x_n$ (e.g., "The cat sat on the mat"). These tokens are processed by an embedding layer to produce a sequence of vectors $z_1, z_2, \dots, z_n$. These vectors are then processed by a sequence of layers (e.g., LSTM or Transformer) to produce a sequence of hidden states $h_1, h_2, \dots, h_n$. The final hidden state h_n is used to generate the output y (e.g., "The cat sat on the mat").

11.4 Performing OCR

`aidb.perform_ocr()` フังก์ションを使用して、画像からテキストを抽出します。これは、生の画像バイトと登録された OCR モデル名を入力として取得し、`part_id` 列各テキストブロックを識別するおよび `text` 列抽出された文字列を含むテーブルを結果ます。このファンクションはネスト解除を使用するため、単一の画像から複数の行、テキストセグメントごとに 1 行が生成される場合があります。一部のプロバイダーにはフォーマット制限があります。たとえば、NVIDIA NIM プロバイダーは、現在 `png` および `jpeg` 形式のみをサポートしています。

ステップ 10CR 対応モデルを登録する

```
SELECT aadb.create_model(
    my_paddle_ocr_model,
    nim_paddle_ocr,
    credentials => {"api_key": "__NVIDIA_NIM_API_KEY__"}::JSONB
);
```

ステップ2 画像からテキストを抽出する

[illegible]

(次のページに続く)

(次のページに続く)

(前のページからの続き)

[illegible]

(次のページに続く)

(前のページからの続き)

```
__OUTPUT__
create_model
-----
my_paddle_ocr_model
(1 row)

part_id |      text
-----+-----
      0 | Tesseract sample
(1 row)
```

11.5 Text reranking

`aidb.rerank_text()` フังก์ションを使用して、クエリーとの関連性によって一連の候補テキストパッセージを並べ替えます。再ランキングは通常、取得後のステップとして適用され、候補の初期セットがナレッジベースから取得された後、最も関連性の高い結果を上部に表示します。

ステップ 1 リランキングモデルを登録する

```
SELECT aidb.create_model(
  my_reranker,
  nim_reranking,
  config => aidb.nim_reranking_config(
    api_key => nvapi-...,
    model   => nvidia/nv-rerankqa-mistral-4b-v3
  )
);
```

ステップ 2: 候補セットを再ランク付けする

```
SELECT * FROM aidb.rerank_text(
  model_name => my_reranker,
  query      => What is retrieval-augmented generation?,
  input      => ARRAY[
    RAG combines retrieval and generation to produce grounded answers.,
    Vector databases store high-dimensional embeddings for similarity search.,
    RAG systems retrieve relevant documents before generating a response.
  ]
) ORDER BY logit_score DESC;
```

```
__OUTPUT__

      text | logit_score | id
-----+-----+-----
RAG systems retrieve relevant documents before generating ... | 0.97 | 2
```

(次のページに続く)

(前のページからの続き)

```
RAG combines retrieval and generation to produce grounded ... | 0.91 | 0
Vector databases store high-dimensional embeddings for ... | 0.43 | 1
(3 rows)
```

id 列は、input 配列内の各文字列の 0 ベースのインデックスであるため、元のデータに結合して戻すことができます。

完全なパラメーターのリファレンスについては、[Models reference](#) を参照してください。

11.5.1 ナレッジベースの取得後に再ランキングを使用する

一般的なパターンは、ナレッジベースから初期候補セットを取得し、結果を再ランク付けすることです。

```
WITH candidates AS (
  SELECT key, value
  FROM aidb.retrieve_text(my_kb, What is RAG?, topk => 20)
)
SELECT c.key, c.value, r.logit_score
FROM candidates c
JOIN aidb.rerank_text(
  model_name => my_reranker,
  query      => What is RAG?,
  input      => ARRAY(SELECT value FROM candidates ORDER BY key)
) r ON r.text = c.value
ORDER BY r.logit_score DESC
LIMIT 5;
```

!!!注再ランキングモデルは、クエリに対してパッセージをスコアリングしますが、テキストは生成しません。リランキングモデルを登録するには、プロバイダーに適切な `xxx xxx xxx` を使用します。たとえば、NVIDIA NIM リランキングエンドポイントの場合は `aidb.nim_reranking_config()` です。

11.6 Text summarization

`aidb.summarize_text()` は、登録された言語モデルを使用して、単一のテキスト文字列を簡潔な概要に圧縮します。`aidb.summarize_text_aggregate()` は、SQL 集計フォームです。GROUP BY とともに使用して、複数の行にわたってテキストを要約し、グループごとに 1 つの要約を結果ます。

パラメーターテーブルと戻り型については、[Reference → Functions](#) を参照してください。

11.6.1 quietb.summarize_text()

ステップ 1 モデルを登録する

```
SELECT aidb.create_model(my_t5_model, t5_local);
```

ステップ 2 テキスト文字列を要約する

11.7 Text summarization

```

Control model behavior at runtime using `inference_config`:

```
``` sql SELECT * FROM promoteb.summarize_text( input => '長い記事テキスト...', options => promoteb.summarize_config( 'my_llm', NULL, ' Create a resume_config( 温度=> 0.3、 max_tokens => 50、 seed => 42 ) ):json );
```

11.8 Text summarization

- - 呼び出し時に失敗する `SELECT * FROM agentb.summarize_text( input => 'Hello world', options => promoteb.summarize_text_config(model => 'bert_model' ):json );`

OUTPUT ERROR 要求されたアダプターは、モデルプロバイダーでサポートされていません。bert_local ```

11.9 Text summarization

<i>Provider</i> 'bert_local'	Embedding-only provider used for summarization	Use <i>completions</i> , <i>openai_completions</i> , <i>t5_local</i> , or another language provider
<i>Invalid parameters:</i> 'desired_length' must be greater than 0	Invalid chunk config	Use a positive integer for <i>desired_length</i>
<i>Invalid parameters:</i> 'max_length' must be >= 'desired_length'	<i>max_length</i> is smaller than <i>desired_length</i>	Set <i>max_length</i> >= <i>desired_length</i>
<i>Invalid parameters:</i> 'overlap_length' must be less than 'desired_length'	Overlap too large	Reduce <i>overlap_length</i> (10–20% of <i>desired_length</i> is typical)
<i>Text summarization failed to reduce size</i>	<i>reduce</i> strategy produced output longer than input	Increase chunk size (use >= 50 words or >= 800 chars)

11.9.1 Summarization cookbook

基本的な要約を超えて `aidb.summarize_text_aggregate()` を使用するための現実世界のレシピ。各レシピは、データに適応できる自己完結型のパターンです。

すべての例では、テーブルを含む sales データセットを使用します

レシピ

1. カテゴリ別にまとめます

使用する場合任意のディメンション製品、顧客、地域、期間ごとにグループ化されたテキストの概要が必要です。

```
SELECT p.product_name,
       count(sn.note_id) AS note_count,
       aidb.summarize_text_aggregate(
         sn.note_text,
         aidb.summarize_text_config(my-llm)::json
       ) AS summary
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
GROUP BY p.product_name
ORDER BY note_count DESC;
```

- チャンクしない場合、各グループのすべてのテキストは単一の LLM 呼び出しとして送信されます。
- 500 ノート各平均〜250 文字未満のグループの場合、チャンクなしが最速です。
- 大規模なグループの場合、チャンクを追加してコンテキストウィンドウのオーバーフローを回避します Recipe 7 を参照してください。

2. ディール損失分析

使用する場合取引が失われている理由を、実行可能な理由とともに理解したいと考えています。

```
SELECT p.product_name,
       aidb.summarize_text_aggregate(
         sn.note_text,
         aidb.summarize_text_config(
           my-llm,
           prompt => Analyze these sales notes and answer: Why are we losing
           ↳ deals?
           What are the top reasons for failure?
           Provide specific, actionable insights.
         )::json
       ) AS loss_analysis
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
WHERE so.status IN (lost, closed_lost)
GROUP BY p.product_name;
```

- LLM が失敗パターンに焦点を当てるように、失われた取引のみ WHERE status IN ('lost', 'closed_lost') にフィルターします。

- プロンプトは具体的である必要があります。「top resume」は、「analyze」よりも良い結果をもたらします。
- より多くの制約された出力については、[Recipe 8](#) を参照してください。

3. 感情分析

使用する場合グループ化されたテキストから全体的な感情ポジティブ/ネガティブ/混合を測定したい。

```
-- Compare sentiment between won and lost deals
SELECT p.product_name,
       so.status AS deal_outcome,
       aidb.summarize_text_aggregate(
         sn.note_text,
         aidb.summarize_text_config(
           my-llm,
           prompt => Analyze the overall sentiment of these entries. Provide:
                     1) Overall sentiment (positive/negative/neutral/mixed),
                     2) Key positive themes,
                     3) Key negative themes,
                     4) Notable concerns if any.
         )::json
       ) AS sentiment
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
WHERE so.status IN (won, lost)
GROUP BY p.product_name, so.status
ORDER BY p.product_name, so.status;
```

- 獲得した感情と失われた感情を並べて比較すると、成功した取引の違いが明らかになります。
- プロンプトの番号付きフォーマットは、構造化された出力をガイドしますが、LLM の遵守は規模で異なります。❏❏❏ ❏❏ ❏❏ を参照してください。

4. 重要なインサイトの抽出

使用する場合多くのエントリにわたって最も重要な所見の簡潔なリストが必要です。

```
SELECT p.product_name,
       aidb.summarize_text_aggregate(
         sn.note_text,
         aidb.summarize_text_config(
           my-llm,
           prompt => Extract the top 5 key insights from these entries as a
→numbered list.
                     Focus on actionable findings, patterns, and important trends.
         )::json
```

(次のページに続く)

```

    ) AS insights
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
GROUP BY p.product_name
ORDER BY count(sn.note_id) DESC
LIMIT 5;

```

- プロンプトの「トップN」と「番号付きリスト」は、出力長をコンストレインします。
- 中程度のグループサイズ 50～500 ノートで最適に動作します。非常に大規模なグループでは、LLM が特定の洞察ではなく一般的な概要を生成する場合があります。

5. 問題の分類

使用する場合問題またはテーマを事前定義されたバケットに分類する必要があります。

```

SELECT p.product_name,
       aidb.summarize_text_aggregate(
         sn.note_text,
         aidb.summarize_text_config(
           my-llm,
           prompt => Categorize these sales notes into the following categories:
                       Pricing Concern, Technical Issue, Competitive Loss,
                       Timing/Budget, Feature Gap, Relationship/Trust, Integration.
→Challenge.
                       For each category provide the count and percentage of
→entries that match,
                       plus a brief description.
         )::json
       ) AS categories
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
WHERE so.status IN (lost, closed_lost)
GROUP BY p.product_name;

```

- プロンプトでカテゴリを明示的に定義します。LLM に選択させないでください。
- カウント/パーセンテージを要求して、定量的な出力を取得します。
- LLM は、要求されたフォーマットに完全に従わない場合があります。より信頼性の高いフォーマットするには、ラッパー関数 [Recipe 8](#) を使用します。


```
JOIN products p ON so.product_id = p.product_id
GROUP BY p.product_name;
```

仕組みテキストは、20 のワードがオーバーラップする 200 ワードのチャンクに分割されます。各チャンクは独立して要約されます。概要が連結されます。出力は約 5,000~15,000 文字冗長、詳細です。

- 大きなチャンク 200 ワード = 少ない LLM 呼び出し = 高速ですが、粒度が低くなります。
- 小さなチャンク 50 ワード = より多くの LLM 呼び出し = 遅くなりますが、より詳細です。
- 20 ワードのオーバーラップにより、チャンク境界でコンテキストが失われません。

8. 再利用可能な構成ラッパーファンクション

使用する場合制約された出力形式での一貫した反復可能な分析が必要です。

事前構築された構成を返す SQL ファンクションを作成し、`summarize_text_aggregate()` に直接渡します。このパターンは、JSON を返すファンクション内のプロンプトロジックをラップするため、長いプロンプト文字列を繰り返すことなく GROUP BY でクリーンに使用できます。

焦点を絞った分析

```
CREATE OR REPLACE FUNCTION analyze_top_reasons_config(
  p_question TEXT,
  p_n         INTEGER DEFAULT 3,
  p_model     TEXT      DEFAULT my-llm
) RETURNS JSON AS $$
  SELECT aidb.summarize_text_config(
    p_model,
    prompt => format(
      Answer this question about the text: %s
      Give exactly %s reasons, each as a single sentence.
      No preamble, no extra commentary.,
      p_question, p_n
    )
  )::json;
$$ LANGUAGE sql;

-- Usage: pass the config function directly to the aggregate
SELECT product_name,
       aidb.summarize_text_aggregate(
         note_text,
         analyze_top_reasons_config(Why are customers churning?, 3)
       ) AS churn_reasons
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
```

(次のページに続く)

(前のページからの続き)

```
WHERE so.status = lost
GROUP BY product_name;
```

構造化された感情

```
CREATE OR REPLACE FUNCTION analyze_sentiment_config(
  p_model TEXT DEFAULT my-llm
) RETURNS JSON AS $$
  SELECT aidb.summarize_text_config(
    p_model,
    prompt => Analyze sentiment. Respond ONLY in this format:
OVERALL: [positive/negative/neutral/mixed]
SCORE: [1-10]
TOP_POSITIVE: [one sentence]
TOP_NEGATIVE: [one sentence]
TREND: [improving/declining/stable]
  )::json;
$$ LANGUAGE sql;

-- Usage
SELECT product_name,
       aidb.summarize_text_aggregate(note_text, analyze_sentiment_config()) AS
→sentiment
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
GROUP BY product_name;
```

問題カテゴライザー

```
CREATE OR REPLACE FUNCTION categorize_issues_config(
  p_categories TEXT[] DEFAULT ARRAY[
    pricing, technical, competition,
    timing, budget, feature_gap, other],
  p_model TEXT DEFAULT my-llm
) RETURNS JSON AS $$
  SELECT aidb.summarize_text_config(
    p_model,
    prompt => format(
      Categorize the issues in this text into ONLY these categories: %s.
      For each category, give a count and one representative example.
      Format: CATEGORY: count - "example quote",
      array_to_string(p_categories, , )
    )
  )::json;
```

(次のページに続く)

```

$$ LANGUAGE sql;

-- Usage
SELECT product_name,
       aidb.summarize_text_aggregate(note_text, categorize_issues_config()) AS
→categories
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
WHERE so.status IN (lost, closed_lost)
GROUP BY product_name;

```

- 構成ラッパー機能は、一貫したインターフェイスを提供し、クエリを変更せずにプロンプトを調整できます。
- 戻り値の型は `summarize_text_aggregate` のパラメーター型と一致するように、JSONB ではなく JSON である必要があります。
- 「プリアンブルなし」、「のみ」、「このフォーマットで応答」は効果的なステアリングフレーズです。

9. 要約前のテキストのスクラブ

使用する場合テキストデータには、LLM トークンを無駄にするフィラーワード、挨拶、署名、またはボーラプレートが含まれています。

クリーンアップファンクションを作成し、集計の前に適用します。

```

CREATE OR REPLACE FUNCTION clean_note_text(p_text TEXT)
RETURNS TEXT AS $$
BEGIN
    RETURN regexp_replace(
        regexp_replace(
            regexp_replace(
                regexp_replace(p_text,
                    -- Remove email signatures
                    E(Best regards|Sincerely|Thanks|Cheers), ?\s * \n. * $,
                    , gi),
                -- Remove greeting lines
                E^(Hi|Hello|Dear|Hey)\s+[A-Za-z]+, ?\s * \n,
                , gi),
                -- Remove filler phrases
                E\b(basically|essentially|actually|literally|you know|I mean|sort of|kind
→of)\b,
                , gi),
            -- Collapse whitespace
            E\s+, , g);
END;

```

(前のページからの続き)

```

$$ LANGUAGE plpgsql IMMUTABLE;

-- Use it:
SELECT product_name,
       aidb.summarize_text_aggregate(
           clean_note_text(note_text),
           aidb.summarize_text_config(my-llm)::json
       ) AS summary
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
GROUP BY product_name;

```

- IMMUTABLE PostgreSQL は、繰り返しの呼び出しを最適化します。
- データの正規表現パターンたとえば、内部専門用語、標準ヘッダーなどをカスタマイズします。
- 一般的な削減トークンが 15～30%減少します。これは、チャンクが減り、処理が高速になることを意味します。

レシピの組み合わせ 完全な最適化スタック

最大の速度と品質を得るには、スクラビング+フィルタリング+構成ラッパーを組み合わせます。

```

SELECT p.product_name,
       aidb.summarize_text_aggregate(
           clean_note_text(sn.note_text),
           analyze_top_reasons_config(Why are we losing these deals?, 3)
       ) AS loss_reasons
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
WHERE so.status IN (lost, closed_lost)
GROUP BY p.product_name;

```

このパイプライン 関連する行 失われた取引にフィルタリングします。clean_note_text() を介して行ごとにフィルターテキストをストリップしますトークンが 15～30%減少します。クリーン化されたすべてのテキストを蓄積し、LLM に送信します。構成ラッパーを介して制約されたプロンプトを適用して、短く焦点を当てた出力を実現します。

パフォーマンスベンチマーク

テストスイートからのすべてのデータは、AIDB 6.1.0 で Ollama を介して granite4:1b を使用して、sales データセット 18,503 ノート、平均 266 文字/ノートに対して実行されます。

チャンクなしのパフォーマンス

チャンキングがない場合、グループ内のすべてのテキストは単一の LLM 呼び出しとして送信されます。高速ですが、モデルのコンテキストウィンドウによって制限されます。

Group size	Duration	Result size	Notes
77–84 notes	21–28s	1,100–2,500 chars	安全な範囲、完全なデータ処理
100 notes	28s	2,200 chars	約 26K の文字入力、十分なコンテキスト内
2,915 notes	98s	3,200 chars	131K トークンを超えるコンテキストウィンドウ以下を参照してください
3,248 notes	91s	1,700 chars	部分的なデータのみが処理されました

チャンク戦略の比較 50 ノート

Config	Est. LLM calls	Duration	ms/call	Result chars	Status
No chunking	1	12s	11,947	1,443	PASS
words/50/append	32	152s	4,759	12,159	PASS
words/100/append	16	77s	4,832	7,021	PASS
words/200/append	8	48s	6,044	4,781	PASS
words/50/reduce rf=2	63	217s	3,452	545	PASS
words/100/reduce rf=2	31	102s	3,280	475	PASS
words/200/reduce rf=2	15	51s	3,367	767	PASS
words/50/reduce rf=3	50	170s	3,396	410	PASS
words/100/reduce rf=3	25	72s	2,897	867	PASS
words/200/reduce rf=3	12	44s	3,661	535	PASS
chars/200/append	40	246s	6,154	16,355	PASS
chars/400/append	20	137s	6,835	11,793	PASS
chars/800/append	10	66s	6,584	5,897	PASS
chars/200/reduce rf=2	81	971s	11,988	—	FAIL
chars/400/reduce rf=2	41	368s	8,967	441	PASS
chars/800/reduce rf=2	21	144s	6,842	850	PASS

戦略効率の概要

Strategy	Avg LLM calls	Avg total time	Avg ms/call	Avg output
No chunking	1	12s	11,947	1,443 chars
Append	21	121s	5,868	9,668 chars
Reduce rf=2	34	176s	5,182	616 chars
Reduce rf=3	29	95s	3,318	604 chars

予想される規模

Scenario	100 notes	500 notes	1,000 notes
No chunking	28s (1 call)	~60s (1 call)	~100s (1 call, partial data*)
words/100/append	~77s (16 calls)	~385s (80 calls)	~770s (160 calls)
words/100/reduce rf=3	~72s (25 calls)	~180s (62 calls)	~240s (125 calls)

- 1,000 以上のノートでのノーチャンクは、コンテキストウィンドウを超える場合があります。完全を期するためにチャンクを使用します。

推奨される構成

Use case	Config	Why
Fastest possible (< 500 notes)	No chunking	Single LLM call, stays within context window
Fast + detailed	words/200/append	Fewest calls for append, good detail
Balanced detail	words/100/append	More granular, moderate speed
Executive summary	words/100/reduce rf=3	Compressed output, efficient reduce
Maximum compression	words/50/reduce rf=3	Smallest output, most LLM calls

一般的なガイダンスチャンクなしで開始します。グループが~500 ノートを超える場合にのみチャンクを追加します。常に、文字ベースよりもワードベースのチャンクを優先します。Reduce の場合、rf=2 よりも rf=3 を優先します。同様の出力品質で呼び出しの合計数が少なくなります。

既知の制限

コンテキストウィンドウのオーバーフローサイレント

入力テキストの合計がモデルのコンテキストウィンドウを超えると、テキストはサイレントに切り捨てられます。エラーは発生しません。LLM は、適合するものを処理し、残りを無視します。

例えば granite4:1b には、131,072 トークンのコンテキストウィンドウがあります。3,248 ノート 806K 文字、~201K トークンのグループは、これを 53%超えます。結果は有効な概要ですが、データの一部にすぎません。

安全な制限チャンキングなし

Avg note size	Safe note count	Total tokens
250 chars	~500 notes	~33K tokens
500 chars	~250 notes	~33K tokens
1,000 chars	~125 notes	~33K tokens

回避策これらの制限を超える可能性のあるグループにはチャンクを使用します。チャンク化は、コンテキストウィンドウのサイズに関係なく、すべてのデータを処理します。

```
- - Safe for large groups:
aidb.summarize_text_config(
    my-llm,
    aidb.chunk_text_config(200, 200, 20, words),
    NULL,
    append
)::json
```

小さなチャンクでの障害の削減

reduce ストラテジーでは、各要約ラウンドが入力よりも短い出力を生成する必要があります。チャンクサイズが小さい場合、LLM は入力よりも長い概要を生成することが多く、次の原因が発生します。

Text summarization failed to reduce size. The summarized output **is** larger than **or**
 ↳ equal to the **input**.
 Consider adjusting the chunking parameters.

障害しきい値

Chunk type	Size	Result
chars/200/reduce	200 chars	Always fails
chars/400/reduce	400 chars	Unreliable
chars/800/reduce	800 chars	Reliable
words/50/reduce	50 words	Reliable
words/100/reduce	100 words	Reliable
words/200/reduce	200 words	Reliable

回避策 **reduce** でワードベースのチャンクを使用しますテストされたすべてのサイズで 100 の合格率。char ベースを使用する必要がある場合は、**>= 800** chars を使用します。

温度、**max_tokens**、およびシステムプロンプトの制御

デフォルトでは、LLM はプロバイダーのデフォルト温度通常 0.7~1.0 を使用します。これにより、実行ごとに出力に大きな分散が発生する可能性があります。**inference_config** を使用して、コールごとにこれらのパラメーターを制御します。

```

- - Control temperature and max_tokens for more consistent, bounded output
SELECT p.product_name,
       aidb.summarize_text_aggregate(
         sn.note_text,
         aidb.summarize_text_config(
           my-llm,
           NULL,
           Summarize the key points,
           NULL,
           NULL,
           aidb.inference_config(
             temperature => 0.3,
             max_tokens  => 500
           )
         )::json
       ) AS summary
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
GROUP BY p.product_name;

- - Override the system prompt for non-summarization tasks
SELECT aidb.summarize_text(
  input  => Customer complained about slow response times and billing errors.,
  options => aidb.summarize_text_config(
    my-llm,
    NULL,
    NULL,
    NULL,
    NULL,
    aidb.inference_config(
      system_prompt => You are a customer support analyst. Classify issues by
→severity.,
      temperature  => 0.2
    )
  )::json
);

```

!!! ヒント 最良の結果を得るには、`inference_config` と制約されたプロンプトを組み合わせます。たとえば、`max_tokens => 200` を設定し、`'Give exactly 3 bullet points, max 1 sentence each.'` のようなプロンプトを使用します。トークン制限はハードキャップとして機能し、プロンプトがフォーマットをガイドします。

エラーなしの空の結果

場合によっては、LLM は処理後に空の文字列を返す場合があります。AIDB は、これをエラーなしの成功結果として保存します。アプリケーションコードで `length(result)` を常に確認してください。

```
SELECT product_name,
       CASE
         WHEN length(summary) = 0 THEN ERROR: Empty result
         WHEN length(summary) < 50 THEN WARNING: Suspiciously short
         ELSE summary
       END AS validated_summary
FROM (
  SELECT p.product_name,
         aidb.summarize_text_aggregate(
           sn.note_text,
           aidb.summarize_text_config(
             my-llm,
             prompt => Your prompt here
          )::json
         ) AS summary
  FROM sales_notes sn
  JOIN sales_orders so ON sn.order_id = so.order_id
  JOIN products p ON so.product_id = p.product_id
  GROUP BY p.product_name
) sub;
```

プロンプトアドヒアランスは大規模に低下する

大量のテキスト 1,000 以上のノートがある場合、LLM はカスタムプロンプトに関係なく、要約動作がデフォルトになる傾向があります。構造化された出力要求番号付きリスト、カテゴリフォーマット、センチメントラベルは、フォローの信頼性が低くなります。

回避策 **LIMIT** 以上の積極的なフィルタを使用して入力ボリュームを減らします。 **append** でチャンクを使用するため、各チャンクはより小さな入力に対して独立してプロンプトを受信します。非常に明示的なフォーマット制約を指定したラッパー関数を使用します。要約する前にスクラブテキスト Recipe 9 を使用してノイズを削減します。

最適化パターン

1. テキストのスクラブ

要約する前に、つなぎ言葉、挨拶、および定型文を削除します。Recipe 9 を参照してください。影響トークンが 15~30%削減され、それに比例して処理が高速化します。

2. コンストレインされたプロンプト

特定の短いフォーマットの出力を生成するように LLM に指示すると、トークン生成時間各呼び出しの最も遅い部分が削減されます。

```
- - Slow (open-ended, LLM may generate 2,000+ chars):
Summarize these notes.

- - Fast (constrained, LLM generates ~200 chars):
Give exactly 3 bullet points, max 1 sentence each.
```

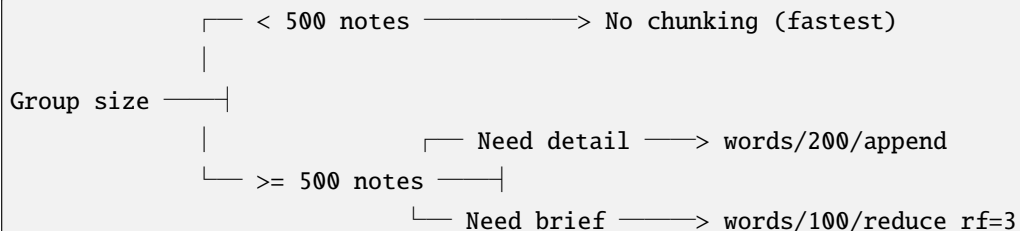
3. 抽出プレフィルタリング

非常に大規模なデータセットの場合、LLM に送信する前に SQL を使用して最も関連性の高いテキストにプレフィルタリングします。

```
- - Only send notes containing relevant keywords
SELECT aidb.summarize_text_aggregate(
  note_text,
  aidb.summarize_text_config(
    my-llm,
    prompt => What are the pricing concerns?
  )::json
)
FROM sales_notes
WHERE note_text ILIKE %price%
      OR note_text ILIKE %cost%
      OR note_text ILIKE %budget%
      OR note_text ILIKE %expensive%
GROUP BY product_id;
```

インパクト 処理する行が大幅に減少。LLM は関連するテキストのみを認識するため、速度と品質の両方が向上します。

4. 適切な戦略の選択



5. パフォーマンスコストモデル

支配的なコストは LLM 呼び出し数です。入力サイズに応じて、各呼び出しには 3～12 秒かかります。

Step	Time
Network overhead	~50ms
Prompt processing	~1-3s
Token generation	~2-8s
Response parsing	~10ms
Total per call	~3-12s

プロンプト処理は入力長に応じてスケーリングしますが、トークン生成は出力長に応じてスケーリングします。サマリーは入力量に関係なくほぼ固定サイズであるため、コンテキストウィンドウの制限までは、多数の小さな呼び出しよりも、より少なくて大きな呼び出しの方が効率的です。これが、ノーチャックが小規模なグループに非常に効果的な理由です。呼び出しごとのテキストを最大化し、呼び出し数を最小限に抑えます。集計は処理前にすべての行を累計しているため、テキストは自然にバッチ化されます。

11.9.2 Summarization use cases

`aidb.summarize_text_aggregate()` は、テキストの要約をネイティブ SQL 集計に変換します。すべてのアナリストが既に知っているのと同じ `GROUP BY` セマンティクス。このページでは、ビジネス価値とエンタープライズユースケースについて説明します。

問題

企業は、PostgreSQL 内に大量の非構造化テキスト—顧客のフィードバック、サポートチケット、セールスノート、コンプライアンスログ、臨床ナラティブなどを保存しています。このデータから洞察を抽出するには、従来、脆弱なパイプラインが必要です。データベースから抽出、変換およびバッチ、外部 LLM API に送信し、応答を解析し、結果をロードして戻します。すべての手順で、遅延、障害モード、セキュリティ露出、およびエンジニアリングコストが発生します。

SQL を知っているアナリストは完全にロックアウトされています。「北東部の顧客は何について不満を持っていますか？」のような質問に答えるだけでもデータエンジニアと ML エンジニアが必要です。

AIDB の違い

AIDB は、データを LLM に移動するのではなく、LLM をデータにもたらしめます。要約はネイティブ SQL 集計になります。

```
SELECT region,
       aidb.summarize_text_aggregate(
         customer_feedback,
         aidb.summarize_text_config(gpt-4o)::json
       ) AS feedback_summary
FROM support_tickets
```

(次のページに続く)

(前のページからの続き)

```
WHERE created_at > NOW() - INTERVAL 7 days
GROUP BY region;
```

1つのクエリー。パイプラインはありません。パイソンはありません。外部オーケストレーションはありません。各行には、そのリージョンのすべてのフィードバックの LLM 生成の概要が含まれており、SQL クライアントに直接返されます。

同じデータ、別の質問、別のプロンプト

```
SELECT region,
       aidb.summarize_text_aggregate(
         customer_feedback,
         aidb.summarize_text_config(
           gpt-4o,
           prompt => Identify the top 3 product issues and suggest fixes
        )::json
       ) AS issues
FROM support_tickets
WHERE created_at > NOW() - INTERVAL 7 days
GROUP BY region;
```

同じ機能。文字列を 1 本変更しました。全く新しい分析 — コードの展開、パイプラインの変更はありません。

ビジネス価値

データはデータベースから離れることはありません 規制対象業界ヘルスケア、金融、政府などでは、機密データを外部 API に移動するには、数か月のコンプライアンスレビューが必要になる場合があります。AIDB は、すべてのデータを PostgreSQL 内に保持します LLM エンドポイントは、オンプレミス、プライベート VPC 内、またはローカルで実行できます。

SQL はユニバーサルインターフェイス — 世界中には約 1,000 万人の SQL リテラシーの専門家がいます。AIDB は、すべてのアナリスト、DBA、および BI 開発者を、新しいツール、ノートブック、または API キーを管理せずに AI 搭載アナリストに変えます。

インフラストラクチャのカテゴリ全体を排除します — Airflow DAG、Lambda ファンクション、SQS キュー、または Kubernetes ジョブはありません。データベースはコンピューティングレイヤーです。

リアルタイム、アドホックインテリジェンス — 従来の NLP パイプラインはバッチ指向です。AIDB は、ライブデータに対するアドホッククエリーを有効にします。エグゼクティブは、午後 3 時に新しい質問をし、午後 3 時 1 分までに回答を得ることができます。

PostgreSQL でスケール — PostgreSQL の成熟したクエリプランナー、パラレルワーカー、接続プーリング、およびロールベースのアクセス制御を活用します。すでに PostgreSQL インフラストラクチャに投資している企業は、限界インフラコストゼロで AI 機能を入手できます。

エンタープライズユースケース

小売と e コマース — 何百万ものレビューとフィードバックエントリにわたる GROUP BY 製品カテゴリ。好み、苦情、および新しいトレンドのカテゴリごとの概要を取得します。新しいレビューが到着すると、リアルタイムで更新されます。

金融サービス — 融資担当者ノート、借り手とのコミュニケーション、および引受コメントにわたる GROUP BY リスクセグメント。量的モデルが見逃す表面パターン—デフォルトの前にあるフリーテキストフィールドのナラティブシグナル。

ヘルスケアとライフサイエンス — GROUP BY 患者のフィードバック、臨床ノート、および退院概要にわたる GROUP BY 施設、部門、または状態。PHI を外部サービスに公開せずに、ケアの質の変動を特定します。

法とコンプライアンス — GROUP BY 数千のコンプライアンス文書、監査結果、および規制対応にわたる GROUP BY 管轄または規制タイプ。数週間の手動レビューを単一のクエリーに削減します。

営業とカスタマーサクセス — GROUP BY CRM ノート、通話のトランスクリプト、およびメールの概要全体の営業担当者、地域、または取引ステージ。すべての CRM の非構造化テキストから勝敗パターンとコーチングの機会を抽出します。

製造とサプライチェーン — GROUP BY 調達通信、品質レポート、および出荷ノートにわたる GROUP BY サプライヤーまたはコンポーネント。配信メトリックに表示される前に、ナラティブシグナルからサプライチェーンリスクを検出します。

HR とタレント — GROUP BY 退職インタビューの記録と従業員アンケートの回答にわたる部門、在職期間、またはロールレベル。人々が去った本当の理由を表面化します。

その仕組み

AIDB は、データベースのネイティブ状態蓄積フレームワークを使用して、真の PostgreSQL 集計を実装します。

- **State ファンクション** — グループ内の各行からテキストを蓄積し、メモリ内のグループごとの状態を維持します
- **ファイナライズファンクション** - 蓄積されたテキストを構成済みの LLM に送信し、結果を返します
- **チャンキングエンジン** - 大きなテキストグループを管理可能なチャンクに自動的に分割します。ワード数または文字数で構成可能、状況保存のためのオーバーラップあり
- **削減戦略** — `append` はチャンクサマリーを連結します。`reduce` は、出力が収束するまで繰り返し再要約します。データベース内で実行される真のマップ-リデュースパターン
- **カスタムプロンプト** — 各クエリーは独自の指示を指定できます。要約、翻訳、センチメントの抽出、分類、トレンドの特定—同じ機能を無限に柔軟にする
- **モデルに依存しない** — クエリーごとに構成可能な OpenAI 互換エンドポイント GPT-4o、Llama、Mistral、オンプレミスモデルで動作します

すごい瞬間

顧客設定データのテーブルから始めます - 20 の顧客と 5 つの製品カテゴリにわたる 126 行。生データは、非構造化テキストの壁です。

次のクエリーを実行します。

```
SELECT name,
       aidb.summarize_text_aggregate(
         preferences,
         aidb.summarize_text_config(gpt-4o)::json
       ) AS style_profile
FROM customer_preferences
GROUP BY name;
```

数秒で、126 行は 20 行になります。それぞれは、その顧客のすべてのデータポイントから合成されたリッチで一貫したスタイルのプロファイルです。パイプラインはありません。エンジニアリングチームはありません。データベースからデータは残りませんでした。

次に、プロンプトを変更します。

```
SELECT category,
       aidb.summarize_text_aggregate(
         preferences,
         aidb.summarize_text_config(
           gpt-4o,
           prompt => Identify the 3 strongest purchasing signals and recommend
           →inventory actions
         )::json
       ) AS buying_signals
FROM customer_preferences
GROUP BY category;
```

同じデータ。同じ機能。別の質問。異なる洞察力。アナリストは文字列を 1 つ変更し、まったく新しい分析を取得しました。コードの展開、パイプラインの変更、チケットのファイルはありません。

それは、カチッとした瞬間です。アナリストが既に知っているツールを使用して、組織全体の非構造化データが自然言語で照会可能になります。

チャンクとカスタムプロンプトを使用した例

大規模なテキストグループの場合、チャンクとターゲットを絞ったプロンプトを組み合わせます。

```
SELECT product_category,
       aidb.summarize_text_aggregate(
         review_text,
         aidb.summarize_text_config(
           local-llama,
           aidb.chunk_text_config(120, 120, 15, words),

```

(次のページに続く)

(前のページからの続き)

```
        Identify the top 3 quality issues and suggest improvements,
        reduce
    )::json
) AS quality_analysis
FROM product_reviews
GROUP BY product_category;
```

競争力のあるポジション

Capability	Traditional ETL + LLM API	AIDB in-database aggregate
Data movement	Required (security risk)	None (data stays in place)
User interface	Python / custom app	SQL (universal)
Infrastructure	Airflow + queues + compute	PostgreSQL (existing)
Time to first insight	Days to weeks	Minutes
Ad-hoc queries	Requires new pipeline	Change the SQL
Access control	Separate system	PostgreSQL RBAC / RLS
Audit trail	Must be built	PostgreSQL native logging
Compliance	Requires separate review	Inherits DB compliance posture
Model flexibility	Locked to provider	Any OpenAI-compatible endpoint