



**AIDB**

**リリース 7.3.2**

**EnterpriseDB Corporation**

**2026 年 04 月 30 日**



# 目次

|              |                                        |           |
|--------------|----------------------------------------|-----------|
| <b>第 1 章</b> | <b>AIDB</b>                            | <b>3</b>  |
| 1.1          | どこから始めるべきか                             | 3         |
| 1.2          | ドキュメントマップ                              | 3         |
| <b>第 2 章</b> | <b>AIDB release notes</b>              | <b>5</b>  |
| 2.1          | AIDB 7.x release notes                 | 5         |
| 2.1.1        | AIDB 7.2.0 release notes               | 6         |
| 2.1.2        | AIDB 7.3.0 release notes               | 7         |
| 2.1.3        | AIDB 7.3.2 release notes               | 9         |
| <b>第 3 章</b> | <b>AIDB overview</b>                   | <b>11</b> |
| 3.1          | AIDB の役割                               | 11        |
| 3.2          | スタンドアロンの拡張機能としての AIDB                  | 11        |
| 3.3          | 主要な機能                                  | 12        |
| 3.3.1        | Known issues                           | 12        |
| <b>第 4 章</b> | <b>Compatibility</b>                   | <b>13</b> |
| 4.1          | AIDB                                   | 13        |
| 4.1.1        | サポートされているプラットフォーム                      | 13        |
| 4.1.2        | サポートされている Postgres バージョン               | 13        |
| 4.2          | PGFS                                   | 13        |
| <b>第 5 章</b> | <b>Installing and configuring AIDB</b> | <b>15</b> |
| 5.1          | 次のステップ                                 | 15        |
| 5.1.1        | Installing AIDB                        | 15        |
| 5.1.2        | Configuring AIDB                       | 17        |
| <b>第 6 章</b> | <b>Integrating models</b>              | <b>19</b> |
| 6.1          | Local models                           | 19        |
| 6.1.1        | ビルトインモデル                               | 19        |
| 6.1.2        | カスタムローカルモデル                            | 20        |
| 6.1.3        | 登録されたモデルを使用する                          | 23        |
| 6.2          | Support matrix                         | 23        |
| 6.2.1        | ローカル組み込みモデル                            | 23        |
| 6.2.2        | NIM モデル                                | 25        |
| 6.3          | External model connections             | 27        |
| 6.3.1        | OpenAI 互換エンドポイント                       | 27        |
| 6.3.2        | NVIDIA NIM                             | 29        |
| 6.3.3        | Google Gemini                          | 32        |
| 6.3.4        | OpenRouter                             | 32        |

|              |                                      |           |
|--------------|--------------------------------------|-----------|
| <b>第 7 章</b> | <b>Configuring external storage</b>  | <b>35</b> |
| 7.1          | その仕組み                                | 35        |
| 7.2          | ステップ 1 保存場所を作成する                     | 35        |
| 7.2.1        | S3 互換オブジェクトストア                       | 35        |
| 7.2.2        | ローカルファイルシステム                         | 36        |
| 7.3          | ステップ 2 ボリュームを作成する                    | 36        |
| 7.3.1        | 例 S3 の PDF                           | 37        |
| 7.3.2        | 例ローカルディレクトリ内の画像                      | 37        |
| 7.4          | 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        | 37        |
| 7.5          | ボリュームの管理                             | 38        |
| 7.5.1        | リストと削除                               | 38        |
| 7.5.2        | ボリュームの内容を検査する                        | 38        |
| <b>第 8 章</b> | <b>AI pipelines</b>                  | <b>39</b> |
| 8.1          | Pipelines overview                   | 39        |
| 8.1.1        | 統合されたデータのライフサイクル                     | 39        |
| 8.1.2        | パイプラインアーキテクチャ                        | 40        |
| 8.1.3        | サポートされているモデル                         | 40        |
| 8.2          | Creating pipelines                   | 40        |
| 8.2.1        | テーブルソース                              | 41        |
| 8.2.2        | ボリュームソース PGFS                        | 41        |
| 8.3          | Pipeline steps                       | 42        |
| 8.3.1        | ChunkText                            | 43        |
| 8.3.2        | ParseHtml                            | 43        |
| 8.3.3        | ParsePdf                             | 44        |
| 8.3.4        | PerformOcr                           | 45        |
| 8.3.5        | SummarizeText                        | 46        |
| 8.3.6        | ナレッジベース                              | 47        |
| 8.4          | Orchestration                        | 48        |
| 8.4.1        | Auto-processing                      | 49        |
| 8.4.2        | Background workers                   | 50        |
| 8.4.3        | Observability                        | 53        |
| 8.4.4        | Pipeline error log                   | 54        |
| 8.5          | Pipelines API reference              | 57        |
| 8.5.1        | タイプ                                  | 57        |
| 8.5.2        | ドメイン                                 | 60        |
| 8.5.3        | ビュー                                  | 61        |
| 8.5.4        | ファンクション                              | 63        |
| 8.5.5        | パイプラインステップ構成ヘルパー                     | 66        |
| 8.5.6        | Vector インデックス構成ヘルパー                  | 70        |
| 8.5.7        | モデル構成ヘルパー                            | 73        |
| 8.6          | Examples                             | 73        |
| 8.6.1        | パイプラインの作成テーブルソース                     | 73        |
| 8.6.2        | パイプラインの作成 ボリュームソース                   | 74        |
| 8.6.3        | パイプラインの実行と更新                         | 75        |
| 8.6.4        | モニタリングと削除                            | 75        |
| 8.6.5        | 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 | 76        |

|                                     |            |
|-------------------------------------|------------|
| <b>第 9 章 Knowledge bases</b>        | <b>79</b>  |
| 9.1 取得ファンクション                       | 79         |
| 9.2 検索ファンクションの流れ                    | 79         |
| 9.2.1 promoteb.retrieve_text()      | 79         |
| 9.2.2 promoteb.retrieve_key()       | 80         |
| 9.3 高度なクエリー 中間ステップの結合               | 80         |
| 9.3.1 構文の例                          | 80         |
| <b>第 10 章 SQL functions</b>         | <b>101</b> |
| 10.1 Text chunking                  | 101        |
| 10.1.1 構成オプション                      | 102        |
| 10.2 Data parsing                   | 103        |
| 10.2.1 HTML を解析する                   | 103        |
| 10.2.2 PDF を解析する                    | 104        |
| 10.3 Performing OCR                 | 105        |
| 10.4 Text summarization             | 107        |
| 10.4.1 quietb.summarize_text()      | 107        |
| 10.5 Text summarization             | 108        |
| 10.6 Text summarization             | 108        |
| 10.6.1 Summarization use cases      | 108        |
| 10.6.2 Summarization cookbook       | 112        |
| 10.7 AI inference functions         | 127        |
| 10.7.1 quietb.encode_text()         | 127        |
| 10.7.2 promoteb.encode_text_batch() | 127        |
| 10.7.3 quietb.encode_image()        | 128        |
| 10.7.4 quietb.decode_text()         | 128        |
| 10.7.5 quietb.decode_text_batch()   | 129        |
| 10.7.6 quietb.inference_config()    | 130        |
| 10.8 Text reranking                 | 130        |
| 10.8.1 ナレッジベースの取得後に再ランキングを使用する      | 131        |
| <b>第 11 章 Reference</b>             | <b>133</b> |
| 11.1 Models reference               | 133        |
| 11.1.1 カタログビュー                      | 133        |
| 11.1.2 モデル管理機能                      | 134        |
| 11.1.3 HCP モデルファンクション               | 137        |
| 11.1.4 推論ファンクション                    | 138        |
| 11.1.5 推論構成ヘルパー                     | 142        |
| 11.1.6 モデル構成ヘルパー                    | 144        |
| 11.2 Knowledge bases reference      | 151        |
| 11.2.1 ビュー                          | 151        |
| 11.2.2 タイプ                          | 152        |
| 11.2.3 ナレッジベース機能                    | 153        |
| 11.2.4 ボリュームファンクション                 | 156        |
| 11.3 Operations reference           | 157        |
| 11.3.1 エラーログファンクション                 | 157        |
| 11.3.2 エラーカテゴリ                      | 162        |

|        |                                           |     |
|--------|-------------------------------------------|-----|
| 11.4   | Functions reference . . . . .             | 162 |
| 11.4.1 | helpb.chunk_text . . . . .                | 162 |
| 11.4.2 | quietb.parse_html . . . . .               | 164 |
| 11.4.3 | quietb.parse_pdf . . . . .                | 164 |
| 11.4.4 | quietb.perform_ocr . . . . .              | 165 |
| 11.4.5 | quietb.summarize_text . . . . .           | 166 |
| 11.4.6 | quietb.summarize_text_aggregate . . . . . | 167 |
| 11.4.7 | パラメーターのデフォルトのクイックリファレンス . . . . .         | 168 |
| 11.5   | Pipelines reference . . . . .             | 169 |

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 AIDB

## 1.1 どこから始めるべきか

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*** \*\* **オプション** — 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、パイプラインがそこから読み取ることができるように 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。

## 1.2 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                             |

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>Knowledge bases reference</i> | Semantic search, hybrid search, and vector index options                          |
| <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 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 |
|------------------------------------------|--------------|
| <a href="#">AIDB 7.3.2 release notes</a> | 17 Apr 2026  |
| <a href="#">AIDB 7.3.0 release notes</a> | 08 Apr 2026  |
| <a href="#">AIDB 7.2.0 release notes</a> | 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  |

### 2.1 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 |
|------------------------------------------|--------------|
| <a href="#">AIDB 7.3.2 release notes</a> | 17 Apr 2026  |
| <a href="#">AIDB 7.3.0 release notes</a> | 08 Apr 2026  |
| <a href="#">AIDB 7.2.0 release notes</a> | 30 Mar 2026  |

## 2.1.1 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 | <b>マルチパイプラインのナレッジベース</b> 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 | <b>ハイブリッド検索 BYOSQL</b> 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 | <b>マルチステップパイプラインの取得サポート</b> 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 | <b>HuggingFace API のサポート</b> 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 | <b>モデルオプション</b> 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 | <b>パイプラインデッドレターキュー DLQ – フェーズ 1:</b> 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 | <b>構成可能な CPU 制限</b> 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 | <b>HCP モデル同期ヒント</b> 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。                                           |
| Enhancement | <b>オプションバージョン付けインターフェイス OVI</b> 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できるようになり、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 | <b>非推奨 UX の向上</b> 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。                                                                |

[次のページに続く](#)

表 3 – 前のページからの続き

| Type        | Description                                                                                                                                                       |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enhancement | <code>`aidb.models`</code> ビューの <code>`functions`</code> タグ <code>`aidb.models`</code> ビューには <code>`functions`</code> 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     | <b>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</b> 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      | <b>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</b> 、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。                                                |

## 2.1.2 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 | <b>パイプライン DLQ – フェーズ 2 ノンブロッキングエラー</b> 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 | <b>一時的なエラー分類</b> モデルエラーとデータベースエラーが必要に応じて <code>`PipelineTemporary`</code> 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 | <b>DLQ ユーザ API</b> 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 | <b>セマンティックナレッジベース検索のコメント</b> 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 ビューとファン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                                                      |
|-----------------------------|------------------------------------------------------------------|
| <b>Views</b>                |                                                                  |
| <code>aidb.preparers</code> | <code>aidb.pipelines</code> (filter by operation type if needed) |

次のページに続く

表 5 – 前のページからの続き

| Deprecated objects                                    | Replacement                                                         |
|-------------------------------------------------------|---------------------------------------------------------------------|
| <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)                            |
| <b>Functions</b>                                      |                                                                     |
| <i>aidb.create_table_knowledge_base(...)</i>          | <i>aidb.create_pipeline(... step_1 =&gt; 'KnowledgeBase')</i>       |
| <i>aidb.create_volume_knowledge_base(...)</i>         | <i>aidb.create_pipeline(... step_1 =&gt; 'KnowledgeBase')</i>       |
| <i>aidb.set_auto_knowledge_base(...)</i>              | <i>aidb.update_pipeline(name, auto_processing =&gt; ...)</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 =&gt; 'KnowledgeBase')</i>       |
| <i>aidb.create_retriever_for_volume(...)</i>          | <i>aidb.create_pipeline(... step_1 =&gt; '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 =&gt; ...)</i>        |
| <i>aidb.enable_auto_embedding_for_table(...)</i>      | <i>aidb.update_pipeline(name, auto_processing =&gt; 'Live')</i>     |
| <i>aidb.disable_auto_embedding_for_table(...)</i>     | <i>aidb.update_pipeline(name, auto_processing =&gt; '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 =&gt; ...)</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 =&gt; mode)</i>       |
| <i>aidb.upsert_chunk_text(name, key, value)</i>       | Use pipeline execution via <i>aidb.run_pipeline()</i>               |
| <i>aidb.upsert_summarize_text(name, key, value)</i>   | Use pipeline execution via <i>aidb.run_pipeline()</i>               |
| <i>aidb.upsert_parse_html(name, key, value)</i>       | Use pipeline execution via <i>aidb.run_pipeline()</i>               |
| <i>aidb.upsert_parse_pdf(name, key, value)</i>        | Use pipeline execution via <i>aidb.run_pipeline()</i>               |
| <i>aidb.upsert_perform_ocr(name, key, value)</i>      | Use pipeline execution via <i>aidb.run_pipeline()</i>               |

### 2.1.3 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 | <b>パイプライン DLQ – フェーズ 3 メトリックとエラーログ API</b> 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 | <b>マルチユーザーパイプライン PGD トリガーラッパー</b> 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 | <b>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ابل</b> ナレッジベースステابل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 | <b>PDF 解析の向上</b> <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 | <b>HTTP 429 再試行処理</b> 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 | <b>ソースキーによる取得結果の重複除去</b> 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     | <b>間違った入力タイプエラーの処理を修正します。</b> 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     | <b>マルチパイプライン ID のバグを修正します。</b> 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     | <b>自動処理状態のバグを修正する</b> 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。                                                              |



## 第3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らします。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。

### 3.1 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。

### 3.2 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、AI 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 AI 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、AI 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。

## 3.3 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。VectorChord は、高パフォーマンスの代替手段としてもサポートされており、大規模なインデックス作成とより優れたスループットを提供します。

### 3.3.1 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4章 Compatibility

### 4.1 AIDB

#### 4.1.1 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。

#### 4.1.2 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。

### 4.2 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 Installing AIDB

AIDB は、スタンドアロンの PostgreSQL 拡張機能です。AI Factory または Hybrid Manager に依存せず、Postgres インスタンスに直接インストールされます。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 AI Factory を介して展開する場合、AIDB はそのセットアップの一部としてバンドルおよび構成されます。代わりに AI 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。

1. VectorChord をインストールしますオプション高性能、低レイテンシーのベクトル取得。[Installing VectorChord](#) を参照してください。
2. VectorChord-BM25 インストールオプションオプション。ハイブリッドキーワード+セマンティック検索が必要な場合にのみ必要です。[Installing VectorChord-BM25](#) 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。

### 5.1.2 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` に既に他の拡張機能がリストされている場合、コンマ区切りを使用して `aidb` を追加します。順序は関係ありません。 .. ::

1. Postgres を再起動します。

#### 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。

#### VectorChord を有効にするオプション

VectorChord をインストールしている場合、データベースに拡張機能を作成します。

```
CREATE EXTENSION vectorchord;
```

ハイブリッド検索 BM25 +ベクターも有効にするには

```
CREATE EXTENSION vectorchord_bm25;
```

VectorChord ナレッジベースは、ナレッジベースレベルで構成できます。Knowledge bases 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。

## インストールの検証

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、および `vectorchord` および `vectorchord_bm25` これらの拡張機能が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                         | 7.3.1   | 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       |
| vectorchord                  | 0.3.0   | public | vectorchord: fast vector search for postgres               |
| vectorchord_bm25             | 0.1.0   | public | vectorchord_bm25: bm25 full-text search for postgres       |

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

### 6.1 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。

#### 6.1.1 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) |

次のページに続く

表 2 – 前のページからの続き

| Model name   | Provider                    | Use                                     |
|--------------|-----------------------------|-----------------------------------------|
| <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、名前ですべて使用します。

```

- - 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.);

```

### 6.1.2 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、オプションのリビジョン、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。 |

### 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、 <i>model</i> 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、 <i>model</i> 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。

### 6.1.3 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(
  my_pipeline,
  ...
  step_1_operation => Embed,
  step_1_options   => aidb.knowledge_base_config(my_bert, Text),
  ...
);
```

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。

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。

## 6.2 Support matrix

### 6.2.1 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 8 – 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 =&gt; '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 =&gt; '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 =&gt; '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 =&gt; '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 =&gt; '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 =&gt; '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 =&gt; '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 =&gt; 'main')</i> |

次のページに続く

表 10 – 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> | <code>aidb.llama_config('HuggingFaceTB/SmolLM2-135M-Instruct', revision =&gt; 'main')</code> |
| <i>HuggingFaceTB/SmolLM-360M-Instruct</i> | <i>HuggingFaceTB/SmolLM2-360M-Instruct</i> | <code>aidb.llama_config('HuggingFaceTB/SmolLM2-360M-Instruct', revision =&gt; 'main')</code> |
| <i>TinyLlama/TinyLlama-1.1B-Chat-v1.0</i> | <i>TinyLlama/TinyLlama-1.1B-Chat-v1.0</i>  | <code>aidb.llama_config('TinyLlama/TinyLlama-1.1B-Chat-v1.0', revision =&gt; 'main')</code>  |

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>        | <code>aidb.t5_config('t5-small', revision =&gt; 'main')</code> |
| <i>t5-base</i>  | <i>t5-base</i>         | <code>aidb.t5_config('t5-base', revision =&gt; 'main')</code>  |
| <i>t5-large</i> | <i>t5-large</i>        | <code>aidb.t5_config('t5-large', revision =&gt; 'main')</code> |

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。

## 6.2.2 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 12 – 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。

## 6.3 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。

### 6.3.1 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
  )
);
```

(次のページに続く)

```

- - 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数の最大数。                           |
| <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。

### 6.3.2 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,
```

(次のページに続く)

```
{ "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。 |

次のページに続く

表 21 – 前のページからの続き

| Parameter           | Type    | Default | Description                        |
|---------------------|---------|---------|------------------------------------|
| <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](https://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。

### 6.3.3 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。               |

### 6.3.4 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<br>`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 24 – 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。

## 第7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 **ストレージ場所** 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。

### 7.1 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 ストレージ場所を AIDB に接続します。データ形式 Text、Image、Pdf と、保存場所内のオプションのサブパスを指定します。 `aidb.create_volume()` で作成されました。

ボリュームが存在したら、Postgres テーブル名と同様に、`aidb.create_pipeline()` の `source` として参照します。

!!!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。

### 7.2 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。

#### 7.2.1 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,
```

(次のページに続く)

(前のページからの続き)

```

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"}
);

```

## 7.2.2 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。

## 7.3 ステップ2 ボリュームを作成する

`aidb.create_volume()` を使用して、PGFS 格納場所を AIDB に接続します。ボリュームは、パイプラインと SQL ファンクションが参照するものです。

```

SELECT aidb.create_volume(
  name          => my_volume,
  server        => 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。パイプラインで参照するために使用されます。                                                                |
| <i>storage_location</i> | アタッチする PGFS ストレージ場所の名前。                                                                          |
| <i>sub_path</i>         | 格納場所内のオプションのパス。同じバケット内の異なるフォルダーを複数のボリュームでポイントする場合に役立ちます。                                         |
| <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。

### 7.3.1 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);
```

### 7.3.2 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);
```

## 7.4 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)
);
```

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

## 7.5 ボリュームの管理

### 7.5.1 リストと削除

すべてのボリュームをリストします。

```
SELECT aidb.list_volumes();
```

ボリュームを削除します。

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

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

### 7.5.2 ボリュームの内容を検査する

ボリュームをパイプラインに接続する前に、次の機能を使用して検証します。

```
- - List all objects in the volume
SELECT * FROM aidb.list_volume_content(my_volume);

- - Read a specific file as BYTEA
SELECT aidb.read_volume_file(my_volume, report.pdf);

- - Read a plain text file as text
SELECT convert_from(
    aidb.read_volume_file(my_volume, notes.txt),
    utf8
);
```

これらの直接アクセス機能は、パイプラインから独立して、外部ストレージに対するカスタム SQL クエリーを構築するのにも役立ちます。

## 第8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                                                                                                                                                                      |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Pipelines overview</a>    | Core concepts: sources, steps, destinations, and how pipelines relate to knowledge bases.                                                                                           |
| <a href="#">Creating pipelines</a>    | Defining a pipeline with <code>aidb.create_pipeline()</code> — source, steps, auto-processing, and volume sources.                                                                  |
| <a href="#">Pipeline steps</a>        | Available step types: <code>ChunkText</code> , <code>ParseHtml</code> , <code>ParsePdf</code> , <code>PerformOcr</code> , <code>SummarizeText</code> , <code>KnowledgeBase</code> . |
| <a href="#">Orchestration</a>         | Auto-processing modes, background workers, observability, and error handling.                                                                                                       |
| <a href="#">Reference → Functions</a> | Full API reference for pipeline types, views, CRUD functions, and config helpers.                                                                                                   |
| <a href="#">Examples</a>              | End-to-end worked example.                                                                                                                                                          |

### 8.1 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 反復インデックススキャン](#) 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。

#### 8.1.1 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。

### 8.1.2 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。

### 8.1.3 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` テーブルで使用可能なオプションをすべて表示できます。

## 8.2 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               |
|----------------------|------------------------------------------------------------------------------------------------|--------------------------------|
| <b>Source</b>        | <code>source</code> , <code>source_key_column</code> , <code>source_data_column</code>         | This page                      |
| <b>Steps</b>         | <code>step_1</code> ... <code>step_10</code> , <code>step_N_options</code>                     | <a href="#">Pipeline steps</a> |
| <b>Orchestration</b> | <code>auto_processing</code> , <code>background_sync_interval</code> , <code>batch_size</code> | <a href="#">Orchestration</a>  |

次のページに続く

表 2 – 前のページからの続き

| Parameter group    | Parameters                                                                          | More information                                                          |
|--------------------|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| <b>Destination</b> | <i>destination</i> , <i>destination_key_column</i> , <i>destination_data_column</i> | Set automatically by the <i>KnowledgeBase</i> step, or specify explicitly |

### 8.2.1 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)
);
```

### 8.2.2 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,
```

(次のページに続く)

```
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。

## 8.3 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。

### 8.3.1 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。

ヘルパー関数 `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。  |
| <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`。 |

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。

### 8.3.2 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`。 |

```

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

```

### 8.3.3 ParsePdf

非標準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
  ),
);

```

(次のページに続く)

(前のページからの続き)

```

    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。

### 8.3.4 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。

ヘルパー関数 `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。

## 8.3.5 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
  ),
```

(次のページに続く)

(前のページからの続き)

```

    step_2          => KnowledgeBase,
    step_2_options  => aidb.knowledge_base_config(model => bert, data_format =>
→ Text)
);

```

### 8.3.6 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 | 'Text' または 'Image'。                                                     |
| <i>distance_operator</i> | <i>aidb.DistanceOperator</i> | NULL     | 類似度メトリック L2 デフォルト 'Cosine、または 'InnerProduct'                            |
| <i>vector_index</i>      | <i>JSONB</i>                 | NULL     | 'aidb.vector_index_hnsw_config()' 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(
    model          => bert,
    data_format    => Text,
    distance_operator => Cosine,
    vector_index   => aidb.vector_index_hnsw_config(m => 16, ef_construction =>
→ 64)
  )
);

```

複数のパイプラインを同じナレッジベースにリンクするには、代わりに `aidb.knowledge_base_config_from_kb(data_format)` を使用します。この手法は、既存のナレッジベースからモデルと距離オペレーター設定を継承します。

## 宛先テーブル

KnowledgeBase ステップは、次のスキーマを使用して `pipeline_<pipeline_name>` という名前の宛先テーブルを自動的に作成します。

| 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。                  |

## マルチパイプラインのナレッジベース

単一のナレッジベースで、複数のパイプラインからエンベディングを集約できます。内部 `knowledge_base_pipeline` ジャンクションテーブルは、これらのマッピングを管理します。`aidb.retrieve_text()` を介して結果を取得する場合、各行には `pipeline_name` 列が含まれるため、各エンベディングを生成したパイプラインを特定できます。

ナレッジベースのビューと統計については、[Knowledge bases reference](#) を参照してください。

完全なエンドツーエンドの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。

## 8.4 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。

### 8.4.1 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つの主要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 <code>`background_sync_interval`</code> 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 <code>`aidb.run_pipeline()`</code> 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
);
```

これらのファン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 <code>Disabled</code> 。                             |
| <code>background_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 <code>Background</code> の場合にのみ適用されます。 |

## 8.4.2 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 の前提条件](#) を参照してください。

### オプションのチューニング

``background\_sync\_interval`` は、ワーカーが新しいデータまたは変更されたデータをポーリングする頻度を制御します。これは、`postgresql.conf` ではなく、`aidb.create_pipeline()` または `aidb.update_pipeline()` を介してパイプラインごとに構成されます。

```
SELECT aidb.create_pipeline(
  name           => my_pipeline,
  source         => my_table,
  auto_processing => Background,
  background_sync_interval => 30 seconds,
  ...
);
```

デフォルトの間隔は、ほとんどのテーブルソースに適しています。AWS S3 のようなクラウドオブジェクトストアによってバックアップされるボリュームソースの場合、各スキャンではリストオペレーションが発生し、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。

### 8.4.3 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。

概要を把握するのに最適な場所は、統計テーブルです。ビュー `helpb.knowledge_base_stats` またはその短いエイリアス `aidb.kbstat` を使用します。ビューには、構成されたすべてのナレッジベースパイプライン、設定されている処理モード、および処理されたレコードに関する統計が表示されます。

```
SELECT * from aidb.kbstat;
__OUTPUT__
  knowledge base      | auto processing | table: unprocessed rows | volume: scans_
→completed | count(source records) | count(embeddings)
-----+-----+-----+-----+
→-----+-----+-----+-----+
kb_table_text_bg      | Background      | 0 |
  | 15 | 15
kb_table_text_manual  | Disabled        | 0 |
  | 15 | 15
kb_table_image_manual | Disabled        | 0 |
  | 3 | 3
kb_table_text_live    | Live            | 0 |
  | 15 | 15
kb_table_image_bg     | Background      | 0 |
  | 3 | 3
kb_volume_text_bg     | Background      | 6 | 7 | 7
kb_volume_text_manual | Disabled        | 0 | 0 | 0
kb_volume_image_bg    | Background      | 4 | 177 | 6
kb_volume_image_manual | Disabled        | 1 | 177 | 6
(9 rows)
```

**変更検出** メカニズムは、自動処理の動作の中心です。ボリュームソースとテーブルソースでは異なります。このため、統計テーブルには、これら 2 つのソースタイプに対して異なる列があります。

- **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。 - 自動処理が無効になっている場合、新しい変更イベントはキャプチャされません。
- **volume: scans completed** これまでに完了したソースの完全なリストの数。
- **count(source records)** このパイプラインのソースに存在するレコードの数。
  - テーブルソースの場合、この数値は常に正確です。
  - ボリュームソースの場合、この数値はフルスキャンが完了した後にのみ更新されます。
- **count(embeddings)** このパイプラインのベクター宛先テーブルに存在するエンベディングの数。

### 8.4.4 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                                     |
|--------------------------|----------|------------------|-------------------------------------------------|
| <i>RecordTemporary</i>   | 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。                  |
| <i>RecordPermanent</i>   | 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。                     |
| <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
```

```
- - - - -
→ - - - - -
```

(次のページに続く)

(前のページからの続き)

```

→-----
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,
  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 `promoteb.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、[モニタリングとオブザーバビリティ](#) を参照してください。

## 8.5 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。

### 8.5.1 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。            |
| <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。 |

[次のページに続く](#)

表 18 – 前のページからの続き

| Value                 | Description                   |
|-----------------------|-------------------------------|
| <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 マンハッタンの距離。 |
| <i>Hamming</i>      | ハミング距離。       |
| <i>Jaccard</i>      | ジャックカード距離。    |

**8.5.2 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。

### 8.5.3 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。                 |
| <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。  |

次のページに続く

表 21 – 前のページからの続き

| Column                            | Type   | Description                 |
|-----------------------------------|--------|-----------------------------|
| <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 <b>パイプラインステータスの統合</b> 。 |
| <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) |  |                            |  |                |
| -----+-----+-----+-----+-----  |  |                        |  |                            |  |                |
| →-----+-----+-----+-----+----- |  |                        |  |                            |  |                |
| →-----+-----+-----+-----+----- |  |                        |  |                            |  |                |
| pipeline__7471a                |  | Background             |  | 0                          |  |                |
|                                |  | 5                      |  | 5                          |  | UpToDate       |
| 0                              |  | 0                      |  |                            |  |                |
| pipeline__7471b                |  | Background             |  | 0                          |  |                |
|                                |  | 5                      |  | 5                          |  | UpToDate       |
| 0                              |  | 0                      |  |                            |  |                |
| animal_facts_kb                |  | Disabled               |  | 0                          |  |                |
|                                |  | 99                     |  | 372                        |  | UpToDate       |
| 0                              |  | 0                      |  |                            |  |                |
| animal_facts_kb_bert           |  | Disabled               |  | 0                          |  |                |
|                                |  | 99                     |  | 372                        |  | UpToDate       |
| 0                              |  | 0                      |  |                            |  |                |
| mpkb_pipe_int                  |  | Disabled               |  | 0                          |  |                |
|                                |  | 2                      |  | 4                          |  | UpToDate       |
| 0                              |  | 0                      |  |                            |  |                |
| mpkb_pipe_text                 |  | Disabled               |  | 0                          |  |                |
|                                |  | 2                      |  | 4                          |  | UpToDate       |
| 0                              |  | 0                      |  |                            |  |                |

(6 rows)

## 8.5.4 ファンクション

### promoteb.create\_pipeline

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

| Parameter                                 | Type                                        | Default  | Description                                            |
|-------------------------------------------|---------------------------------------------|----------|--------------------------------------------------------|
| <i>name</i>                               | TEXT                                        | Required | パイプラインの名前。最大 50 文字。                                    |
| <i>source</i>                             | TEXT                                        | Required | ソーステーブルまたはボリュームの名前。                                    |
| <i>step_1</i>                             | <a href="#">helpb.PipelineStepOp</a>        | Required | 最初のパイプラインステップのオペレーション。                                 |
| <i>source_key_column</i>                  | TEXT                                        | NULL     | ソーステーブルの一意のキー列。                                        |
| <i>source_data_column</i>                 | TEXT                                        | NULL     | 処理するデータを含む列。                                           |
| <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>                    | <a href="#">helpb.PipelineAutoPr</a><br>モード | 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 <a href="#">パイプラインステップ構成ヘルパー</a> を使用します。 |
| <i>step_2 ... step_10</i>                 | <a href="#">helpb.PipelineStepOp</a>        | 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。        |

次のページに続く

表 23 – 前のページからの続き

| Column                         | Type | Description |
|--------------------------------|------|-------------|
| <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,
  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>          | <a href="#">helpb.PipelineAutoProcessingMode</a> | 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);
```

### 8.5.5 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。                                       |
| <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> ' <sup>*</sup> 。 |

#### 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 ' <code>aidb.chunk_text_config</code> ' 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> ' <sup>*</sup> 。         |

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>reduction_factor</i> | INTEGER | NULL    | `reduce` 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 `aidb.inference_config` 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。     |
| <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.PipelineDataFo</i> | Required | 埋め込まれるデータのフォーマット。       |
| <i>distance_operator</i> | <i>helpb.DistanceOpera</i>  | NULL     | 類似検索のための距離機能。デフォルト `L2` |

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>vector_index</i> | JSONB | NULL    | ベクトルインデックス構成 <i>Vector</i> インデックス構成ヘルパー 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.PipelineDataFc</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。 |

次のページに続く

表 34 – 前のページからの続き

| Parameter             | Type             | Default | Description                |
|-----------------------|------------------|---------|----------------------------|
| <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
);
```

### 8.5.6 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    | ベクターストレージタイプ。        |
| <i>m</i>                | INTEGER | NULL    | ノードごとの最大接続数デフォルト 16。 |

次のページに続く

表 35 – 前のページからの続き

| Parameter              | Type    | Default | Description         |
|------------------------|---------|---------|---------------------|
| <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。         |

**i 注釈**

HNSW は、最大 2000 ディメンションをサポートします。高次元ベクトルの場合、`aidb.vector_index_disabled_config()` を使用します。!!!

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    | ベクターストレージタイプ。      |
| <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\_chord\_hnsw\_config**

VectorChord HNSW 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    | ベクターストレージタイプ。        |
| <i>m</i>                | INTEGER          | NULL    | ノードごとの最大接続数。         |
| <i>ef_construction</i>  | INTEGER          | NULL    | ビルド時の検索の深さ。          |
| <i>max_connections</i>  | INTEGER          | NULL    | グラフ内の最大接続数。          |
| <i>ml</i>               | DOUBLE PRECISION | NULL    | グラフレイヤー構造を制御するレベル乗数。 |

**例**

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

**promoteb.vector\_index\_chord\_vchordq\_config**

VectorChord Vchordq 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    | ベクターストレージタイプ。                    |
| <i>lists</i>               | TEXT    | NULL    | クラスターの数。                         |
| <i>spherical_centroids</i> | BOOLEAN | NULL    | クラスターリングするときは、球形の正規化された重心を使用します。 |

**例**

```
SELECT aidb.vector_index_chord_vchordq_config(lists => 100, spherical_centroids => true);
```

**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     | ベクターストレージタイプ。 |

### 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 aidb.vector_index_disabled_config();
```

## 8.5.7 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。

## 8.6 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。

### 8.6.1 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,
  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
);

```

### 8.6.2 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
  )
);

```

(次のページに続く)

(前のページからの続き)

```
)
);
```

### 8.6.3 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
);
```

### 8.6.4 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_
```

(次のページに続く)

(前のページからの続き)

```
→single_step
(1 row)
```

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

### 8.6.5 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,
  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(
```

(次のページに続く)

(前のページからの続き)

```
    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;
```

## 第9章 Knowledge bases

ナレッジベースは、エンベディングのベクトルインデックス付きストアです。これは、パイプラインに **ナレッジベース** が含まれる場合に自動的に作成されます。パイプラインはエンベディング生成とインデックス作成を処理し、ナレッジベースは、結果のクエリ可能ストアです。

| Page                     | What it covers                                                                       |
|--------------------------|--------------------------------------------------------------------------------------|
| <i>Hybrid search</i>     | Combining semantic search with relational filters and BM25 keyword search            |
| <i>Vector extensions</i> | VectorChord and VectorChord-BM25 for high-performance dense and sparse vector search |
| <i>Examples</i>          | End-to-end worked examples for table and volume sources                              |

### 9.1 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。

### 9.2 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。

#### 9.2.1 `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。

### 9.2.2 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。

## 9.3 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。

### 9.3.1 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];
```

## 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。
- **BM25 全文検索** BM25 インデックス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。

## 前提条件

### pgvector 反復インデックススキャン

pgvector 0.8.0 以降が必要です EDB Postgres AI のデフォルトは 0.8.1 です。

反復インデックススキャンは、ベクトル `ORDER BY` と `WHERE` フィルターを効率的に組み合わせるためのキーイネーブラーです。これらがないと、ベクトルインデックスはフィルタリング前に固定数の候補をプリフェッチします。つまり、選択性の高いフィルタが `LIMIT` 要求よりも少ない結果を返す場合があります。反復スキャンが有効になっている場合、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。

HNSW インデックスの反復スキャンを有効にします。

```
SET hnsw.iterative_scan = relaxed_order;
```

IVFFlat インデックスの場合

```
SET ivfflat.iterative_scan = on;
```

非常に選択的なフィルター一致する行がほとんどない場合、安全制限を設定して、スキャンがインデックス全体を走査しないようにします。

```
SET hnsw.iterative_scan_max_tuples = 20000;
```

### 重要

反復スキャンを有効にするには、ベクトル `ORDER BY` と `WHERE` フィルターが同じクエリレベルである必要があります CTE 境界で分離されていません。以下のクエリパターンは、それに応じて構造化されています。

## BM25 オプション

BM25 フルテキストフィルタリングには、`vchord_bm25` および `pg_tokenizer` 拡張機能が必要です。これらはユーザーがインストールおよび管理します。設定の詳細については、[VectorChord-BM25](#) を参照してください。

## ヘルパーファンクションとメタデータ

### `promoteb.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ーをエンコードします。

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

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

### `aidb.knowledge_bases` ビュー

`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ー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                    |
| <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;
```

### promoteb.rerank\_text()

オプション候補者を取得した後、最終的な関連性の高い結果を得るためのリ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。

## クエリパターン

### 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 hnsf.iterative_scan = relaxed_order;

SELECT s.id, s.description,
       kb.value <=> aidb.kb_query_encode(public.my_kb_vector, summer fashion) 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)
LIMIT 10;
```

### Vector + BM25 フィルタとして

WHERE 条件として BM25 スコアしきい値を使用し、結果をベクトル距離で並べ替えます。これにより、キーワード一致も満たす、意味的に最も近い結果が返されます。

```
SET hnsf.iterative_scan = relaxed_order;

SELECT s.id, s.description,
       kb.value <=> aidb.kb_query_encode(public.my_kb_vector, summer fashion) AS
→distance
FROM my_kb_vector kb
JOIN my_source_table s ON s.id::TEXT = kb.source_id
WHERE s.bm25_col <&> to_bm25query(my_bm25_idx, tokenize(shoes, bert)) < -1.0 -- BM25
→filter
ORDER BY kb.value <=> aidb.kb_query_encode(public.my_kb_vector, summer fashion)
LIMIT 10;
```

### Vector + BM25 相互ランク融合経由

Reciprocal Rank Fusion(RRF) は、セマンティック検索とキーワード検索を独立して実行し、結果セットをマージおよびスコア付けします。これにより、オーバーフィルタリングが回避され、いずれのディメンションでも良好なランクの結果が表面化されます。

```
WITH semantic AS (
  SELECT source_id AS id,
         ROW_NUMBER() OVER (ORDER BY value <=> aidb.kb_query_encode(public.my_kb_
→vector, summer fashion) AS rank
  FROM my_kb_vector
  ORDER BY value <=> aidb.kb_query_encode(public.my_kb_vector, summer fashion)
  LIMIT 50
),
fulltext AS (
  SELECT id::TEXT AS id,
         ROW_NUMBER() OVER (ORDER BY bm25_col <&> to_bm25query(my_bm25_idx,
→tokenize(shoes, bert))) AS rank
  FROM my_source_table
  ORDER BY bm25_col <&> to_bm25query(my_bm25_idx, tokenize(shoes, bert))
  LIMIT 50
)
SELECT COALESCE(s.id, f.id) AS id,
       COALESCE(1.0 / (60 + s.rank), 0) + COALESCE(1.0 / (60 + f.rank), 0) AS rrf_
→score
FROM semantic s
FULL OUTER JOIN fulltext f ON s.id = f.id
ORDER BY rrf_score DESC
LIMIT 10;
```

RRF 式の定数 60 は、標準のランク平滑化係数です。これを調整して、高ランクと低ランクの結果の相対的な重みを制御します。

### Vector +リレーショナル+BM25 すべて 3 つ

3 つのフィルタリングメカニズムすべてを単一のクエリーに組み合わせます。

```
SET hnsw.iterative_scan = relaxed_order;

SELECT s.id, s.description,
       kb.value <=> aidb.kb_query_encode(public.my_kb_vector, summer fashion) AS
→distance,
       s.bm25_col <&> to_bm25query(my_bm25_idx, tokenize(shoes, bert)) AS bm25_score
FROM my_kb_vector kb
JOIN my_source_table s ON s.id::TEXT = kb.source_id
WHERE s.category_id = 2
      -- relational filter
```

(次のページに続く)

(前のページからの続き)

```

AND s.bm25_col <=> to_bm25query(my_bm25_idx, tokenize(shoes, bert)) < -1.0 --
BM25 filter
ORDER BY kb.value <=> aidb.kb_query_encode(public.my_kb_vector, summer fashion)
LIMIT 10;

```

## 実用的な例

### 例 1 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

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
);

```

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);

```

`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,
  source_data_column => product_name,
  step_1        => KnowledgeBase,
  step_1_options => aidb.knowledge_base_config(
                                model      => bert,

```

(次のページに続く)

(前のページからの続き)

```

        data_format => Text
    )
);

```

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

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

\_\_OUTPUT\_\_

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

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

```
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)
LIMIT 10;
```

\_\_OUTPUT\_\_

| source_id | product_name  | category   |
|-----------|---------------|------------|
| 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。

## 例 2 RRF を使用した BM25 全文検索

この例では、逆位ランク融合 RRF を使用してセマンティックベクトル検索と BM25 キーワード検索を組み合わせて、2つの結果セットをマージする方法を示しています。RRF は、セマンティックランキングとキーワードランキングでの位置を組み合わせて結果を再ランク付けし、両方で高いスコアの項目を表示します。この例は、例 1 と同じ products テーブルと food\_products パイプラインに基づいて構築されています。

### ソーステーブルに BM25 インデックスを追加します

pg\_tokenizer および vchord\_bm25 拡張機能を有効にします。

```
CREATE EXTENSION IF NOT EXISTS pg_tokenizer;
CREATE EXTENSION IF NOT EXISTS vchord_bm25;
```

BERT モデルを使用してトークナイザーを作成します。

```
SELECT tokenizer_catalog.create_tokenizer(bert, $$
    model = "bert_base_uncased"
$$);
```

BM25 ベクトル列をソーステーブルに追加し、トークン化された値を入力します。

```
ALTER TABLE products ADD COLUMN embedding_bm25 bm25_catalog.bm25vector;
UPDATE products SET embedding_bm25 = tokenizer_catalog.tokenize(product_name, bert);
```

新しい列に BM25 インデックスを作成します。

```
CREATE INDEX products_bm25_idx ON products USING bm25 (embedding_bm25 bm25_catalog.
→bm25_ops);
```

### RRF を使用したハイブリッドクエリ

```
SET hnsf.iterative_scan = relaxed_order;
SET search_path TO public, tokenizer_catalog, bm25_catalog;

WITH semantic AS (
    SELECT source.id,
           source.product_name,
           source.category,
           ROW_NUMBER() OVER (ORDER BY vector.value <=> aidb.kb_query_encode(public.
→pipeline_food_products, fast food) AS rank
    FROM public.pipeline_food_products AS vector
    JOIN products AS source ON source.id::TEXT = vector.source_id
    ORDER BY vector.value <=> aidb.kb_query_encode(public.pipeline_food_products,
→fast food)
    LIMIT 20
),
keyword AS (
    SELECT id,
           product_name,
           ROW_NUMBER() OVER (ORDER BY embedding_bm25 <&> to_bm25query(products_bm25_
→idx, tokenize(fast food, bert))) AS rank
    FROM products
    ORDER BY embedding_bm25 <&> to_bm25query(products_bm25_idx, tokenize(fast food,
→bert))
```

(次のページに続く)

```

LIMIT 20
)
SELECT
    COALESCE(s.id, k.id) AS_
    →id,
    COALESCE(s.product_name, k.product_name) AS_
    →product_name,
    s.category,
    s.rank AS_
    →semantic_rank,
    k.rank AS_
    →keyword_rank,
    COALESCE(1.0 / (60 + s.rank), 0) + COALESCE(1.0 / (60 + k.rank), 0) AS rrf_
    →score
FROM semantic s
FULL OUTER JOIN keyword k ON s.id = k.id
ORDER BY rrf_score DESC
LIMIT 10;
__OUPUT__
id | product_name | category | semantic_rank | keyword_rank | rrf_score
-----+-----+-----+-----+-----+-----
→-----
1 | Hamburger | meat | 6 | 1 | 0.
→03154495777446597119
4 | Fries | vegetarian | 3 | 4 | 0.
→03149801587301587302
6 | Pizza | meat | 2 | 6 | 0.
→03128054740957966765
8 | Veggie Burger | vegetarian | 1 | 8 | 0.
→03109932497589199614
3 | Fish n Chips | fish | 7 | 3 | 0.
→03079838900734423123
5 | Burrito | meat | 5 | 5 | 0.
→03076923076923076924
2 | Cheeseburger | meat | 9 | 2 | 0.
→03062178588125292193
7 | Sandwich | meat | 4 | 7 | 0.
→03055037313432835821
9 | Kebab | meat | 8 | 9 | 0.
→02919863597612958227

```

RRF スコアは、両方のランキングを組み合わせたものです。セマンティックまたはキーワード検索サーフェスのいずれかで高ランクのアイテムが上部にあり、両方でよくランクされているアイテムが最高スコアを取得します。

## ユーザー管理の BM25 のセットアップ

BM25 は、AI パイプラインによって管理されません。ソーステーブルで独立して設定し、ハイブリッドクエリで参照します。

### 1. 拡張機能のインストール

```
CREATE EXTENSION IF NOT EXISTS pg_tokenizer;
CREATE EXTENSION IF NOT EXISTS vchord_bm25;
```

### 2. トークナイザーを作成します

```
SELECT create_tokenizer(bert, $$
    model = "bert_base_uncased"
$$);
```

### 3. ``bm25vector`` 列とインデックスをソーステーブルに追加します

```
ALTER TABLE my_source ADD COLUMN bm25_tokens bm25vector;
UPDATE my_source SET bm25_tokens = tokenize(description, bert);

CREATE INDEX my_source_bm25 ON my_source USING bm25 (bm25_tokens bm25_ops);
```

### 4. インデックスを最新の状態に保ちます

新しい行は、BM25 結果に表示される前にトークン化する必要があります。データ取り込みプロセスの一部としてトリガーを使用するか、列を更新します。

```
UPDATE my_source SET bm25_tokens = tokenize(description, bert) WHERE bm25_tokens IS_
→ NULL;
```

トークナイザーの完全な構成オプションについては、[VectorChord-BM25](#) および [VectorChord-BM25 in pg\\_extensions](#) 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 BM25 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。

## Vector extensions

AI パイプラインは、ベクター検索用の 2 つの高性能 PostgreSQL 拡張機能と統合しています。密なセマンティック埋め込みベースの検索用の **VectorChord** と、スパースキーワード BM25 検索用の **VectorChord-BM25**。どちらもパイプラインの埋め込みナレッジベースステップの一部として構成され、EDB Postgres AI に組み込まれている標準の `pgvector` 機能を補完します。

| Extension                      | Search type           | Best for                                                                                  |
|--------------------------------|-----------------------|-------------------------------------------------------------------------------------------|
| <i>VectorChord</i> を有効にするオプション | Dense vector (ANN)    | Semantic similarity at billion-scale; lower latency than HNSW/IVFFlat at high dimensions. |
| <i>VectorChord-BM25</i>        | Sparse keyword (BM25) | Exact keyword relevance ranking; complements dense search in hybrid retrieval.            |

これらの拡張機能と一緒に、ハイブリッド検索パイプラインの基礎を形成します。意味的な意味 *VectorChord* とキーワード精度 *VectorChord-BM25* を組み合わせて、高品質の検索結果を生成します。

インストールおよびリリースノートを含む完全な拡張機能のドキュメントについては、以下を参照してください。

- *VectorChord* を有効にするオプション
- *VectorChord-BM25*

### VectorChord-BM25

*VectorChord-BM25* は、*VectorChord* と同じ高性能フレームワークを使用して、ベストマッチング 25BM25 キーワード検索を Postgres にもたらしめます。*VectorChord* は AI 埋め込みの密なセマンティック検索を処理しますが、*VectorChord-BM25* はスパースキーワード検索を処理します。統計的な関連性によってランク付けされた、適切な単語を含むドキュメントの検索。

インストール、構成、および完全なテクニカルリファレンスについては、*VectorChord-BM25 documentation* を参照してください。

### VectorChord-BM25 を使用する場合

BM25 キーワード検索は、ナレッジベースパイプラインでの高密度ベクトル検索を補完します。次の場合に *VectorChord-BM25* を使用します。

- ユーザーは、セマンティックモデルでは高くランクされない可能性がある正確な用語、製品名、または技術識別子で検索します。
- キーワードの関連性と意味的類似性を組み合わせたハイブリッド取得パイプラインを構築したい。
- コーパスが大規模であり、シークエンシャルなテキストスキャンではなく、高速なインデックス付きキーワードルックアップが必要です。

### その仕組み

*VectorChord-BM25* は、`bm25vector` データタイプと `bm25` インデックスを導入しました。テキストは、トークナイザー BERT などを使用してスパースベクトル表現にトークン化され、`bm25vector` として保存され、BM25 インデックスでインデックス付けされ、`<&>` 距離演算子を使用して照会されます。

クエリー時に埋め込みモデルを必要とする高密度ベクトル検索とは異なり、BM25 スコアリングは完全に統計的です。取得時にモデル推論は必要ありません。

## パイプラインとともに VectorChord-BM25 を使用する

VectorChord-BM25 は、自分自身がパイプラインステップオペレーションではありません。代わりに、密な知識ベースと並んで**並列検索レイヤー**として動作します。典型的なパターンは次のとおりです。

1. パイプラインは、KnowledgeBase ステップを介して高密度のナレッジベース埋め込みインデックステーブルを設定します。
2. 別の BM25 インデックステーブルには、`tokenize()` でトークン化された同じまたは関連するコンテンツが保存されます。
3. クエリー時に、両方のインデックスの結果がマージされ、再ランク付けされます。

### ステップ 1 トークナイザーを設定する

BM25 テーブルを作成する前にトークナイザーを初期化します。BERT が最も一般的な選択です。

```
SELECT create_tokenizer(bert, $$
model = "bert_base_uncased"
$$);
```

### ステップ 2 BM25 テーブルを作成する

パイプラインのソースまたは宛先テーブルの横に `bm25vector` 列を含むテーブルを作成します。

```
CREATE TABLE documents_bm25 (
  id      serial PRIMARY KEY,
  source_id bigint,          -- references the pipeline source key
  passage text,
  embedding bm25vector
);
```

### ステップ 3 入力とインデックス付け

コンテンツをロードし、トークン化して、BM25 インデックスを構築します。

```
-- Tokenize text into bm25vector
UPDATE documents_bm25
SET embedding = tokenize(passage, bert);

-- Create the BM25 index
CREATE INDEX documents_embedding_bm25
ON documents_bm25
USING bm25 (embedding bm25_ops);
```

## ステップ 4BM25 で照会する

<&> 演算子と to\_bm25query() を使用して、キーワードランク付けの結果を取得します。

```
SELECT
    id,
    source_id,
    passage,
    embedding <&> to_bm25query(
        documents_embedding_bm25,
        tokenize(PostgreSQL full-text search, bert)
    ) AS bm25_score
FROM documents_bm25
ORDER BY bm25_score
LIMIT 10;
```

## BM25 と緻密な知識ベースの組み合わせ

最も効果的な取得パターンはハイブリッド検索です。密なセマンティッククエリーと BM25 キーワードクエリーの両方を実行し、結果セットをマージします。この例では、検索とユニオンの両方を実行して、ダウンストリームの再ランキングの上位結果を実行します。

```
- - Dense semantic results from the pipeline knowledge base
WITH dense_results AS (
    SELECT key, distance AS score, dense AS source
    FROM aidb.retrieve_key(my_knowledge_base, PostgreSQL full-text search, 10)
),

- - BM25 keyword results from the BM25 table
bm25_results AS (
    SELECT source_id::text AS key,
        embedding <&> to_bm25query(
            documents_embedding_bm25,
            tokenize(PostgreSQL full-text search, bert)
        ) AS score,
        bm25 AS source
    FROM documents_bm25
    ORDER BY score
    LIMIT 10
)

SELECT key, score, source
FROM dense_results
UNION ALL
SELECT key, score, source
FROM bm25_results
ORDER BY score;
```

マージされた結果は、最終的な関連性の高い応答のために再ランキングモデル `aidb.rerank_text()` 経由で渡すことができます。

## ハイブリッド検索との関係

VectorChord-BM25 は、*Hybrid search* で説明されているハイブリッド検索パターンのビルディングブロックの 1 つです。VectorChord を使用した緻密な検索は、セマンティックの幅を提供します。BM25 はキーワードの精度を提供します。一緒に、偽陽性トピック外の意味的一致と偽陰性完全一致結果の見逃しの両方を削減します。

## さらに読む

- [VectorChord-BM25 documentation](#) — インストール、トークナイザー構成、リファレンス、およびリリースノート
- [VectorChord を有効にするオプション](#) — パイプラインを使用した密ベクトルインデックス
- [Hybrid search](#) — デンス検索とスパース検索の組み合わせ
- [パイプラインステップ構成ヘルパー](#) — KnowledgeBase ステップ構成リファレンス

## VectorChord

VectorChord は、高密度ベクトル検索のための高性能 Postgres 拡張機能です。大規模な場合により高速でメモリ効率の高いインデックスタイプで `pgvector` を拡張し、数百万または数十億の埋め込みを含むナレッジベースに最適です。

インストールと完全なテクニカルリファレンスについては、[VectorChord in pg\\_extensions](#) を参照してください。

## VectorChord を使用する場合

標準の `pgvector` HNSW および IVFFlat インデックスは、ほとんどのワークロードでうまく動作します。VectorChord は、次の場合に最適な選択です。

- ナレッジベースには数百万または数十億の埋め込みが含まれ、HNSW インデックスが RAM を消費しすぎています。
- 高スループットでのクエリレイテンシーの低下が必要です。
- 埋め込みは高次元です `pgvector` の HNSW インデックスでサポートされている 2000 次元制限を超えています。

## インデックスタイプ

VectorChord は 2 つのインデックスタイプを提供し、どちらも [Vector インデックス構成ヘルパー](#) を介して構成できます。

## Chord\_hnsw

グラフの接続性と検索の深さの詳細な制御を備えた HNSW の VectorChord 実装。より多くのチューニング制御または大規模なパフォーマンスの向上が必要な場合、pgvector の HNSW からのドロップインアップグレードとして使用します。

`aidb.vector_index_chord_hnsw_config()` で構成します。

| Parameter               | Description                 |
|-------------------------|-----------------------------|
| <i>m</i>                | グラフ内のノードごとの最大接続数デフォルト 16。   |
| <i>ef_construction</i>  | インデックス構築中の検索の深さデフォルト 64。    |
| <i>max_connections</i>  | グラフ接続のハードキャップ。              |
| <i>ml</i>               | グラフレイヤー構造を制御するレベル乗数。        |
| <i>vector_data_type</i> | ベクターストレージタイプたとえば 'float32'。 |

## Chord\_vchordq

非常に大規模なスループットを最大にするように設計された量子化ベースのインデックス IVF-RaBitQ。少量のリコール精度と引き換えに、HNSW よりも大幅に高速な検索と低いメモリ使用量を実現します。

`aidb.vector_index_chord_vchordq_config()` で構成します。

| Parameter                  | Description                        |
|----------------------------|------------------------------------|
| <i>lists</i>               | IVF クラスターの数。値が高いほど、スケールでの精度が向上します。 |
| <i>spherical_centroids</i> | 正規化された重心を使用します - コサイン類似性に推奨されます。   |
| <i>vector_data_type</i>    | ベクターストレージタイプたとえば 'float32'。        |

## パイプラインで VectorChord を使用する

VectorChord インデックスタイプは、`aidb.knowledge_base_config()` の `vector_index` パラメーターを介して構成されます。これは、パイプラインの作成時に `step_N_options` として渡されます。

## Chord\_hnsw を使用したパイプラインの構成

```
SELECT aidb.create_pipeline(
  name          => my_semantic_pipeline,
  source        => source_docs,
  source_key_column => id,
  source_data_column => content,
  destination   => my_knowledge_base,
  step_1        => KnowledgeBase,
  step_1_options => aidb.knowledge_base_config(
    my_embedding_model,
    Text,
    distance_operator => Cosine,
    vector_index      => aidb.vector_index_chord_hnsw_
```

(次のページに続く)

(前のページからの続き)

```

→config(
                                m                => 16,
                                ef_construction => 64,
                                max_connections => 32
                                )
                                ),
    auto_processing    => Live
);

```

### Chord\_vchordq を使用したパイプラインの構成

```

SELECT aidb.create_pipeline(
    name                => my_large_scale_pipeline,
    source              => billion_scale_docs,
    source_key_column   => id,
    source_data_column  => content,
    destination         => my_large_kb,
    step_1              => KnowledgeBase,
    step_1_options      => aidb.knowledge_base_config(
                                my_embedding_model,
                                Text,
                                distance_operator => Cosine,
                                vector_index     => aidb.vector_index_chord_vchordq_
→config(
                                lists                => 1000,
                                spherical_centroids => true
                                )
                                ),
    auto_processing    => Background,
    background_sync_interval => 60 seconds
);

```

### チャンクと VectorChord を使用したマルチステップパイプライン

ドキュメントパイプラインの場合、最初にテキストをチャンクして、VectorChord でインデックスされたナレッジベースに埋め込みます。

```

SELECT aidb.create_pipeline(
    name                => doc_pipeline_vchord,
    source              => raw_documents,
    source_key_column   => id,
    source_data_column  => body,
    destination         => doc_knowledge_base,
    step_1              => ChunkText,
    step_1_options      => aidb.chunk_text_config(200, 250, 25, words),

```

(次のページに続く)

(前のページからの続き)

```

step_2          => KnowledgeBase,
step_2_options  => aidb.knowledge_base_config(
                    my_embedding_model,
                    Text,
                    vector_index => aidb.vector_index_chord_hnsw_config(
                                    m => 32,
                                    ef_construction => 128
                                )
                ),
auto_processing  => Live
);

```

## インデックスタイプの選択

| Scenario                                  | Recommended index                                               |
|-------------------------------------------|-----------------------------------------------------------------|
| Up to ~10M vectors, general use           | <i>chord_hnsw</i>                                               |
| 100M+ vectors, memory-constrained         | <i>chord_vchordq</i>                                            |
| High-dimensional embeddings (>2000 dims)  | <i>chord_vchordq</i>                                            |
| Cosine similarity with normalized vectors | <i>chord_vchordq</i> with <i>spherical_centroids =&gt; true</i> |
| Drop-in pgvector HNSW replacement         | <i>chord_hnsw</i>                                               |

## さらに読む

- [VectorChord documentation](#) — インストール、インデックスリファレンス、およびリリースノート
- [Vector インデックス構成ヘルパー](#) — すべてのインデックス構成関クションの完全なパラメーターリファレンス
- [Hybrid search](#) — VectorChord の密集検索と BM25 キーワード検索の組み合わせ

## 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
    )
);
__OUTPUT__
INFO: Created pipeline "pipeline__6545"
INFO: Created knowledge base "public.pipeline_pipeline__6545" for pipeline "pipeline__6545"
```

(次のページに続く)

(前のページからの続き)

```
→_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

処理可能なファイルを確認します。

```
SELECT * FROM aidb.list_volume_content(volume__6545);
__OUTPUT__
file_name | size | last_modified
-----+-----+-----
```

(次のページに続く)

(前のページからの続き)

```

Bilge Ince.html | 8878 | 2025-02-19 09:58:29 UTC
all_tags.html  | 11170 | 2025-02-19 09:58:29 UTC
hello_world.html | 108 | 2025-02-19 09:58:30 UTC
simple.html     | 662 | 2025-02-19 09:58:29 UTC
(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)

```



## 第10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                                                                |
|----------------------------------------|--------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| <a href="#">Text chunking</a>          | <code>aidb.chunk_text()</code>                                                       | Splitting text into segments for LLM context windows                          |
| <a href="#">Data parsing</a>           | <code>aidb.parse_html()</code> , <code>aidb.parse_pdf()</code>                       | Extracting readable text from HTML and PDF inputs                             |
| <a href="#">Performing OCR</a>         | <code>aidb.perform_ocr()</code>                                                      | Extracting text from images using an OCR-capable model                        |
| <a href="#">Summarization cookbook</a> | <code>aidb.summarize_text()</code> ,<br><code>aidb.summarize_text_aggregate()</code> | Generating and aggregating AI-powered text summaries, use cases, and cookbook |
| <a href="#">AI inference functions</a> | <code>aidb.encode_text()</code> , <code>aidb.decode_text()</code> , and variants     | Embedding text and images into vectors; generating completions                |
| <a href="#">Text reranking</a>         | <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。

### 10.1 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)

```

→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)

```

### 10.1.1 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

この操作は、データの形状を変換し、`part_id` 列を導入することによりコレクションのネストを自動的に解除します。パイプラインがマルチパート出力を処理する方法の詳細については、[Creating pipelines](#) を参照してください。

## 10.2 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。

### 10.2.1 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>,
  options => {"method": "StructuredPlaintext"} -- Default
);
```

(次のページに続く)

(前のページからの続き)

```

__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。

### 10.2.2 PDF を解析する

`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!
|
(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

どちらの解析操作もネスト解除を実行します。これは、単一の入力マルチページ PDF のような複数行のデータに変換し、`part_id` を自動的に生成してセグメントを正しい順序に保つことを意味します。

## 10.3 Performing OCR

`aidb.perform_ocr()` ファンクションを使用して、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 1 OCR 対応モデルを登録する

```
SELECT aidb.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

```
SELECT * FROM aidb.perform_ocr(
  decode(/9j/4AAQSkZJRgABAQAAQABAAD/2wCEAAUFBQUBQUGBgUICAcIC
  AsKCQkKCxEMDQwNDBEaEBMQEBMQGhcbFhUWGxcpIBwcICkvJyUnLzkzMzlhRE
  ddXX0BBQUBQUBQYGBQgIBwgICwoJCQoLEQwNDA0MERoQExAQExAaFxsWFRYb
  FykgHBwgKS8nJScvOTMzOUdER11dfff/CABEIAWYDKgMBIgACEQEDEQH/xAAvAA
  EAAQUBAQEAAAAAAAAAAAAAAAAQIDBAcIBgkFAQEAAAAAAAAAAAAAAAAAA/9oA
  DAMBAAIQAxAAAAADsuKLhCRRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRC
  CRRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRCRC
  UAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAWrtq6U1U1AAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFq7au1NVNQAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABau2rpTVTUAAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAWrtq6U1U1AAAAAAAAAAAA
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFq7au1NVNQAAAAAAAAAAAA
```

(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)

```

uma02U7c0tLvfvvs6h8K4muKidcSnVLaTYq6Y5tcLy6svQU296q5RZC2bYjK0nSAaxWN457Qommqn
WnXa0FrQsqfonYnqxMv6PZowyZHiqySMVHovvmmxrqtW/nV3xtu9VWkk0bijHuD8qz9Ekd9Wnbl4Q
psrijMq4cxdG1QUHkbqGQu+3ZSchukv1r2GfUKHeYTQrYMGsYyaMWvVikmU9ai90No5tH2LaTZr1V
oUfo6AVPAf8VQXx5ANbjzXWzdU0kRERdS1uspGw/wCrH0uy67yy5Rirhn+eXY6pZZda7ouMFsdSNqW
xARc01vDqLs7rblIRx0pyaqc9cUaZfj1+afaj2EvybiPcF0jpl/k4ZphlZvGlxkenJGyJn+sQk//bT
H5Ee17xxknfqNqxV75zj7Euv9005CybEu9qi5eZ1xRbthpxsnCWdlF3a+E4K4Z0yrstTMMCu8Xv7AtgPo
eUkGHTBKvm15ykRSszibQzc5bXCj jrHLpFNZvk55ivGqEcS8k6jpDAs6/
→jekoqDZ3mZz7xWMgZyQRxHbLXK
GQXn6lj7CseCR+prPS8Z0Z7YvYvqohl3Vq29LJ9N11sZexW8IcjJM4pg6fvsU5ou/
→IN81xVfVgW4izRhqojx
  options => {"model": "my_paddle_ocr_model"}
);

__OUTPUT__
create_model
-----
my_paddle_ocr_model
(1 row)

part_id |      text
-----+-----
      0 | Tesseract sample
(1 row)

```

## 10.4 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。

### 10.4.1 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

```
edb_notranlate_1 sql SELECT promoteb.summarize_text( input => '長い記事テキスト...', options =>
fenced_text_config( 'my_t5_model', promote => '非技術者向けのキーポイントの要約' )::json );
```

```
edb_notranlate_2 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 );
```

10.5 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 ``

10.6 Text summarization

|                                                                                |                     |                                                          |                                                                                                     |
|--------------------------------------------------------------------------------|---------------------|----------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| <i>Provider</i>                                                                | <i>'bert_local'</i> | Embedding-only provider used for summarization           | Use <i>completions</i> , <i>openai_completions</i> , <i>t5_local</i> , or another language provider |
| <i>does not support language operations</i>                                    |                     |                                                          |                                                                                                     |
| <i>Invalid parameters: 'desired_length' must be greater than 0</i>             |                     | Invalid chunk config                                     | Use a positive integer for <i>desired_length</i>                                                    |
| <i>Invalid parameters: 'max_length' must be &gt;= 'desired_length'</i>         |                     | <i>max_length</i> is smaller than <i>desired_length</i>  | Set <i>max_length</i> >= <i>desired_length</i>                                                      |
| <i>Invalid parameters: 'overlap_length' must be less than 'desired_length'</i> |                     | 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)                                               |

10.6.1 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
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(
```

(次のページに続く)

(前のページからの続き)

```

        review_text,
        aidb.summarize_text_config(
            local-llama,
            aidb.chunk_text_config(120, 120, 15, words),
            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 |

## 10.6.2 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,

```

(次のページに続く)

(前のページからの続き)

```

    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. デील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 [既知の制限](#) 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
        ) 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
```

(次のページに続く)

(前のページからの続き)

```
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。

## 6. エグゼクティブサマリー リデュース戦略

使用する場合詳細な説明ではなく、短く圧縮された概要が必要です。ダッシュボードとレポートに適しています。

```
SELECT p.product_name,
       aidb.summarize_text_aggregate(
         sn.note_text,
```

(次のページに続く)

```

    aidb.summarize_text_config(
        my-llm,
        aidb.chunk_text_config(100, 100, 10, words),
        NULL,
        reduce,
        3 -- reduction_factor: each chunk targets desired_length/3 words
    )::json
) AS executive_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;

```

仕組みテキストは 100 ワードのチャンクに分割されます。各チャンクは `desired_length / reduction_factor` ワード  $100/3 \approx 33$  ワードをターゲットに要約されます。最終結果が `desired_length` 内に収まるまで、要約は結合、再チャンク、および再要約されます。出力は約 400～900 文字高圧縮です。

- `reduction_factor=3` は、各チャンクをそのサイズの約 33% に圧縮します。rf=2 ～50% まで。
- Reduce では常にワードベースのチャンクを使用します。char ベースは失敗する場合があります [既知の制限](#) を参照してください。
- 最小安全なチャンクサイズ 50 ワード。

## 7. 詳細な概要追加戦略

使用する場合テキストのすべての部分の詳細を保持する包括的なカバレッジが必要です。

```

SELECT p.product_name,
    aidb.summarize_text_aggregate(
        sn.note_text,
        aidb.summarize_text_config(
            my-llm,
            aidb.chunk_text_config(200, 200, 20, words),
            NULL,
            append
        )::json
    ) AS detailed_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;

```

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
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(
```

(次のページに続く)

```

        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 $$
    SELECT aidb.summarize_text_config(
        p_model,
        prompt => format(
            Categorize the issues in this text into ONLY these categories: %s.
            For each category, give a count and one representative example.
            Format: CATEGORY: count - "example quote",
            array_to_string(p_categories, , )
        )
    )::json;
$$ 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

```

(前のページからの続き)

```
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、またはボイ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 $$
BEGIN
    RETURN regexp_replace(
        regexp_replace(
            regexp_replace(
                regexp_replace(p_text,
                    -- Remove email signatures
                    E(Best regards|Sincerely|Thanks|Cheers), '?s *\\n.*$',
                    , gi),
                -- Remove greeting lines
                E^(Hi|Hello|Dear|Hey)\\s+[A-Za-z]+, '?s*\\n',
                , gi),
            -- Remove filler phrases
            E\\b(basically|essentially|actually|literally|you know|I mean|sort of|kind
of)\\b,
            , gi),
        -- Collapse whitespace
        E\\s+, , g);
END;
$$ 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
```

(次のページに続く)

(前のページからの続き)

```
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ラ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 |

次のページに続く

表 4 – 前のページからの続き

| Group size  | Duration | Result size | Notes                               |
|-------------|----------|-------------|-------------------------------------|
| 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*) |

次のページに続く

表 7 – 前のページからの続き

| Scenario              | 100 notes       | 500 notes        | 1,000 notes       |
|-----------------------|-----------------|------------------|-------------------|
| 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(
```

(次のページに続く)

```

        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

```

(次のページに続く)

(前のページからの続き)

```

    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.

```

(次のページに続く)

(前のページからの続き)

```
- - Fast (constrained, LLM generates ~200 chars):
Give exactly 3 bullet points, max 1 sentence each.
```

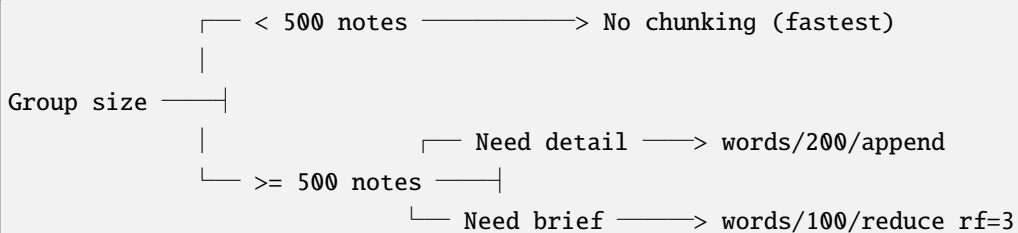
### 3. 抽出プレフィルタリング

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 |

次のページに続く

表 11 – 前のページからの続き

| Step                  | Time          |
|-----------------------|---------------|
| <b>Total per call</b> | <b>~3–12s</b> |

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。

## 10.7 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。

### 10.7.1 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(
  input    => What is pgvector?,
  options => {"model": "my_embedder"}
);
```

\_\_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。

### 10.7.2 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。

### 10.7.3 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。

### 10.7.4 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、オプションで `inference_config` オブジェクトを渡します。

```

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
    )

);

```

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。

### 10.7.5 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 * FROM aidb.decode_text_batch(
    model_name => my_model_name,
    input      => ARRAY[
        What is a vector database?,
        What is retrieval-augmented generation?
    ]
);

```

\_\_OUTPUT\_\_

| id | value                                                 |
|----|-------------------------------------------------------|
| 0  | A vector database stores high-dimensional vectors ... |
| 1  | 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。

### 10.7.6 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 `decode_text()`、`decode_text_batch()`、または `summarize_text()` の options JSONB 内の "inference\_config" キーとして渡します。

```
SELECT aidb.decode_text(
  input    => Write a haiku about PostgreSQL.,
  options => jsonb_build_object(
    model,          my_llm,
    inference_config, aidb.inference_config(
      temperature => 0.9,
      max_tokens  => 64,
      seed        => 42
    )
  )
);
```

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。

## 10.8 Text reranking

`aidb.rerank_text()` ファンクションを使用して、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(
  query    => What is retrieval-augmented generation?,
  inputs   => 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.
  ]
);
```

(次のページに続く)

(前のページからの続き)

```

    ],
    options => [{"model": "my_reranker"}]
)
ORDER BY relevance_score DESC;

__OUTPUT__
id | relevance_score
-- +-----
 2 |           0.97
 0 |           0.91
 1 |           0.43
(3 rows)

```

id 列は、inputs 配列内の各文字列のゼロベースのインデックスに対応し、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。

### 10.8.1 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
        source_id,
        content
    FROM aidb.retrieve_text(my_kb, What is RAG?, 20)
    JOIN my_source_table ON source_id = id::TEXT
)
SELECT
    c.source_id,
    c.content,
    r.relevance_score
FROM candidates c
JOIN aidb.rerank_text(
    query    => What is RAG?,
    inputs   => ARRAY(SELECT content FROM candidates),
    options  => [{"model": "my_reranker"}]
) r ON r.id = (
    SELECT array_position(ARRAY(SELECT content FROM candidates), c.content) - 1
)
ORDER BY r.relevance_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 **モデル構成ヘルパー** を使用します。たと

例えば、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。

## 第11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、およびブロ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                             |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------|
| <a href="#">Models reference</a>          | 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 | <a href="#">Integrating models</a>        |
| <a href="#">Knowledge bases reference</a> | Knowledge base views and stats, retrieval functions ( <i>retrieve_key</i> , <i>retrieve_text</i> ), volume functions, and auto-processing                                                                                           | <a href="#">Knowledge bases reference</a> |
| <a href="#">Operations reference</a>      | Pipeline error log functions ( <i>get_error_logs</i> , <i>clear_error_logs</i> , <i>retry_pipeline_errors</i> )                                                                                                                     | <a href="#">Pipeline error log</a>        |
| <a href="#">Functions reference</a>       | Standalone SQL functions: <i>chunk_text</i> , <i>parse_html</i> , <i>parse_pdf</i> , <i>perform_ocr</i> , <i>summarize_text</i>                                                                                                     | <a href="#">SQL functions</a>             |
| <a href="#">Pipelines reference</a>       | Pipeline types, views, CRUD functions, step config helpers, and vector index config helpers                                                                                                                                         | <a href="#">Reference → Functions</a>     |

### 11.1 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。

#### 11.1.1 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    |
| <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)
```

**11.1.2 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<br>`aidb.model_providers`を参照してください。          |
| <i>config</i>              | JSONB   | <code>{}</code> | プロバイダー固有の構成。 <b>モデル構成ヘルパー</b> でビルドします。               |
| <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)

### 11.1.3 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 `skipped` |
| <i>model</i>  | text | 同期モデルの名前                                                          |

**11.1.4 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>text</i>       | TEXT | エンコードするテキスト。   |

## 返品

VECTOR — 入力テキストの埋め込み。

## 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>text</i>       | TEXT[] | エンコードする文字列の配列。 |

## 返品

TABLE(id INT, value VECTOR) — 入力ごとに 1 行、id を介して入力順序を保存します。

## 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>image</i>      | BYTEA | Raw 画像バイト。     |

## 返品

VECTOR — 入力画像の埋め込み。

## 例

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

## helpb.encode\_image\_batch

1 回の呼び出しでバイナリイメージ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>images</i>     | BYTEA[] | 生の画像バイトの配列。    |

## 返品

TABLE(id INT, value VECTOR) — 入力画像ごとに 1 行、id を介して入力順序を保存します。

## 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>text</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。<br>:ref:`aidb.inference_config()`<br><#aidbinference_config>` を使用します。 |

## 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。<br>:ref:`aidb.inference_config`<br><#aidbinference_config>` を使用します。 |

## 返品

TABLE(id INT, value TEXT) — 入力プロンプトごとに 1 行。

## 例

```
SELECT * FROM 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      |

次のページに続く

表 18 – 前のページからの続き

| Column             | Type             | Description         |
|--------------------|------------------|---------------------|
| <i>logit_score</i> | double precision | 関連性スコア高いほど関連性が高い    |
| <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;
```

### 11.1.5 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 19 – 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
);
```

### 11.1.6 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<br>`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 22 – 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 26 – 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 30 – 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<br>`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-...
  )
);
```

## 11.2 Knowledge bases reference

ナレッジベースの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。

### 11.2.1 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>     | <a href="#">helpb.DistanceOperator</a> | 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 <->, <=>    |
| <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  |

次のページに続く

表 33 – 前のページからの続き

| Column | Type | Description |
|--------|------|-------------|
|--------|------|-------------|

helpb.knowledge\_base\_stats

aidb.kbstat としてもアクセスできます。すべてのナレッジベースパイプラインの現在の処理統計を提供します。

:raw-latex: d a`gentb.knowledge\_base\_stats View “aidb.knowledge\_base\_stats” 列 | タイプ | 照合 | Nullable | デフォルト |  
+-----+-----+-----+-----+-----+ 名前 | テキスト | | パイプライン | 整数 | | |  
埋め込みに  
アタッチされたパイプラインの数 | bigint | | | ← これは古い「count(embeddings)」ステータスです。テキスト | | |  
“aidb.pipeline\_metrics” と同じステータスフィールドですが、接続されているすべてのパイプラインの「最悪」ステータスを収集します

例

SELECT \* from aidb.kbm;

\_\_OUTPUT\_\_

| name                                 | pipelines | embeddings | status   |
|--------------------------------------|-----------|------------|----------|
| public.pipeline_pipeline__7471a      | 1         | 5          | UpToDate |
| public.pipeline_pipeline__7471b      | 1         | 5          | UpToDate |
| public.pipeline_animal_facts_kb      | 1         | 372        | UpToDate |
| public.pipeline_animal_facts_kb_bert | 1         | 372        | UpToDate |
| public.mpkb_shared_vectors           | 2         | 4          | UpToDate |

(5 rows)

11.2.2 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 |
|--------------|-------------|
| L2           | ユークリッド距離    |
| InnerProduct | 内積          |
| Cosine       | コサイン類似度     |
| L1           | L1 距離       |
| Hamming      | ハミング距離      |

次のページに続く

表 34 – 前のページからの続き

| Value          | Description |
|----------------|-------------|
| <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.bulk_embedding()`で手動実行 |

**11.2.3 ナレッジベース機能****promoteb.set\_auto\_knowledge\_base**

既存のナレッジベースの自動処理モードを更新します。

**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。 |
| <i>mode</i>                | <a href="#">helpb.PipelineAutoPr</a><br>モード | Required | 新しい自動処理モード。 |

次のページに続く

表 37 – 前のページからの続き

| Parameter                       | Type     | Default | Description                          |
|---------------------------------|----------|---------|--------------------------------------|
| <i>batch_size</i>               | INTEGER  | NULL    | バッチごとに処理されたレコード<br>バックグラウンドおよび無効モード。 |
| <i>background_sync_interval</i> | INTERVAL | NULL    | ポーリング間隔バックグラウンド<br>モード。              |

例

```
SELECT aidb.set_auto_knowledge_base(my_kb, Live);
SELECT aidb.set_auto_knowledge_base(my_kb, Background, background_sync_interval => 1_
↳minute);
SELECT aidb.set_auto_knowledge_base(my_kb, Disabled, batch_size => 200);
```

### 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**では、呼び出しは、マルチステップパイプラインのステップごとに、クエリー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。

### 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>query</i>               | TEXT    | Required | 検索するクエリテキスト。                                  |
| <i>topk</i>                | INTEGER | 1        | 返される結果の数。                                     |
| <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>distance</i> | double precision | クエリーからのベクトル距離                       |
| <i>part_ids</i> | bigint[]         | どの部分つまりパイプラインステップの結果が一致するチャンクまたはページ |

次のページに続く

表 39 – 前のページからの続き

| Column               | Type | Description             |
|----------------------|------|-------------------------|
| <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 |

例

```
SELECT * FROM aidb.retrieve_key(public.pipeline_animal_facts_kb, birds, topk=>3);
```

\_\_OUTPUT\_\_

| key | distance           | part_ids | pipeline_name   |
|-----|--------------------|----------|-----------------|
| 5   | 1.1931772046185758 | {0,0}    | animal_facts_kb |
| 93  | 1.1980633963685476 | {2,0}    | animal_facts_kb |
| 98  | 1.2150919866080878 | {2,0}    | animal_facts_kb |

(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、呼び出しは、マルチステップパイプラインのステップごとに、クエリー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、中間ステップ結果の実際の結果テキストが含まれます。

### 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>query_string</i>        | TEXT    | Required | 検索するクエリテキスト。                                  |
| <i>number_of_results</i>   | INTEGER | 0        | 返される結果の数。`0` の場合、デフォルトの `topk` を使用します。        |
| <i>knowledge_base_name</i> | TEXT    | Required | ナレッジベースの名前。                                   |
| <i>query</i>               | TEXT    | Required | 検索するクエリテキスト。                                  |
| <i>topk</i>                | INTEGER | 1        | 返される結果の数。                                     |
| <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             | ソーステキストの内容                              |
| <i>distance</i>           | double precision | クエリーからのベクトル距離                           |
| <i>part_ids</i>           | bigint[]         | どの部分つまりパイプラインステップの結果が一致するチャンクまたはページ     |
| <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**

ナレッジベースを削除します。つまり、その宛先ベクトルテーブルとその構成。アタッチされたすべてのパイプラインも削除します。基になるソーステーブルは削除しません。

## パラメーター

| Parameter                  | Type | Description     |
|----------------------------|------|-----------------|
| <i>knowledge_base_name</i> | TEXT | 削除するナレッジベースの名前。 |

## 例

**11.2.4 ボリュームファンクション****promoteb.create\_volume**

ナレッジベースデータソースとして使用するために、PGFS 格納場所から AIDB ボリュームを作成します。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。                                         |
| <i>server_name</i> | TEXT | PGFS 格納場所の名前。                                     |
| <i>path</i>        | TEXT | 保存場所内のサブパス。                                       |
| <i>mime_type</i>   | TEXT | データ型 <i>Text</i> 、 <i>Image</i> 、または <i>Pdf</i> 。 |

## 例

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

### promoteb.list\_volumes

データベース内のすべての AIDB ボリュームをリストします。

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

### promoteb.delete\_volume

AIDB ボリュームを削除します。注基になる 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。

## 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);
```

## 11.3 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。

### 11.3.1 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。                                                                                                |
| <i>p_error_category</i> | TEXT     | NULL     | カテゴリによるフィルター<br><i>RecordTemporary</i> 、 <i>RecordPermanent</i> 、 <i>PipelineTemporary</i> 、または <i>PipelinePermanent</i> |
| <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
```

(次のページに続く)

(前のページからの続き)

```
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 |
|---------------|------|-------------|
| pipeline_name | TEXT | パイプラインの名前。  |

返品

| Column           | Type       | Description   |
|------------------|------------|---------------|
| pipeline_step    | smallint   | ステップ番号        |
| step_operation   | text       | オペレーション名      |
| error_category   | text       | エラーカテゴリ       |
| error_count      | bigint     | このグループのエラーの数  |
| latest_failed_at | timestampz | 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.
```

(次のページに続く)

(前のページからの続き)

```
→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。



| 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。<br>`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}
);
```

### 11.4.2 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"}
);
```

### 11.4.3 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 | <i>true</i>  | <i>true</i> 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}
);
```

## 11.4.4 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"}
);
```

### 11.4.5 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 66 – 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    | <code>'append'</code> | 要約戦略 <code>'append'</code> 各チャンクを独立して要約および連結する または <code>'reduce'</code> 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"}
);
```

### 11.4.6 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 67 – 前のページからの続き

| Parameter      | Type | Required | Description                                                            |
|----------------|------|----------|------------------------------------------------------------------------|
| <i>options</i> | JSON | Yes      | `aidb.summarize_text_config()`でビルドされた構成オブジェクト。少なくとも`model`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;
```

### 11.4.7 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-<br/>rize_text_aggregate</i> | <i>input</i>            | — (required)      | —                         |
| <i>summa-<br/>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>       | <code>3</code>            |
| <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 68 – 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>                |

## 11.5 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> ,<br><i>aidb.PipelineSourceType</i> , <i>aidb.PipelineDestinationType</i> ,<br><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> ,<br><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> ,<br><i>aidb.html_parse_config</i> , <i>aidb.pdf_parse_config</i> , <i>aidb.knowledge_base_config</i> ,<br><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> ,<br><i>aidb.vector_index_chord_hnsw_config</i> , <i>aidb.vector_index_chord_vchordq_config</i> ,<br><i>aidb.vector_index_hsphere_optimized_config</i> , <i>aidb.vector_index_disabled_config</i> |