



AIDB

リリース 7.6.0

EnterpriseDB Corporation

2026 年 08 月 27 日

目次

第 1 章	Introduction	3
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
第 2 章	AIDB 7.x release notes	5
2.1	AIDB 7.6.0 release notes	5
2.2	AIDB 7.5.0 release notes	7
2.3	AIDB 7.4.0 release notes	8
2.4	AIDB 7.3.2 release notes	10
2.5	AIDB 7.3.0 release notes	10
2.5.1	非推奨のインターフェイス	11
2.6	AIDB 7.2.0 release notes	12
第 3 章	AIDB overview	15
3.1	AIDB の役割	15
3.2	スタンドアロンの拡張機能としての AIDB	16
3.3	主要な機能	16
3.3.1	Known issues	16
第 4 章	Compatibility	17
4.1	AIDB	17
4.1.1	サポートされているプラットフォーム	17
4.1.2	サポートされている Postgres バージョン	17
4.2	PGFS	17
第 5 章	Installing and configuring AIDB	19
5.1	次のステップ	19
5.1.1	Configuring AIDB	19
5.1.2	Installing AIDB	27
5.1.3	Threading limits	28
第 6 章	Integrating models	31
6.1	Local models	31
6.1.1	デフォルトモデル	32
6.1.2	モデルプロバイダー	33
6.1.3	カスタムモデル	33
6.1.4	作成時にモデルの検証	42
6.1.5	ローカルモデルのダウンロード	43
6.1.6	登録されたモデルを使用する	44
6.1.7	障害対応	44
6.2	External model connections	45

6.2.1	作成時にモデルの検証	45
6.2.2	OpenAI 互換エンドポイント	46
6.2.3	OpenAI Responses API	48
6.2.4	Anthropic Messages API	49
6.2.5	NVIDIA NIM	51
6.2.6	Google Gemini	53
6.2.7	OpenRouter	54
6.3	Support matrix	55
6.3.1	ローカルモデル	55
6.3.2	リモートモデル	59
第 7 章	Configuring external storage	63
7.1	その仕組み	63
7.2	ステップ 1 保存場所を作成する	63
7.2.1	S3 互換オブジェクトストア	64
7.2.2	ローカルファイルシステム	64
7.3	ステップ 2 ボリュームを作成する	65
7.3.1	パス処理	65
7.3.2	例 S3 の PDF	66
7.3.3	例ローカルディレクトリ内の画像	66
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	66
7.5	ボリュームの管理	66
7.5.1	ボリュームのリスト	66
7.5.2	ボリュームの削除	67
7.6	SQL を使用してボリュームコンテンツにアクセスする	67
7.6.1	ボリューム内のオブジェクトのリスト	67
7.6.2	単一オブジェクトの読み取り	68
7.6.3	オブジェクトの書き込み、上書き、および削除	68
第 8 章	Agents	69
8.1	例 text-to- SQL エージェント	69
8.1.1	セットアップ	69
8.1.2	会話	71
8.1.3	比較	71
8.2	ドキュメントマップ	72
8.2.1	Managing agents	72
8.2.2	Invoking agents	76
8.2.3	Delegation	79
第 9 章	Tools	81
9.1	ドキュメントマップ	82
9.1.1	Native tools	82
9.1.2	Custom SQL tools	87
9.1.3	MCP tools	89
第 10 章	AI pipelines	93
10.1	例 RAG のナレッジベース	93
10.1.1	パイプラインのセットアップ	93

10.1.2	パイプラインを実行する	94
10.1.3	Retrieve セマンティック検索	94
第 11 章	Knowledge bases	151
11.1	取得ファンクション	151
11.1.1	検索機能の流れ	151
11.1.2	promoteb.retrieve_text()	151
11.1.3	promoteb.retrieve_key()	152
11.2	高度なクエリー 中間ステップの結合	152
11.2.1	構文の例	152
11.3	重複排除と中間結果	153
第 12 章	Knowledge bases	155
12.1	Multi-pipeline knowledge bases	155
12.1.1	既存の KB へのパイプラインのアタッチ	155
12.1.2	互換性ルール	156
12.1.3	削除セマンティクス	157
12.1.4	接続されたパイプラインの検査	157
12.1.5	ベクターテーブルスキーマ	157
12.2	Hybrid search	158
12.2.1	前提条件	158
12.2.2	ボリュームソースの制限	159
12.3	Hybrid search	159
12.3.1	次のステップ	159
12.4	Examples	168
12.4.1	例 1 Postgres テーブルからの大量埋め込み	168
12.4.2	例 2 S3 バケット PGFS からの大量埋め込み	169
第 13 章	Semantic knowledge bases	173
13.1	セマンティック KB とベクターナレッジベース	173
13.2	セマンティック KB インデックスとは	173
13.3	セマンティック KB を作成および管理する	174
13.3.1	シングル KB のショートカット	174
13.4	KB を最新の状態に保つ	174
13.5	セマンティック KB の使用方法	175
13.6	このセクションでは	175
13.7	関連ドキュメント	175
13.7.1	Searching a semantic knowledge base	176
13.7.2	Semantic aliases	179
13.7.3	Text-to-SQL	182
13.7.4	Example	184
第 14 章	SQL functions	189
14.1	Text chunking	189
14.1.1	構成オプション	190
14.2	Data parsing	191
14.2.1	HTML を解析する	191
14.2.2	PDF を解析する	192

14.3	Performing OCR	193
14.4	Text summarization	196
14.4.1	quietb.summarize_text()	196
14.5	Text summarization	196
14.6	Text summarization	196
14.7	Text summarization	196
14.7.1	Summarization use cases	197
14.7.2	Summarization cookbook	201
14.8	AI inference functions	215
14.8.1	quietb.encode_text()	216
14.8.2	promoteb.encode_text_batch()	216
14.8.3	quietb.encode_image()	216
14.8.4	quietb.generate_text()	217
14.8.5	quietb.generate_text_batch()	218
14.8.6	quietb.inference_config()	218
14.9	Text reranking	219
14.9.1	ナレッジベースの取得後に再ランキングを使用する	220
第 15 章 Reference		221
15.1	Models reference	221
15.1.1	構成モデル	221
15.1.2	カタログビュー	222
15.1.3	モデル管理機能	223
15.1.4	HCP モデルファンクション	227
15.1.5	推論ファンクション	228
15.1.6	推論構成ヘルパー	232
15.1.7	モデル構成ヘルパー	236
15.2	Knowledge bases reference	247
15.2.1	カタログビュー	247
15.2.2	タイプ	248
15.2.3	ナレッジベース機能	249
15.2.4	ボリューム管理機能	252
15.3	Operations reference	254
15.3.1	エラーログファンクション	254
15.3.2	エラーカテゴリ	259
15.4	Functions reference	259
15.4.1	helpb.chunk_text	259
15.4.2	quietb.parse_html	260
15.4.3	quietb.parse_pdf	261
15.4.4	quietb.perform_ocr	262
15.4.5	quietb.summarize_text	263
15.4.6	quietb.summarize_text_aggregate	264
15.4.7	パラメーターのデフォルトのクイックリファレンス	265
15.5	Pipelines reference	265
15.6	Agents reference	266
15.6.1	カタログビュー	266
15.6.2	エージェント管理機能	268

15.6.3	会話機能	271
15.6.4	構造化出力ヘルパー	274
15.6.5	タイプ	275
15.7	Tools reference	276
15.7.1	カタログビュー	276
15.7.2	ツール管理機能	277
15.7.3	パラメーターヘルパー	281

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 Introduction

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. **** ナレッジベースの概要 **** — 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。デフォルトのローカルモデル設定は必要ありません。または `aidb.create_model()` に登録する外部 API モデルのいずれかです。パイプラインを作成する前にこれを行います。
4. **** Configuring external storage ** (optional)** ソースデータが Postgres テーブルではなく S3 またはローカルファイルシステムにある場合、パイプラインが読み取れるように PGFS を使用してボリュームを構成します。
5. **** Agents reference ** (optional)** — 複数のステップを推論し、**Tools reference** — AIDB 独自のオペレーション、独自の SQL クエリー、または外部 MCP サーバーからインポートされたツールを呼び出すデータベース内エージェントを構築します。
6. **** AI 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。
7. **** 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。
8. **** Semantic knowledge bases ** (optional)** — 自然言語スキーマ検索とテキストから SQL のために、スキーマのテーブル、ビュー、列、コメントをインデックスします。したがって、質問は適切な関係と再利用可能にマップされます。パラメーター化されたクエリー。

1.2 ドキュメントマップ

Section	What it covers
AIDB 7.2.0 release notes	What changed in each version
Compatibility	Supported platforms and Postgres versions for AIDB and PGFS

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>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>Agents reference</i>	Building in-database agents, invoking them, and delegation
<i>Tools reference</i>	The unified tool catalog: native tools, custom SQL tools, and MCP tools
<i>AI pipelines</i>	Defining pipelines, pipeline steps, orchestration, and examples
<i>Knowledge bases reference</i>	Semantic search, hybrid search, and vector index options
<i>Semantic knowledge bases</i>	Schema-metadata search and text-to-SQL: semantic KBs, semantic aliases, and agent tools
<i>SQL functions</i>	Standalone SQL functions for chunking, parsing, OCR, summarization, and embedding
<i>AI pipelines</i>	Full API reference for all AIDB functions and views

第2章 AIDB 7.x release notes

AIDB メジャーバージョン 7 のリリースノート。

Version	Release date
AIDB 7.6.0 release notes	26 Aug 2026
AIDB 7.5.0 release notes	29 Jul 2026
AIDB 7.4.0 release notes	06 Jul 2026
AIDB 7.3.2 release notes	17 Apr 2026
AIDB 7.3.0 release notes	08 Apr 2026
AIDB 7.2.0 release notes	30 Mar 2026

2.1 AIDB 7.6.0 release notes

リリース日 2026 年 8 月 26 日

AIDB 7.6.0 では、データベース内 AI エージェントを構築するための SQL ネイティブフレームワークである **エージェントとツール**を導入しています。エージェントは、ワンショットで回答するのではなく、ツールを呼び出し、その結果を観察し、繰り返すというマルチプルのステップを検討します。ツールは、大規模なビルトインネイティブツールセット、独自のカスタム SQL クエリー、および外部 MCP サーバーからインポートされたツールに及ぶ統合カタログ `aidb.tools` から取得します。エージェントは、コールごとの予算トークン、推論の反復、実測時間、他のエージェントへの委任、およびユーザーごとの会話の可視性をサポートしています。まずは、[Agents reference](#) および [Tools reference](#) を参照してください。

AIDB 7.6.0 には、次の機能強化とバグ修正が含まれています。

Type	Description
Enhancement	エージェントとツール 上記の新しい SQL ネイティブエージェントフレームワークと統合ツールハブ。 Agents reference および Tools reference を参照してください。
Enhancement	ネイティブツール呼び出しモデルアダプター 新しい <code>`openai_responses`</code> および <code>`anthropic_messages`</code> プロバイダーと <code>`openai_responses_azure`</code> および <code>`anthropic_messages_azure`</code> <code>`lanthropic_messages_bedrock`</code> ホスティングバリエーションは、 <i>OpenAI</i> の <i>Responses API</i> および <i>Anthropic</i> のメッセージ API にネイティブに接続し、本物の <code>`tools / tool_choice`</code> 要求フィールドを生成し、実際のツールを解析しますプロンプトインジェクションを介してツール呼び出しをシミュレートする代わりに、呼び出し応答。 <code>:ref: `External model connections<External model connections></code> を参照してください。

次のページに続く

表 2 – 前のページからの続き

Type	Description
Enhancement	環境変数からのモデル資格情報 <code>`aidb.create_model()`</code> は、新しい <code>`credentials_env`</code> 引数を受け入れ、資格情報を JSON としてインラインで渡すのではなく、モデル使用時にサーバーサイドの環境変数から読み取ることができます。
Enhancement	llama.cpp を介したローカルリランキング 新しい <code>`llamacpp_reranking`</code> プロバイダーは、GGUF ファイルからテキストをローカルに再ランク付けし、クロスエンコーダーモデルたとえば、 <code>bge-reranker-v2-m3`</code> と <code>Qwen3-Reranker`</code> デコーダー専用ファミリーの両方をサポートします。外部 API は必要ありません。 :ref: Local models<Local models> を参照してください。
Enhancement	llama.cpp を介したローカル OCR 新しい <code>`llamacpp_ocr`</code> プロバイダーは、事前に登録されたデフォルトモデル <code>`lightonocr-2-1b-Q8_0`</code> を使用して、GGUF ビジョンモデルからローカルに OCR を実行します。 Local models を参照してください。
Enhancement	<code>`decode_text`</code> から <code>`generate_text`</code> に名前変更しました。 <code>aidb.decode_text()</code> および <code>aidb.decode_text_batch()</code> の名前は <code>aidb.generate_text()</code> および <code>aidb.generate_text_batch()</code> に変更され、埋め込み側の <code>`encode_text()`</code> の名前付けと一致します。古い名前はまだ機能しますが、非推奨であり、将来のバージョンでは削除される予定です。 :ref: 推論ファンクション<推論ファンクション> を参照してください。
Enhancement	ソース全体のセマンティック KB 検索 新しい <code>`aidb.semantic_kb_search()`</code> ファンクションは、相互ランク融合を使用して、スキーマメタデータと厳選されたセマンティックエイリアスからのランク付けベクター検索結果を単一の結果セットに融合します。
Enhancement	簡素化されたセマンティック KB 呼び出し <code>kb_name`</code> および <code>`name`</code> は、 <code>aidb.get_column_definitions()</code> 、 <code>aidb.get_metadata()</code> 、 <code>aidb.get_entity_definitions()</code> 、 <code>aidb.search_by_comment()</code> 、 <code>aidb.create_semantic_kb()</code> 、 <code>aidb.delete_semantic_kb()</code> 、 <code>aidb.search_by_comment()</code> 、 <code>aidb.create_semantic_kb()</code> 、 <code>aidb.delete_semantic_kb()</code> 、 <code>aidb.refresh_semantic_kb()</code> 、 <code>aidb.update_semantic_kb_auto_processing()</code> 、および <code>aidb.semantic_kb_stats()`</code> でオプションになりました。省略すると、既存の AIDB セマンティックが解決されます。
Change	セマンティックエイリアスは複数の KB に属することができます。 <code>`aidb.create_semantic_alias()`</code> および関連ファンクションは <code>`model`</code> 引数をドロップします。エイリアスは、正確に 1 つの KB とモデルにバインドされるのではなく、SQL が読み取るスキーマを所有するナレッジベースごとに 1 回埋め込まれるようになりました。
Enhancement	パイプラインメトリックは、最後の実行時間を示します <code>`aidb.get_pipeline_metrics()`</code> とパイプラインメトリックビューには <code>`last_run_completed`</code> タイムスタンプが含まれるようになりました。

次のページに続く

表 2 – 前のページからの続き

Type	Description
Change	漏洩資格情報の検出 <code>aidb.create_model()</code> は、 <code>`api_key`</code> または <code>`basic_auth`</code> を直接埋め込む <code>`config`</code> を拒否するようになりました。代わりに <code>`credentials`</code> / <code>credentials_env`</code> を介してそれらを渡します。新しい <code>`aidb.audit_leaked_credentials()`</code> ファンクションは、構成に埋め込まれ資格情報が検出された既存のモデルをレポートします。
Bug fix	無効なヘッダー文字を含む資格情報 HTTP ヘッダーで有効でない文字たとえば、野良改行などを含む <code>`api_key`</code> または <code>`basic_auth`</code> 値は、バックエンドをクラッシュするのではなく、明確な構成エラーで失敗するようになりました。

2.2 AIDB 7.5.0 release notes

リリース日 2026 年 7 月 29 日

AIDB 7.5.0 には、次の機能強化とバグ修正が含まれています。

Type	Description
Enhancement	作成時のモデルの検証 <code>aidb.create_model()</code> は、デフォルトでモデルを検証し、小さなプローブ推論ローカルモデルのダウンロードとロード、または外部プロバイダーへのテスト要求の送信を実行するため、構成と資格情報の問題は、作成時ではなく作成時に表面化します初めての使用。 <code>`validate` => <i>false</i></code> を渡してスキップするか、新しい <code>`aidb.validate_model()`</code> を使用して既存のモデルを確認します。 :ref: `Integrating models`<Integrating models> を参照してください。
Enhancement	llama.cpp を使用したローカル LLM 推論 AIDB は、既存の Candle ベースのローカルモデルと並んで、新しい <code>`llamacpp_generate`</code> および <code>`llamacpp_embeddings`</code> プロバイダーを介して GGUF 形式のモデルをローカルで実行できるようになりました。7つの事前登録されたデフォルトモデル埋め込み用に5つ、生成用に2つが同梱されます。 Local models を参照してください。
Enhancement	失敗したパイプラインレコードの再キューイング 新しい <code>`aidb.requeue_pipeline_errors()`</code> ファンクションは、ソースデータに触れる必要がなく、パイプラインの次回の実行で再処理のためにレコードレベルのエラーログエントリをダーティにマークします。 失敗したレコードの再キューイング を参照してください。
Enhancement	ライブ調整可能なモデルスレッド制限 新しい <code>`aidb.max_threads`</code> パラメーターは、ローカルモデル推論 Candle および llama.cpp に使用される CPU スレッドをキャップし、再起動せずにライブで変更できます。 スレッド制限 を参照してください。
Enhancement	I/O スレッドプール制御 新しい <code>`aidb.max_io_threads`</code> パラメーターは、リモートモデルの HTTP 呼び出しとボリューム/オブジェクトストアアクセスに使用される非同期 I/O スレッドプールをキャップし、再起動なしでライブ調整できます。 スレッド制限 を参照してください。

次のページに続く

表 3 – 前のページからの続き

Type	Description
Enhancement	llama.cpp ネイティブログ制御 新しい <code>`aidb.enable_llamacpp_logs`</code> パラメーターは、モデルのロードと推論の問題のトラブルシューティングのために、サーバーログに llama.cpp 独自の診断ログをオンにします。デフォルトではオフ。有効にするには再起動が必要です。 AI pipelines を参照してください。
Enhancement	パイプラインカラム/ステップタイプの不一致の明確なエラー <code>`aidb.create_pipeline()`</code> は、SQL タイプが構成済みステップたとえば、 <code>`KnowledgeBase(Image)`</code> ステップをフィードする <code>`text`</code> 列と一致しないソース列を、作成時に特定のエラーで拒否するようになりました。後で不透明な内部エラーで失敗する代わりに。
Change	メトリックコレクターがオプトインになりました。 パイプラインメトリックを <code>`aidb.pipeline_metrics_collection`</code> にマテリアライズするバックグラウンドワーカーは、デフォルトでは無効になっています。新しい <code>`aidb.enable_background_metrics_collector`</code> 設定を使用して、インスタンス全体またはデータベースごとに有効にします。 マテリアライズされたメトリック を参照してください。
Change	<code>`topk`</code> は正の整数である必要があります。 <code>`aidb.retrieve_key()`</code> および <code>`aidb.retrieve_text()`</code> は、サイレントに 1 にクランプするのではなく、明確なエラーで 1 未満の <code>`topk`</code> 値を拒否するようになりました。
Bug fix	エラーログファンクションと空のソースパイプライン <code>aidb.get_error_logs()</code> 、 <code>aidb.get_error_log_summary()</code> 、および <code>`aidb.get_all_error_summaries()`</code> は、空のソースセマンティック KB などを使用したパイプラインのエラーを発生させなくなりました。それらは代わりに空の結果を返し、他の結果をアボートしなくなりました同じ呼び出し内のパイプライン。

2.3 AIDB 7.4.0 release notes

リリース日 2026 年 7 月 6 日

AIDB 7.4.0 には、次の機能強化とバグ修正が含まれています。

Type	Description
Enhancement	クエリ可能外部テーブルとしてのボリューム 各 AIDB ボリュームは PGFS バックアップの外部テーブルとして公開されるため、標準の SQL <code>INSERT</code> 、 <code>UPDATE</code> 、 <code>DELETE</code> を含むで含まれるオブジェクトをリスト、読み取り、書き込み、および削除できます。オブジェクトに対して、および <code>`SELECT body FROM my_volume WHERE key = '...`</code> を使用して単一のオブジェクトを取得します。 :ref: `Configuring external storage<Configuring external storage>` を参照してください。

次のページに続く

表 4 – 前のページからの続き

Type	Description
Enhancement	出力ホワイトリスト モデル呼び出しとモデルのダウンロードのオプションのアウトバウンド HTTP ホワイトリスト。`aidb.egress_allowlist` パラメーター共有フォールバックとして `edb.egress_allowlist` を使用して構成します。ホストおよび CIDR エントリを受け入れます。設定されていない場合、チェックは適用されません。
Enhancement	より信頼性の高い HuggingFace モデルのダウンロード ダウンロードは、不安定な接続を介して再開され、サイズと SHA-256 を検証し、進行状況通知を発行し、再試行の間に Ctrl-C に応答するようになりました。破損したキャッシュされたスナップショットは、次のロード時に自己修復します。
Enhancement	厳密なパイプラインオプションの検証 <code>step_N_options</code> 、ナレッジベース構成、およびスタンドアロンヘルパー `chunk_text`、 <code>summarize_text</code> 、 <code>parse_html</code> 、 <code>parse_pdf</code> 、 <code>pdf_to_image</code> 、`perform_ocr` の不明なキーと間違ったタイプは、サイレントドロップされる代わりに、明確なエラーを発生させるようになりました。
Enhancement	明確なモデルエラーメッセージ モデルプロバイダーからの HTTP エラーに要求 URL が含まれるようになりました。大きな入力値は、エラー出力で切り捨てられます。OCR 応答は、HTTP 200 で JSON としてのみ解析されます。
Enhancement	バッチのサマライズ `SummarizeText` パイプラインステップは、レート制限エラーに対して入力ごとにリトライを使用して、バッチでレコードを処理するようになりました。
Enhancement	パイプライン作成時のパーミッションエラーの明確化 `SELECT` 特権なしでソーステーブルに対してパイプラインを作成すると、不透明な内部障害の代わりに明示的な「permission denied」エラーが返されるようになりました。
Bug fix	空の取得は空の結果を返します 空のナレッジベースの `retrieve_text` は、標準の SQL `SELECT` 動作と一致して、エラーを生成する代わりに警告を含む行を結果ずに結果ます。
Bug fix	非テキストソースの `retrieve_text` 値列 元のソースが非テキスト PDF や画像などのパイプラインからの取得は、テキストとしてデコードされた生のバイトの代わりに `NULL` を `value` で返すようになりました。一致するメタデータは引き続き返されます。
Bug fix	メトリックコレクターでのデッドロック バックグラウンドメトリックコレクターは、単一のトランザクションですべてのユーザーテーブルのロックを保持しなくなり、`delete_pipeline` に対するデッドロックを削除します。
Bug fix	BIGINT パイプラインキー `int4` 範囲を超える `BIGINT` 主キーを持つソーステーブルのパイプラインは、「out of range for type integer」エラーで失敗しなくなりました。
Bug fix	ステップ全体で尊重されるパイプラインのバッチサイズ レコードチャンクなどを爆発させるステップは、次のステップに渡す前に出力をサブバッチ化し、構成済みの `batch_size` 内でメモリと並列性を維持します。
Bug fix	OCR 構成の検証 OCR モデル構成は、モデル作成時に検証されるようになりました。

2.4 AIDB 7.3.2 release notes

リリース日 2026 年 4 月 17 日

AIDB 7.3.2 には、次の機能強化とバグ修正が含まれています。

Type	Description
Enhancement	パイプライン DLQ - フェーズ 3 メトリックとエラーログ API メトリックとエラーログ API でデッドレターキューを拡張し、パイプラインごとのエラー数、エラーの詳細、履歴エラーログへのプログラムによるアクセスを提供します。
Enhancement	マルチユーザーパイプライン PGD トリガーラッパー PGD パイプラインのマルチユーザー対応トリガーラッパーを追加し、パイプライン自動処理トリガーがマルチユーザー PGD 環境で正しく動作することを保証します。
Enhancement	PGD レプリケーションセットアップのナレッジベースステابل 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	PDF 解析の向上 `lopdf` ライブラリを、構造 PDF 解析が向上した新しいバージョンに更新しました。その結果、複雑で不正な形式の PDF ドキュメントからのテキスト抽出が向上しました。
Enhancement	HTTP 429 再試行処理 AIDB は、モデルプロバイダー API から HTTP 429 Too Many Requests 応答を受信した要求を自動的に遅延および再試行するようになり、API レート制限に到達した場合の復元力を向上させます。
Enhancement	ソースキーによる取得結果の重複除去 Retrieve 操作はソースキーに基づいて結果の重複を排除するようになり、同じソースドキュメントが複数回インデックスされている場合、または複数のパイプライン実行にわたって参照されている場合に、取得出力に重複レコードが表示されるのを防ぎます。
Bug fix	間違った入力タイプエラーの処理を修正します。 入力タイプ検証エラーは、予期しないエラーとして下流に伝播するのではなく、入力ステージで正しくキャッチおよび表示されます。
Bug fix	マルチパイプライン ID のバグを修正します。 マルチパイプライン構成でパイプライン ID が誤って割り当てられる問題を修正しました。これにより、パイプラインが誤って識別されるか、不正に実行される可能性があります。
Bug fix	自動処理状態のバグを修正する パイプラインが誤った処理状態になるか、その状態に留まる可能性があるパイプライン自動処理システムの状態管理の問題を解決しました。

2.5 AIDB 7.3.0 release notes

リリース日 2026 年 4 月 8 日

AIDB 7.3.0 には、次の機能強化とバグ修正が含まれています。

Type	Description
Enhancement	パイプライン DLQ – フェーズ 2 ノンブロッキングエラー Dead Letter Queue はノンブロッキングエラーを処理するようになりました。これは、一時的なまたは非クリティカルな処理障害がパイプラインの実行を停止しないことを意味します。失敗したレコードは、検査と再処理のためにキューに入れられます。
Enhancement	一時的なエラー分類 モデルエラーとデータベースエラーが必要に応じて `PipelineTemporary` として分類されるようになり、パイプラインが一時的な障害を永続的なエラーとして扱うのではなく、自動的に再試行できます。
Enhancement	DLQ ユーザ API 失敗したパイプラインレコードをリスト、検査、および再処理する機能を含む、デッドレターキューと対話するためのユーザー対応 SQL ファクションを導入しました。
Enhancement	セマンティックナレッジベース検索のコメント セマンティックナレッジベース検索には、ドキュメントコンテンツとともに保存されているコメントの検索のサポートが含まれるようになりました。

2.5.1 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 ビューとファクションを削除します。それぞれに置換えが記載されています。

Deprecated objects	Replacement
Views	
<i>aidb.preparers</i>	<i>aidb.pipelines</i> (filter by operation type if needed)
<i>aidb.preps</i>	<i>aidb.pipelines</i>
<i>aidb.retrievers</i>	<i>aidb.knowledge_bases</i> or <i>aidb.kbs</i>
<i>aidb.preparer_registry</i>	<i>aidb.pipeline_registry</i> (internal)
<i>aidb.preparer_registry_source_table</i>	<i>aidb.pipeline_registry</i> (internal)
<i>aidb.preparer_registry_source_volume</i>	<i>aidb.pipeline_registry</i> (internal)
Functions	
<i>aidb.create_table_knowledge_base(...)</i>	<i>aidb.create_pipeline(... step_1 => 'KnowledgeBase')</i>
<i>aidb.create_volume_knowledge_base(...)</i>	<i>aidb.create_pipeline(... step_1 => 'KnowledgeBase')</i>
<i>aidb.set_auto_knowledge_base(...)</i>	<i>aidb.update_pipeline(name, auto_processing => ...)</i>
<i>aidb.assert_knowledge_base_preconditions(...)</i>	Removed — no-op; validation is now handled internally
<i>aidb.create_retriever_for_table(...)</i>	<i>aidb.create_pipeline(... step_1 => 'KnowledgeBase')</i>
<i>aidb.create_retriever_for_volume(...)</i>	<i>aidb.create_pipeline(... step_1 => 'KnowledgeBase')</i>
<i>aidb.delete_retriever(...)</i>	<i>aidb.delete_pipeline(...)</i>
<i>aidb.set_retriever_auto_processing(...)</i>	<i>aidb.update_pipeline(name, auto_processing => ...)</i>
<i>aidb.enable_auto_embedding_for_table(...)</i>	<i>aidb.update_pipeline(name, auto_processing => 'Live')</i>
<i>aidb.disable_auto_embedding_for_table(...)</i>	<i>aidb.update_pipeline(name, auto_processing => 'Disabled')</i>

次のページに続く

表 7 – 前のページからの続き

Deprecated objects	Replacement
<code>aidb.create_table_preparer(...)</code>	<code>aidb.create_pipeline(...)</code>
<code>aidb.create_volume_preparer(...)</code>	<code>aidb.create_pipeline(...)</code>
<code>aidb.create_preparer_for_table(...)</code>	<code>aidb.create_pipeline(...)</code>
<code>aidb.create_preparer_for_volume(...)</code>	<code>aidb.create_pipeline(...)</code>
<code>aidb.delete_preparer(...)</code>	<code>aidb.delete_pipeline(...)</code>
<code>aidb.set_preparer_auto_processing(...)</code>	<code>aidb.update_pipeline(name, auto_processing => ...)</code>
<code>aidb.get_preparer_data_source(...)</code>	Query <code>aidb.pipelines</code> view directly
<code>aidb.assert_preparer_preconditions(...)</code>	Removed — validation is now handled internally
<code>aidb.assert_table_preparer_preconditions(...)</code>	Removed
<code>aidb.assert_volume_preparer_preconditions(...)</code>	Removed
<code>aidb.bulk_embedding(name)</code>	<code>aidb.run_pipeline(name)</code>
<code>aidb.bulk_data_preparation(name)</code>	<code>aidb.run_pipeline(name)</code>
<code>aidb.single_embedding(name, key, text)</code>	<code>aidb.run_pipeline(name)</code> or direct model calls
<code>aidb.single_embedding(name, key, bytea)</code>	<code>aidb.run_pipeline(name)</code> or direct model calls
<code>aidb.set_auto_preparer(name, mode)</code>	<code>aidb.update_pipeline(name, auto_processing => mode)</code>
<code>aidb.upsert_chunk_text(name, key, value)</code>	Use pipeline execution via <code>aidb.run_pipeline()</code>
<code>aidb.upsert_summarize_text(name, key, value)</code>	Use pipeline execution via <code>aidb.run_pipeline()</code>
<code>aidb.upsert_parse_html(name, key, value)</code>	Use pipeline execution via <code>aidb.run_pipeline()</code>
<code>aidb.upsert_parse_pdf(name, key, value)</code>	Use pipeline execution via <code>aidb.run_pipeline()</code>
<code>aidb.upsert_perform_ocr(name, key, value)</code>	Use pipeline execution via <code>aidb.run_pipeline()</code>

2.6 AIDB 7.2.0 release notes

リリース日 2026 年 3 月 30 日

AIDB 7.2.0 には、次の機能強化とバグ修正が含まれています。

Type	Description
Enhancement	マルチパイプラインのナレッジベース ナレッジベースをマルチプルのパイプラインに関連付けできるようになり、より豊富で柔軟なデータの取り込みおよび取得アーキテクチャが可能になります。
Enhancement	ハイブリッド検索 BYOSQL 持込 SQL BYOSQL インターフェイスを使用した階層 1 ハイブリッド検索サポートを導入し、カスタム取得ロジックをセマンティック検索と組み合わせることができます。
Enhancement	マルチステップパイプラインの取得サポート マルチステップパイプライン構成内で取得操作がサポートされるようになり、パイプラインステージ全体でエンドツーエンドの RAG ワークフローが有効になります。
Enhancement	HuggingFace API のサポート エンベディングおよびその他の AI タスクのモデルプロバイダーとして、HuggingFace Inference API のサポートを追加しました。

次のページに続く

表 8 – 前のページからの続き

Type	Description
Enhancement	モデルオプション モデル構成にプロバイダー固有のオプションを含めることができるようになり、温度、最大トークン、その他のパラメーターなどのモデルの動作をより詳細に制御できます。
Enhancement	パイプラインデッドレターキュー DLQ – フェーズ 1: パイプラインのデッドレターキューサポートの最初のフェーズが導入されました。失敗したパイプライン操作の処理エラーがログに記録されるようになり、失敗したレコードの検査と再処理が可能になります。
Enhancement	構成可能な CPU 制限 バックグラウンドパイプラインワーカーの構成可能な CPU 制限を追加して、共有環境でのリソース競合を防ぎます。
Enhancement	HCP モデル同期ヒント HCP 同期が必要なモデルをリストする場合、AIDB は HCP からモデルを同期する方法をユーザーにガイドする役立つヒントを表示します。
Enhancement	オプションバージョン付けインターフェイス OVI パイプラインビューとナレッジベースビューがオプションバージョン付けインターフェイスをサポートするようになり、クライアントは特定の API バージョンにピン留めできます。
Enhancement	非推奨 UX の向上 非推奨機能は、代替機能と移行手順に関するガイダンスを含む明確なメッセージを提供するようになりました。
Enhancement	`aidb.models` ビューの `functions` タグ `aidb.models` ビューには `functions` タグが含まれ、ファンクション呼び出し機能をサポートするモデルを特定しやすくなりました。
Bug fix	ディレクトリを使用したボリュームリストを修正します ボリューム内のファイルと一緒にディレクトリが存在する場合、ボリュームリストが失敗する問題を修正しました。
Change	引用符で囲まれた識別子を含むボリューム名がブロックされる: SQL 参照のあいまいさを防ぐため、引用符で囲まれた識別子を必要とする名前たとえば、スペースまたは特殊文字を含む名前で作成することはできなくなりました。

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 統合を含みます。

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、ナレッジベース、エージェント、および ツール の 4 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。

**** *Agents reference* **** モデルにワンショットで回答するのではなく、複数のステップを推論させます。エージェントはツールを呼び出し、その結果を観察し、回答が得られるまで繰り返します。エージェントは、要求の一部を別のエージェントに委任することもできます。

**** *Tools reference* **** は、エージェントがアクションを実行するために呼び出すものです - AIDB 独自のビルトインオペレーションパイプラインのステータスの照会、ナレッジベースの検索、SQL クエリーの実行、独自のカスタム SQL クエリー、または外部 MCP からインポートされたオペレーションサーバー。

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、Agent Factory または Hybrid ManagerHM への実行時依存はありません。他の Postgres 拡張機能と同じようにインストールし、`postgresql.conf` を介して構成し、SQL を介して対話します。

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

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。または、データベースにシークレットを保存するのをまったく回避するには、代わりにそれを読み取る環境変数を名前付けます。[環境変数からの資格情報](#) を参照してください。

データソースのサポート—パイプラインは、標準の Postgres テーブルから、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 [Configuring external storage](#) 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。

3.3.1 Known issues

- **限定的なエアギャップサポート。** ビルトイン CPU モデルランタイム Candle および llama.cpp は、モデルファイルがローカルで利用できるようになると完全にオフラインで実行されますが、リモートモデルを呼び出すパイプラインステップまたはエージェントツールプロバイダーはまだそのプロバイダーへのアウトバウンドネットワークアクセスを必要としています。[Configuring AIDB](#) を参照してください。

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	ppc64le
Ubuntu 22.04 LTS	OK	—	—
Ubuntu 24.04 LTS	OK	—	—
Debian 12 (Bookworm)	OK	OK	—
RHEL 8	OK	—	—
RHEL 9	OK	OK	OK (Postgres 18 only)

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](#)
- スレッド制限

5.1 次のステップ

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

5.1.1 Configuring AIDB

AIDB を使用するには、`shared_preload_libraries` に追加し、データベースに拡張機能を作成し、ユーザーアクセスを管理し、インストールを確認します。このページでは、プロキシ設定と、後で調整できる AIDB 構成パラメーター GUC についても説明します。

`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
```

!!!note `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。

aidb_users ロールを使用してユーザーアクセスを管理する

AIDB 拡張機能を作成すると、Postgres は `aidb_users` という名前のロールを自動的に作成します。このロールのユーザーメンバーシップを付与して、AIDB の SQL ファンクションへのアクセスを付与します。後で拡張機能を削除しても、このロールは削除されません。

`aidb_users` は、AIDB 機能のアクセス管理を簡素化するように設計されています。

- **目的** ユーザーに `aidb_users` ロールを付与すると、すべての AIDB 機能にアクセスできるため、拡張機能の機能を使用できます。
- **セキュリティ** `aidb_users` ロールは AIDB の内部テーブルへのアクセスを付与しないため、拡張機能のパブリック API のみがそれを保持しているユーザーに公開されます。
- **バックグラウンドワーカー** バックグラウンドワーカーが使用される場合、バックグラウンドワーカーは、定義したユーザーとして、またはユーザーが指定したロールとして実行されます。そのユーザーがそのロールにアクセスできる場合。

ユーザーに AIDB 機能を使用させるには、`aidb_users` ロールを付与します。

```
GRANT aidb_users TO your_user;
```

たとえば、最初に特権またはロールメンバーシップのない `alice` という名前のロールを作成します。

```
CREATE ROLE alice
LOGIN
PASSWORD change_me;  -- replace with a real password
```

次に、`alice` を `aidb_users` ロールに追加します。

```
GRANT aidb_users TO alice;
```

`alice` は、`aidb_users` が持っている特権、たとえば、`aidb` スキーマのルーチンに対する `EXECUTE` を継承するようになりました。

`aidb_users` のメンバーシップは、自分のデータへのアクセスを付与しません。 `alice aidb_users` ではないに、AIDB が読み取りまたは書き込みをするテーブルに必要な特権を付与します。この例では、通常はパイプラインで使用されるソーステーブルである 3 つの任意のテーブルへのアクセスを付与します。

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON public.documents,
    public.email_text,
    public.chats
TO alice;
```

alice も public スキーマにオブジェクトを作成する必要がある場合

```
GRANT CREATE ON SCHEMA public TO alice;
```

!!!note パイプラインデザイナーを介して AIDB を使用するハイブリッドマネージャーのパイプラインデザイナーを介して AIDB にアクセスする場合、パイプライン操作は別のロール `visual_pipeline_user` の下で実行されます。これは、`aidb_users` のメンバーシップを付与する必要があります。その設定については、[VPU and permissions](#) 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 AIDB のエアギャップサポートは制限されています。ビルトイン CPU モデルランタイム Candle および llama.cpp は、モデルファイルがローカルで利用できるようになると、完全にオフラインで実行されますが、リモートモデルプロバイダーを呼び出すパイプラインステップまたはエージェントツールは、そのプロバイダーへのアウトバウンドネットワークアクセスが必要です。

構成パラメーター

AIDB は、`aidb.*` 名前空間で Postgres 構成パラメーター GUC のセットを公開します。`postgresql.conf` で設定し、SET とセッションごとに、または `-c` と接続ごとに設定します。以下のパラメーターのほとんどは `USERSET` です。ロールは自分のセッションでそれらを変更できます。異なるコンテキストを持つパラメーターは個別に呼び出されます。

スレッド制限

Parameter	Type	Default	Range	Description
<i>aidb.max_threads</i>	integer	0	0-1024	ローカルモデルの推論 Candle および llama.cpp に使用される CPU スレッドの最大数。0 は、デフォルトの使用可能な CPU の半分を使用します。`SUSET`.
<i>aidb.max_io_threads</i>	integer	0	0-1024	AIDB の非同期 I/O ランタイムリモートモデル HTTP 呼び出し、およびボリューム/オブジェクトストア I/O の最大スレッド数。0 は、`TOKIO_WORKER_THREADS` 環境変数、または Tokio 独自のデフォルト CPU ごとに 1 つが設定されていない場合は、それにフォールバックします。`SUSET`.

どちらもライブで適用されます。セッションローカル SET は、そのバックエンドでのみすぐに有効になります。ALTER SYSTEM または postgresql.conf での設定は、pg_reload_conf() または SIGHUP の後に有効になり、すべてのバックエンドとバックグラウンドワーカーに伝播します。再起動は必要ありません。

promoteb.max_threads

```
SET aidb.max_threads = 4;
```

quietb.max_io_threads

```
SET aidb.max_io_threads = 4;
```

これら 2 つのパラメーターが制御するもの、相互作用、およびそれらを下げる場合については、[スレッド制限](#) を参照してください。

モデルのダウンロード

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

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

`helpb.download_log_level`

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

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

例

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

`quietb.download_max_attempts`

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

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

例

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

llama.cpp ネイティブログ

Parameter	Type	Default	Range	Description
<code>aidb.enable_llamacpp_</code>	boolean	<code>false</code>	—	llama.cpp 自身の診断出力 GGUF メタデータ、テンソルシェイプ、およびバックエンド/モデルの初期化をサーバーログに書き込むかどうかを制御します。SUSET. 拡張機能の初期化時に 1 回読み取ります。以下を参照してください。

agentb.enable_llamacpp_logs

デフォルトでオフ llama.cpp のネイティブログギングは冗長であるため、モデルロードごとに数百行ため、必要な場合を除き、抑制されます。これは、AIDB 自分自身のダウンロード進行状況メッセージを制御する上記の `aidb.download_log_level` とは別です。2 つは独立しています。

このページの他のパラメーターとは異なり、`aidb.enable_llamacpp_logs` は SUSET スーパーユーザー、またはパラメーターで SET が付与されたロールは変更できます。拡張機能の初期化時に 1 回読み取られます。SET、ALTER SYSTEM で設定するか、`postgresql.conf` の編集とリロード `pg_reload_conf()` は、Postgres が再起動されるまで有効になりません。

```
aidb.enable_llamacpp_logs = true
```

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

`llamacpp_generate` および `llamacpp_embeddings` モデルにのみ適用されます。Candle ベースのモデル `bert_local`、`clip_local`、`t5_local`、`llama_instruct_local` には影響しません。これを使用する場合については、[Local models — Troubleshooting](#) を参照してください。

ネットワーク下り

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

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

エントリーフォーマット

コンマ区切りの各エントリーは、次のいずれかです。

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

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

解決順序

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

例

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

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

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

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

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

`aidb.create_model(credentials_env => ...)` および `aidb.import_mcp_tools(headers_env => ...)` は、資格情報をデータベースに保存する代わりに、呼び出し者が名前付けた Postgres バックエンドプロセス環境変数を読み取ります。次のパラメーターは、この方法で参照できる変数名を制限します。

Parameter	Type	Default	Description
<code>aidb.env_var_allowed_prefix</code>	string	<code>AIDB_</code>	<code>credentials_env/headers_env</code> を介して読み取られた環境変数名の必須接頭辞。制限を無効にするには、空の文字列に設定します。スーパーユーザーのみが変更できます。`\$USER`。

呼び出し元制御のプロバイダーまたは MCP サーバー URL と組み合わせると、無制限の変数名は、悪意のある構成に Postgres プロセスの環境に存在する無関係のシークレットを読み取らせる可能性があります。このパラメータは、展開ごとにオーバーライド可能なままで、デフォルトでそれを閉じます。`credentials_env` については [環境変数からの資格情報](#) を、`headers_env` については [MCP tools](#) を参照してください。

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	7.6.0	aidb	aidb: makes it easy to build AI applications with postgres
pgfs	3.3.0	pgfs	pgfs: enables access to filesystem-like storage locations
vector	0.8.0	public	vector data type and ivfflat and hnsw access methods

5.1.2 Installing AIDB

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

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

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

-

PostgreSQL product compatibility

-

EDB Postgres Advanced Server product compatibility

-

EDB Postgres Extended Server product compatibility

前提条件

インストールプロセスを開始する前に

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

- [Installing PostgreSQL](#)

-

Installing EDB Postgres Advanced Server

-

Installing EDB Postgres Extended Server

3. pgvector 拡張機能をインストールします。

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

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

パッケージをインストールする

Package install コマンドの構文は次のとおりです。

```
# For Debian/Ubuntu:

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

# For Linux:

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

そこで

- <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。

5.1.3 Threading limits

AIDB は、ローカルモデル推論とネットワーク/ボリューム I/O を内部スレッドプールで実行します。デフォルトでは、ホストで使用可能な CPU の数から自分自身のサイズを調整します。他のワークロードと共有するデータベースサーバー、またはリソースに制限のある小さなコンテナなどでは、バインドされていないスレッドプールが与えたいよりも多くの CPU を消費する可能性があります。

2つのパラメーターは、制限する必要がある作業の種類に応じて、これらのスレッドプールを独立してキャップします。

ローカルモデル推論スレッド

`promoteb.max_threads` は、AIDB がローカルモデルの推論に使用する CPU スレッドプールをキャップします—両方のランタイム `Candle bert_local`、`clip_local`、`t5_local`、`llama_instruct_local` および `llama.cpp llama.cpp_generate`、`llamacpp_embeddings`。ローカルプロバイダーの完全なリストについては、[Local models](#) を参照してください。

一般にスレッドが多いほど、推論が速くなりますが、CPU 使用率が高くなります。多くの同時パイプラインを実行しているホスト、または CPU 割り当てが小さいホストでは、無制限のプールは他のバックエンドの CPU 時間を枯渇させる可能性があります。

デフォルトの 0 は、使用可能な CPU の半分を使用します。キャップするように特定の数値を設定します。

```
SET aidb.max_threads = 4;
```

非同期 I/O スレッド

`quietb.max_io_threads` は、AIDB の非同期 I/O スレッドプールをキャップします。HTTP OpenAI 互換エンドポイント、NVIDIA NIM、Gemini、および OpenRouter を介してリモートモデルプロバイダーを呼び出すために使用され、PGFS バックিংボリュームに対するボリューム/オブジェクトストア I/O に使用されます。

これは、ローカルモデル推論とは別のリソースです。CPU バウンドの計算ではなく、ネットワークとファイル I/O の同時実行性を制御します。多くの同時外部 API 呼び出しを行うパイプライン、または一度に多くのオブジェクトに触れるボリュームスキャンは、制限されたネットワーク帯域幅またはファイル記述子を持つホストで必要なより多くの同時 I/O を開く可能性があります。

デフォルトの 0 は、`TOKIO_WORKER_THREADS` 環境変数が設定されている場合、それ以外の場合は、CPU ごとに 1 つのスレッド、基になる Tokio ランタイムの独自のデフォルトにフォールバックします。キャップするように特定の数値を設定します。

```
SET aidb.max_io_threads = 4;
```

変更を適用する方法

どちらのパラメーターも、同じ 3 つの設定方法を受け入れます。

- **セッションごと**、SET を使用するとそのバックエンドにのみにすぐに適用されます。ロールアウトする前に制限をテストするのに役立ちます。
- **接続ごと**、接続文字列または `PGOPTIONS` で `-c aidb.max_threads=4` を使用します。
- **インスタンス全体**、`ALTER SYSTEM SET aidb.max_threads = 4;` の後に `SELECT pg_reload_conf();` を使用するか、`postgresql.conf` を直接編集してリロード (SIGHUP)。これは、再起動せずにすべてのバックエンドとバックグラウンドワーカーに伝播します。

例、共有ホストでインスタンス全体の両方をキャップするには

```
ALTER SYSTEM SET aidb.max_threads = 4;
ALTER SYSTEM SET aidb.max_io_threads = 4;
SELECT pg_reload_conf();
```

パラメーターリファレンスタイプ、デフォルト、範囲については、[スレッド制限](#) を参照してください。

第6章 Integrating models

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

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

Page	What it covers
<i>Local models</i>	Default local models (BERT, CLIP, T5, Llama), model providers, registering custom local models, and llama.cpp GGUF models
<i>External model connections</i>	OpenAI-compatible APIs, the native OpenAI Responses and Anthropic Messages APIs, NVIDIA NIM, Google Gemini, and OpenRouter
<i>Support matrix</i>	Pre-vetted local model variants and supported NVIDIA NIM models

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。

- **デフォルトモデル** — 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構成は必要ありませんが、モデルのファイルは、初めて使用するときに HuggingFace からダウンロードされます。[ローカルモデルのダウンロード](#) を参照してください。
- **カスタムモデル** 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。

ローカルモデルは、2 つの推論ランタイム Candle または llama.cpp のいずれかで実行されます。日常の使用法 `aidb.create_model()`、`aidb.encode_text()`、`aidb.generate_text()` など - いずれの方法でも同じです。以下の各モデルのプロバイダーは、使用するランタイムをノートするだけです。

6.1.1 デフォルトモデル

次のモデルは、構成なしですべての AIDB インストールで使用できます。プロバイダー列の意味については、[モデルプロバイダー](#) を参照してください。

Model name	Provider	Use
<i>bert</i>	<i>bert_local</i> (Candle)	Text embeddings
<i>clip</i>	<i>clip_local</i> (Candle)	Multimodal embeddings (text and image)
<i>t5</i>	<i>t5_local</i> (Candle)	Text-to-text generation
<i>llama</i>	<i>llama_instruct_local</i> (Candle)	Text-to-text generation
<i>bge-small-en-v1.5-f16</i>	<i>llamacpp_embeddings</i> (llama.cpp)	Text embeddings
<i>nomic-embed-text-v1.5-Q8_0</i>	<i>llamacpp_embeddings</i> (llama.cpp)	Text embeddings
<i>bge-m3-f16</i>	<i>llamacpp_embeddings</i> (llama.cpp)	Text embeddings
<i>qwen3-embedding-0.6b-Q8_0</i>	<i>llamacpp_embeddings</i> (llama.cpp)	Text embeddings
<i>qwen3-embedding-4b-Q8_0</i>	<i>llamacpp_embeddings</i> (llama.cpp)	Text embeddings
<i>qwen3.5-0.8b-Q8_0</i>	<i>llamacpp_generate</i> (llama.cpp)	Text-to-text generation
<i>llama-3.2-1b-instruct-Q8_0</i>	<i>llamacpp_generate</i> (llama.cpp)	Text-to-text generation
<i>lightocr-2-1b-Q8_0</i>	<i>llamacpp_ocr</i> (llama.cpp)	OCR (text extraction from images)
<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、名前直接使用します。

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

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

- - Text generation with the default T5 model
SELECT aidb.generate_text(t5, Translate to French: Hello, world.);

- - Text embedding with a default llama.cpp embedding model
SELECT aidb.encode_text(bge-small-en-v1.5-f16, The quick brown fox);

- - Text generation with a default llama.cpp model
SELECT aidb.generate_text(qwen3.5-0.8b-Q8_0, Translate to French: Hello, world.);
```

6.1.2 モデルプロバイダー

provider は、モデルにどの推論エンジンを使用するか `aidb.create_model()` に指示します。これは SQL の列挙型ではありません。これは、AIDB のモデルレジストリに登録されている Postgres 外部サーバーの名前であり、モデル作成時にアダプターと照合されます。インストールに登録されているプロバイダーとその説明をリストします。

```
SELECT server_name, server_description FROM aidb.model_providers ORDER BY server_name;
```

AIDB のローカルプロバイダー、それぞれの背後にあるランタイム、およびそれらの目的

Provider	Runtime	Description
<i>bert_local</i>	Candle	シンプルな言語モデル、テキストのエンコードに最適です。
<i>clip_local</i>	Candle	ビジョン/テキストモデル、テキストと画像のエンコードに最適です。
<i>t5_local</i>	Candle	翻訳、要約、質問回答に最適です。
<i>llama_instruct_local</i>	Candle	Meta の大規模言語モデルファミリーの命令バリエーション。
<i>llamacpp_generate</i>	llama.cpp	llama.cpp GGUF 形式を介したローカル LLM 推論。
<i>llamacpp_embeddings</i>	llama.cpp	llama.cpp GGUF 形式を介したローカルテキスト埋め込み。
<i>llamacpp_reranking</i>	llama.cpp	llama.cpp GGUF 形式を介したローカルリランキング。
<i>llamacpp_ocr</i>	llama.cpp	llama.cpp GGUF 形式を介したローカル OCR 画像からのテキスト抽出。
<i>dummy</i>	—	テストに最適な偽のデータを提供します。

`aidb.model_providers` は、外部 API プロバイダー OpenAI、NVIDIA NIM、Gemini、OpenRouter などでもリストします。これらについては、[External model connections](#) を参照してください。完全な列リファレンスについては、[helpb.model_providers](#) を参照してください。

モデルファイル形式とタスクに一致するプロバイダーを選択し、**カスタムモデル** の下の一致する構成ヘルパーで構成します。または、上記のすぐに使用できる **デフォルトモデル** のいずれかを使用します。

6.1.3 カスタムモデル

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に登録します。

```
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。

以下の BERT、CLIP、T5、および Llama は Candle で実行され、それぞれに独自の構成ヘルパーがあります。llama.cpp 独自のサブセクション、最後は別のランタイムを実行し、構成ヘルパーがまだありません。

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。
<i>image_size</i>	INTEGER	NULL	ピクセル単位の入力画像サイズ。

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>revision</i>	TEXT	NULL	モデルのリビジョンまたはブランチ。
<i>temperature</i>	DOUBLE PRECISION	NULL	サンプリング温度。
<i>seed</i>	BIGINT	NULL	再現可能な出力のランダムシード。
<i>max_tokens</i>	INTEGER	NULL	生成する最大トークン。
<i>repeat_penalty</i>	REAL	NULL	繰り返されるトークンに適用されるペナルティ。
<i>repeat_last_n</i>	INTEGER	NULL	リピートペナルティの対象とされる後続トークンの数。
<i>model_path</i>	TEXT	NULL	モデルの重みへの明示的なローカルパス <code>model</code> をオーバーライドします。
<i>cache_dir</i>	TEXT	NULL	モデルファイルをキャッシュするためのローカルディレクトリ。
<i>top_p</i>	DOUBLE PRECISION	NULL	nucleus サンプリングしきい値。

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>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。
<i>system_prompt</i>	TEXT	NULL	デフォルトのシステムプロンプト。
<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>model_path</i>	TEXT	NULL	モデルの重みへの明示的なローカルパス <code>model</code> をオーバーライドします。
<i>seed</i>	BIGINT	NULL	再現可能な出力のランダムシード。
<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>sample_len</i>	INTEGER	NULL	生成する最大トークン。
<i>use_kv_cache</i>	BOOLEAN	NULL	KV キャッシュを有効にします。
<i>repeat_penalty</i>	REAL	NULL	繰り返されるトークンに適用されるペナルティ。
<i>repeat_last_n</i>	INTEGER	NULL	リピートペナルティの対象とされる後続トークンの数。

!!!note 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。

llama.cpp

llama.cpp は、GGUF 形式のモデルファイル—量子化されたオープンウェイトモデルの一般的な配布形式—を 4 つのプロバイダーを介して実行します。テキスト生成は `llamacpp_generate`、テキスト埋め込みは `llamacpp_embeddings`、リランキングは `llamacpp_reranking`、OCR 画像からテキスト抽出は `llamacpp_ocr`。それらのいずれかを、HuggingFace からダウンロードした、または既にディスク上にある GGUF ファイルをポイントします。AIDB の事前登録された GGUF モデルについては、上記の **デフォルトモデル** を参照してください。以下のレシピを使用して、別のレシピを登録します。

llamacpp_generate

```
SELECT aidb.create_model(
  name => my_llamacpp_chat,
  provider => llamacpp_generate,
  config => jsonb_build_object(
    local_path, /var/lib/aidb/models/qwen2.5-3b-instruct-q4_k_m.gguf,
    n_ctx, 4096,
    temperature, 0.2
  )
);
```

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required unless <i>local_path</i> is set	HuggingFace リポ ID たとえば <i>Qwen/Qwen2-0.5B-Instruct-GGUF</i> 。
<i>model_file</i>	TEXT	Required unless <i>local_path</i> is set	リポジトリ内の <code>.gguf</code> のファイル名。分割 GGUF の場合、任意の 1 つのシャードを名前付けます。すべてのシャードがダウンロードされ、llama.cpp がそれらを再アセンブルします。
<i>revision</i>	TEXT	<i>main</i>	HuggingFace リビジョン。
<i>local_path</i>	TEXT	NULL	ローカルの <code>.gguf</code> ファイルへの絶対パス。 <i>model/model_file</i> <i>revision</i> をオーバーライドします。
<i>cache_dir</i>	TEXT	NULL	ダウンロードした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。

次のページに続く

表 9 – 前のページからの続き

Parameter	Type	Default	Description
<i>n_ctx</i>	INTEGER	4096	コンテキストウィンドウ、トークン単位。GGUF がトレーニングされた値に制限されます。
<i>chat_template</i>	TEXT	NULL	GGUF に焼き付けられたチャットテンプレートをオーバーライドします。
<i>temperature</i>	DOUBLE PRECISION	0.7	サンプリング温度。
<i>top_p</i>	DOUBLE PRECISION	0.9	nucleus サンプリングしきい値。
<i>max_tokens</i>	INTEGER	512	生成する最大トークン。
<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	すべての要求の先頭に追加されるシステムプロンプト。
<i>repeat_penalty</i>	REAL	1.1	繰り返されるトークンに適用されるペナルティ。
<i>repeat_last_n</i>	INTEGER	64	リピートペナルティの対象とされる後続トークンの数。
<i>thinking</i>	BOOLEAN	NULL (keeps tags)	レスポンスでの ` <think>` タグ処理を制御します。`false` はそれらをストリップし、`true/unset` はそれらを保持します。</think>

`llamacpp_generate` は、ネイティブの `openai_responses` および `anthropic_messages` プロバイダーと同様に、ツール呼び出しと構造化出力のために `tools`、`tool_choice`、および `response_format` をサポートします [External model connections](#) を参照してください – [ツール呼び出しと構造化出力](#) を参照してください。

llamacpp_embeddings

```
SELECT aidb.create_model(
  name => my_llamacpp_embed,
  provider => llamacpp_embeddings,
  config => jsonb_build_object(
    model, nomic-ai/nomic-embed-text-v1.5-GGUF,
    model_file, nomic-embed-text-v1.5.Q4_K_M.gguf,
    revision, 0188c9bf409793f810680a5a431e7b899c46104c,
    n_ctx, 2048,
    normalize, true
  )
);
```

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required unless <i>local_path</i> is set	HuggingFace リポ ID たとえば <i>nommic-ai/nomic-embed-text-v1.5-GGUF</i> 。
<i>model_file</i>	TEXT	Required unless <i>local_path</i> is set	リポジトリ内の`.gguf`のファイル名。分割 GGUF の場合、任意の 1 つのシャードを名前付けます。すべてのシャードがダウンロードされ、llama.cpp がそれらを再アセンブルします。
<i>revision</i>	TEXT	<i>main</i>	HuggingFace リビジョン。
<i>local_path</i>	TEXT	NULL	ローカルの`.gguf`ファイルへの絶対パス。 <i>model/model_file/revision</i> をオーバーライドします。
<i>cache_dir</i>	TEXT	NULL	ダウンロードした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>n_ctx</i>	INTEGER	2048	コンテキストウィンドウ、トークン単位。GGUF がトレーニングされた値に制限されます。
<i>pooling</i>	TEXT	<i>unspecified</i>	プーリング戦略 <i>unspecified</i> GGUF 独自のメタデータを使用、 <i>none</i> 、 <i>mean</i> 、 <i>cls</i> 、または`last`
<i>normalize</i>	BOOLEAN	<i>true</i>	出力埋め込みベクトルを L2 正規化します。

llamacpp_reranking

1 つのプロバイダーの背後で 2 つの異なるリランカーアーキテクチャを提供します。スコアリングメカニズムは GGUF のメタデータから自動検出するか、明示的に設定できます。

```
- - Cross-encoder (auto-detected from the GGUFs architecture)
SELECT aidb.create_model(
  name => my_llamacpp_reranker,
  provider => llamacpp_reranking,
  config => jsonb_build_object(
    model, gpustack/bge-reranker-v2-m3-GGUF,
    model_file, bge-reranker-v2-m3-Q4_K_M.gguf,
    revision, 3093af03b1a635e67b084b1d8c03c5f5e020fd05,
    n_ctx, 2048
  )
);

- - Decoder-only (Qwen3-Reranker family), with a custom task instruction
```

(次のページに続く)

(前のページからの続き)

```

SELECT aidb.create_model(
  name => my_llamacpp_reranker_decoder,
  provider => llamacpp_reranking,
  config => jsonb_build_object(
    model, ggml-org/Qwen3-Reranker-0.6B-Q8_0-GGUF,
    model_file, qwen3-reranker-0.6b-q8_0.gguf,
    revision, a02f48bb4f057028298c21fa033da2b30d7742d5,
    n_ctx, 2048,
    instruction, Retrieve legal passages relevant to the query
  )
);

```

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required unless <i>local_path</i> is set	HuggingFace リポ ID たとえば <i>gpustack/bge-reranker-v2-m3-GGUF</i> 。
<i>model_file</i>	TEXT	Required unless <i>local_path</i> is set	リポジトリ内の <code>.gguf</code> のファイル名。分割 GGUF の場合、任意の 1 つのシャードを名前付けます。すべてのシャードがダウンロードされ、llama.cpp がそれらを再アセンブルします。
<i>revision</i>	TEXT	<i>main</i>	HuggingFace リビジョン。
<i>local_path</i>	TEXT	NULL	ローカルの <code>.gguf</code> ファイルへの絶対パス。 <i>model/model_file</i> <i>revision</i> をオーバーライドします。
<i>cache_dir</i>	TEXT	NULL	ダウンロードした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>n_ctx</i>	INTEGER	2048	コンテキストウィンドウ、トークン単位。クエリ/ドキュメントのペアは、適合しない場合、呼び出し時にサイレントに切り捨てられるのではなく、拒否されます。
<i>reranker_type</i>	TEXT	NULL (auto-detected)	スコアリングメカニズム <code>cross_encoder</code> または <code>decoder_only</code> を強制します。誤検出を修正する場合にのみ必要です。

次のページに続く

表 11 – 前のページからの続き

Parameter	Type	Default	Description
<i>instruction</i>	TEXT	Built-in generic retrieval instruction	デコーダー専用のリランカー Qwen3-Reranker ファミリーのタスク指示。クロスエンコーダーによって無視されます。

`llamacpp_reranking` は、GGUF の `general.architecture` メタデータからスコアリングメカニズムを自動検出します。 `qwen3` アーキテクチャは、Qwen3-Reranker ファミリーの `yes/noverdict-token` プロンプトを使用して、デコーダーのみとしてスコアします。他のアーキテクチャはモデルのロード時に拒否されます。誤検出されたにもかかわらず、モデルがこれらの 2 つのスコアパスのいずれかに一致することを確認している場合にのみ、`reranker_type` を明示的に設定します。

!!!note `nim_reranking` や他のリモートリランキングプロバイダーとは異なり、`llamacpp_reranking` の `logit_score` は、正規化された 0~1 の関連性スコアではなく、生の無制限のモデルロジットです。プロバイダーまたは呼び出し全体ではなく、1 つの `aidb.rerank_text()` 呼び出し内のスコアを比較します。

!!!note `llama.cpp` にはまだ構成ヘルパー機能がありません。上記のように、`jsonb_build_object()` を使用して `config` をビルドします。また、今日は CPU のみです。GPU オフロードオプションはありません。 `promoteb.max_threads` および `Candle` ランタイムが使用できるスレッドの数を制御するには、`promoteb.max_threads` を参照してください。

llamacpp_ocr

ローカルビジョン言語の GGUF モデルとそのマルチモーダルプロジェクター `mmproj` ファイルを使用して、画像からテキストを抽出します。

```
SELECT aidb.create_model(
  name => my_llamacpp_ocr,
  provider => llamacpp_ocr,
  config => jsonb_build_object(
    model, ggml-org/LightOnOCR-2-1B-GGUF,
    model_file, LightOnOCR-2-1B-Q8_0.gguf,
    mmproj_file, mmproj-LightOnOCR-2-1B-Q8_0.gguf,
    revision, 7d4daf70c2856a155b1413def00f4ae59843a9cf,
    n_ctx, 8192
  )
);
```

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required unless <i>local_path</i> is set	HuggingFace リポ ID たとえば <code>ggml-org/LightOnOCR-2-1B-GGUF</code> 。
<i>model_file</i>	TEXT	Required unless <i>local_path</i> is set	リポジトリ内のベースモデル <code>.gguf</code> のファイル名。

次のページに続く

表 12 – 前のページからの続き

Parameter	Type	Default	Description
<i>mmproj_file</i>	TEXT	Required unless <i>mmproj_local_path</i> is set	ベースモデルと同じ HuggingFace リポジトリ/リビジョンにあるマルチモーダルプロジェクター <i>mmproj` .gguf`</i> のファイル名。
<i>mmproj_local_path</i>	TEXT	NULL	ローカル <i>mmproj` .gguf`</i> ファイルへの絶対パス。HuggingFace ベースの <i>`mmproj_file`</i> ダウンロードをオーバーライドします。
<i>revision</i>	TEXT	<i>main</i>	HuggingFace リビジョン。
<i>local_path</i>	TEXT	NULL	ローカルベースモデル <i>` .gguf`</i> ファイルへの絶対パス。 <i>modellmodel_file`revision`</i> をオーバーライドします。
<i>cache_dir</i>	TEXT	NULL	ダウンロードした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>n_ctx</i>	INTEGER	8192	コンテキストウィンドウ、トークン単位。
<i>prompt</i>	TEXT	Built-in transcription instruction	各画像とともに送信される OCR 指示。プロンプトの影響はモデルによって異なります。
<i>temperature</i>	DOUBLE PRECISION	0.0	サンプリング温度。クリエイティブなサンプリングではなく忠実な文字起こしには、デフォルトの <i>`0.0`</i> が使用されます。

6.1.4 作成時にモデルの検証

デフォルトでは、`aidb.create_model()` は登録の一部としてモデルを検証します。モデルファイルをダウンロードしてロードし、小さなプローブ推論を実行して、登録がコミットする前にモデルが機能していることを確認します。ローカルモデルの場合、これにより、大規模な可能性のあるダウンロードとロードが、最初に使用するのではなく作成時に移動します。

検証が失敗した場合たとえば、モデル識別子が HuggingFace に存在しない、`model_path` が間違っている、または重みをロードできない場合、すぐにエラーが報告され、モデルは登録されません。

```
-- Validated by default: downloads, loads, and probes the model now.
SELECT aidb.create_model(my_bert, bert_local,
    config => aidb.bert_config(sentence-transformers/all-MiniLM-L6-v2));
```

`validate => false` を渡して、ダウンロードまたはロードせずにモデルを登録します。ダウンロードを最初に使用するまで延期したり、ネットワークアクセスがまだない環境にモデルを登録したりするのに役立ちます。

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

`aidb.validate_model()` を使用して、モデルを後で検証したり、既存のモデルを再チェックしたりできます。

6.1.5 ローカルモデルのダウンロード

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

進行状況メッセージ

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

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

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

再試行と再開

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

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

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

検証と自己修復

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

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

6.1.6 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。カスタムモデルとデフォルトモデルは交換可能です。

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

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

すべてのモデル構成ヘルパーファンクションの完全なパラメーターの詳細については、[Models reference](#) を参照してください。一部のファンクション `generate_text()`、`summarize_text()` は、呼び出しごとの構成オーバーライドも受け入れます。作成時と呼び出し時間の設定については、[構成モデル](#) 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。

6.1.7 障害対応

llama.cpp モデルのロードと推論の問題

`llamacpp_generate`、`llamacpp_embeddings`、または `llamacpp_reranking` モデルのロードに失敗したり、タイムアウトしたり、その他の予期しない動作をした場合、AIDB 自分自身のエラーメッセージには、診断するための十分な詳細がない場合があります。llama.cpp のネイティブ診断ログを有効にして、サーバーログに独自の出力 GGUF メタデータ、テンソル形状、およびバックエンド/モデルの初期化を表示します。

```
aidb.enable_llamacpp_logs = true
```

このパラメーターは、拡張機能の初期化時に 1 回読み込まれるため、Postgres を再起動して変更を有効にします。SET および `pg_reload_conf()` だけでは適用されません。完全なパラメーターリファレンスについては、[agentb.enable_llamacpp_logs](#) を参照してください。

llama.cpp のネイティブ出力が冗長であるためモデルロードごとに数百行があるため、デフォルトではオフになっているため、問題を調査しているときにのみオンにして、その後オフに戻します。これは llama.cpp モデルにのみ適用されます。Candle ベースモデル `bert_local`、`clip_local`、`t5_local`、`llama_instruct_local` は影響を受けません。

6.2 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。

!!!note このページの例では、簡潔にするために `credentials => '{"api_key": "..."}'::JSONB` としてインラインで資格情報を渡します。config 自分自身は、ネストの深さで `api_key` または `basic_auth` キーを含めることはできません— `aidb.create_model()` はそれを拒否します—したがって、常に別の `credentials` 引数を介して資格情報を渡します。シークレットをデータベースに保存しないようにするには、代わりに `credentials_env` を渡し、接続時に読み取る環境変数を名前付けます。環境変数からの資格情報を参照してください。

登録されると、外部モデルはデフォルトの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。

6.2.1 作成時にモデルの検証

デフォルトでは、`aidb.create_model()` は、登録の一部として、小さなプローブ要求をプロバイダーに送信することにより、外部モデルを検証します。これにより、登録がコミットされる前に、エンドポイントが到達可能であり、構成と資格情報が正しいことが確認され、無効な API キーや間違った URL などの問題が、最初に使用するときではなくすぐに表面化します。プローブが失敗した場合、エラーが報告され、モデルは登録されません。

```
- - Validated by default: sends a test request to the provider now.
SELECT aidb.create_model(
  my_openai_embedder,
  openai_embeddings,
  config      => aidb.embeddings_config(model => text-embedding-3-small),
  credentials => '{"api_key": "sk-..."}'::JSONB
);
```

`validate => false` を渡してプローブをスキップします。たとえば、エンドポイントまたは資格情報がまだ利用できない場合、またはスクリプトによるセットアップ中の要求を回避します。

```

SELECT aidb.create_model(
  my_openai_embedder,
  openai_embeddings,
  config      => aidb.embeddings_config(model => text-embedding-3-small),
  credentials => {"api_key": "sk-..."}::JSONB,
  validate    => false
);

```

!!!note 検証は実際の要求をプロバイダーに送信するため、作成時に Postgres ホストからのネットワークアクセスが必要で、少額の使用コストが発生する場合があります。

`aidb.validate_model()` を使用して、モデルを後で検証したり、既存のモデルを再チェックしたりできます。

6.2.2 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,
    url   => https://api.openai.com/v1
  ),
  credentials => {"api_key": "sk-..."}::JSONB
);

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

```

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

チャット完了

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

```
SELECT aidb.create_model(
  my_openai_llm,
  openai_completions,
  config => aidb.completions_config(
    model => gpt-4o,
    temperature => 0.2
  ),
  credentials => [{"api_key": "sk-..."}::JSONB
);
```

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> 。

次のページに続く

表 14 – 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。
<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 `aidb.max_tokens_config()` 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、`true` に設定します。

!!!note 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。

6.2.3 OpenAI Responses API

AIDB には、OpenAI モデル用の 2 つのプロバイダーがあります。どちらも同じモデル識別子 `gpt-4o`、`gpt-5.x` など動作するため、どのモデルが必要かではなく、やっていることに基づいてそれらを選択してください。

Provider	API	Best for	Tool calling
<i>openai_completions</i>	Chat Completions (<i>/v1/chat/completions</i>)	Simple, one-off text generation — <i>aidb.generate_text()</i> , summarization, pipeline steps.	Rejects <i>tools/tool_choice/response_format</i> in <i>aidb.generate_text()</i> 's <i>inference_config</i> .
<i>openai_responses</i>	Responses API (<i>/v1/responses</i>)	Agentic workloads — multi-step reasoning, reliable tool calling, structured output.	Native — emits the API's real request fields and parses real tool-call responses back.

どちらのプロバイダーも [Managing agents](#) エージェントをバックグランドすることもできます。AIDB は常にエージェントツール呼び出し機能を提供しますが、**方法** はプロバイダーによって異なります。 `openai_responses`

によってサポートされ、ツール呼び出しは、本物のレスポンス API ツール呼び出しとして有線で送られます。openai_completions またはほぼその他のプロバイダーに支えられて、AIDB はプロンプトテキストで使用可能なツールを説明し、モデルのプレーンテキストの応答からツールコールバックを解析します。モデル自身のツール呼び出し動作ではなく、シミュレーション。openai_responses は、エージェントにとってより信頼できる選択です。ツール呼び出しではなく、テキスト出力が必要な場合に openai_completions を使用します。

```
SELECT aidb.create_model(
  my_gpt,
  openai_responses,
  config      => aidb.openai_responses_config(model => gpt-5.1),
  credentials => {"api_key": "sk-..."}::JSONB
);
```

Azure AI Foundry

Azure AI Foundry の統合された Responses API サーフェスでホストされるモデルに openai_responses_azure を使用します。これは、直接 OpenAI と同じワイヤー形状ですが、一般的なデフォルトがないため url が必要です。リソースのエンドポイントに向けます。

```
SELECT aidb.create_model(
  my_gpt_azure,
  openai_responses_azure,
  config      => aidb.openai_responses_config(
    model => gpt-5.1,
    url   => https://<resource>.openai.azure.com/openai/v1/responses
  ),
  credentials => {"api_key": "..."}::JSONB
);
```

完全なパラメーターリストについては [promoteb.openai_responses_config](#) を参照し、各プロバイダーで tools、tool_choice、および response_format を使用する場合は [ツール呼び出しと構造化出力](#) を参照してください。

6.2.4 Anthropic Messages API

anthropic_messages プロバイダーは、Anthropic のネイティブメッセージ API /v1/messages に接続します。

```
SELECT aidb.create_model(
  my_claude,
  anthropic_messages,
  config      => aidb.anthropic_messages_config(model => claude-opus-4-6),
  credentials => {"api_key": "sk-ant-..."}::JSONB
);
```

Azure AI Foundry と AWS Bedrock

残りの 2 つのバリエーションは、同じ構成ヘルパーとワイヤーボディを共有し、認証とエンドポイントのみが異なります。

- `anthropic_messages_azure` — Azure AI Foundry のネイティブ Anthropic Messages API サーフェス。`url` が必要でソースのメッセージエンドポイント。
- `anthropic_messages_bedrock` — AWS Bedrock の InvokeModel API、Bedrock API キー AWS SigV4 ではなくベアラートークンで認証されます。`url` は必須で、リージョンの `bedrock-runtime` ベースエンドポイントである必要があります。AIDB はモデル ID をパスに自動的に追加します。

```
-- Azure AI Foundry
SELECT aidb.create_model(
  my_claude_azure,
  anthropic_messages_azure,
  config => aidb.anthropic_messages_config(
    model => claude-opus-4-6,
    url => https://<resource>.services.ai.azure.com/anthropic/v1/messages
  ),
  credentials => {"api_key": "..."}::JSONB
);

-- AWS Bedrock
SELECT aidb.create_model(
  my_claude_bedrock,
  anthropic_messages_bedrock,
  config => aidb.anthropic_messages_config(
    model => anthropic.claude-opus-4-6-v1:0,
    url => https://bedrock-runtime.us-east-1.amazonaws.com
  ),
  credentials => {"api_key": "..."}::JSONB
);
```

!!!note Anthropic の Messages API には、ネイティブの構造化出力フィールドがありません。`response_format` を使用すると、AIDB はそれを内部で合成ツールの強制呼び出しに変換します。機能的に同等ですが、これは、`response_format` を同じ呼び出しで `tools/tool_choice` と組み合わせることはできないことを意味します。

`anthropic_messages` は、クロード専用構築された AIDB の唯一のプロバイダーです。`openai_completions` と同等の古い `anthropic_completions` はありません。ベンダーごとのプロバイダーではなく、単一のユニファイドゲートウェイを介してクロードまたは OpenAI、または他のベンダーに到達したい場合は、以下の [OpenRouter](#) を参照してください。トレードオフは上記の `openai_completions` と同じです — OpenRouter モデルは、エージェントをバックグランドにネイティブではなく、シミュレートされたプロンプト挿入されたツール呼び出しを取得します。

完全なパラメーターリストについては、[helpb.anthropic_messages_config](#) を参照してください。

6.2.5 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

`aidb.completions_config()` とともに `nim_completions` プロバイダーを使用します。NIM の補完エンドポイントは OpenAI 互換であるため、`openai_completions` が使用するのと同じ構成ヘルパーです。完全なパラメーターリストについては、上記の [チャット完了](#) を参照してください。

```
SELECT aidb.create_model(
  my_nim_llm,
  nim_completions,
  config      => aidb.completions_config(
    model => meta/llama-3.3-70b-instruct
  ),
  credentials => {"api_key": "nvapi-..."}::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 エンドポイントをポイントして url から `aidb.completions_config()` に渡します。

登録後、`aidb.generate_text()` でモデルを実行します。

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

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

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(
    model => nvidia/nvclip
  ),
  credentials => {"api_key": "nvapi-..."}::JSONB
);
```

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_paddle_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(),
  credentials => {"api_key": "nvapi-..."}::JSONB
);
```

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(
    model => nvidia/nv-rerankqa-mistral-4b-v3
```

(次のページに続く)

(前のページからの続き)

```

),
credentials => {"api_key": "nvapi-..."}::JSONB
);

```

`aidb.nim_reranking_config()` パラメーター

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

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

6.2.6 Google Gemini

`aidb.gemini_config()` で `gemini` プロバイダーを使用します。`api_key` は `aidb.gemini_config()` の最初のパラメーターであり、デフォルトありませんが、`NULL` として渡し、代わりに `create_model()` の `credentials` 引数を介して実際のキーを提供します。`config` には `api_key` フィールドを含めることはできません。

```

SELECT aidb.create_model(
  my_gemini,
  gemini,
  config      => aidb.gemini_config(
    api_key => NULL,
    model  => gemini-2.0-flash
  ),
  credentials => {"api_key": "AIza..."}::JSONB
);

```

`aidb.gemini_config()` パラメーター

Parameter	Type	Default	Description
<i>api_key</i>	TEXT	Required	デフォルトはありませんが、`NULL`を渡し、代わりに `create_model()`の `credentials`引数を介して本当のキーを提供します。`config`に`api_key`を含めることはできません。
<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。

6.2.7 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),
  credentials => {"api_key": "sk-or-..."}::JSONB
);
```

aidb.openrouter_chat_config() パラメーター

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

エンベディング

```
SELECT aidb.create_model(
  my_or_embedder,
  openrouter_embeddings,
  config      => aidb.openrouter_embeddings_config(mistral/mistral-embed),
  credentials => {"api_key": "sk-or-..."}::JSONB
);
```

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。
<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、[Models reference](#) を参照してください。

6.3 Support matrix

このページでは、EDB が AIDB と連携するようにテストおよび確認した特定のモデルバリエーションをリストします。各テーブルは同じ形状に従います。モデル識別子、デフォルトかどうか、およびモデルの適性の説明。説明を使用して、ユースケースに適切なバリエーションを選択します。

6.3.1 ローカルモデル

デフォルトモデル

AIDB は、拡張機能が作成されるときにこれらのモデルを自動的に登録します。使用を開始するための構成は必要ありません。これらの使用方法については [デフォルトモデル](#) を、各プロバイダーの意味については [モデルプロバイダー](#) を参照してください。

Model name	Provider	Description
<i>bert</i>	<i>bert_local</i> (Candle)	汎用の英語テキスト埋め込み。ほとんどのテキスト検索および RAG ユースケースに適したデフォルト。

次のページに続く

表 22 – 前のページからの続き

Model name	Provider	Description
<i>clip</i>	<i>clip_local</i> (Candle)	マルチモーダル検索のための共同テキスト/画像埋め込み。
<i>t5</i>	<i>t5_local</i> (Candle)	軽量のテキストからテキストへの生成 — 翻訳、要約、質問応答。
<i>llama</i>	<i>llama_instruct_local</i> (Candle)	指示に従っているローカルチャット/完了モデル。
<i>bge-small-en-v1.5-f16</i>	<i>llamacpp_embeddings</i> (llama.cpp)	コンパクトで高速な英語テキストの埋め込み。
<i>nomie-embed-text-v1.5-Q8_0</i>	<i>llamacpp_embeddings</i> (llama.cpp)	<i>bge-small-en-v1.5-f16</i> よりも大きいコンテキストウィンドウ 2048 トークンを使用した汎用英語エンベディング。
<i>bge-m3-f16</i>	<i>llamacpp_embeddings</i> (llama.cpp)	8192 トークンコンテキストウィンドウを使用した多言語埋め込み - 長い文書または英語以外の文書に適しています。
<i>qwen3-embedding-0.6b-Q8_0</i>	<i>llamacpp_embeddings</i> (llama.cpp)	16384 トークンのコンテキストウィンドウを使用した大規模な埋め込みモデル。
<i>qwen3-embedding-4b-Q8_0</i>	<i>llamacpp_embeddings</i> (llama.cpp)	最大の事前登録されたエンベディングモデル 20480 トークンコンテキスト - 高品質、より多くのコンピューティング。
<i>qwen3.5-0.8b-Q8_0</i>	<i>llamacpp_generate</i> (llama.cpp)	非常に大きなコンテキストウィンドウ 262144 トークンを使用した、小規模で高速なチャット/完了モデル。
<i>llama-3.2-1b-instruct-Q8_0</i>	<i>llamacpp_generate</i> (llama.cpp)	軽量のローカルチャット/補完のためのコンパクトな Meta Llama 3.2 指示モデル。
<i>lightonocr-2-1b-Q8_0</i>	<i>llamacpp_ocr</i> (llama.cpp)	OCR のコンパクトビジョン言語モデル — 画像からテキストを抽出します。
<i>dummy</i>	<i>dummy</i>	偽のデータを返します — 実際の推論を実行せずにパイプラインをテストするため。

その他のテストされたバリエーション Candle

次の HuggingFace モデルバリエーションは、上記のデフォルトを超えて、AIDB の Candle ベース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ています。`aidb.create_model()` を呼び出すときに、これらの識別子を対応する構成ヘルパーで使います — [カスタムモデル](#) を参照してください。

BERT *bert_local*

Model identifier	Default	Description
<i>sentence-transformers/all-MiniLM-L6-v2</i>	OK	汎用の英語文埋め込み。速くて小さい。これは、デフォルトの <code>'bert'</code> モデルが使用するものです。

次のページに続く

表 23 – 前のページからの続き

Model identifier	Default	Description
<i>sentence-transformers/paraphrase-multilingual-mpnet-base-v2</i>	—	多言語埋め込み — テキストが英語のみでない場合に使用します。
<i>sentence-transformers/multi-qa-MiniLM-L6-cos-v1</i>	—	一般的な文の類似性ではなく、質問応答と検索用に調整されています。
<i>sentence-transformers/paraphrase-TinyBERT-L6-v2</i>	—	デフォルトより小さくて速いですが、品質は多少コストがかかります。

```
SELECT aidb.create_model(my_bert, bert_local,
    config => aidb.bert_config(sentence-transformers/paraphrase-multilingual-mpnet-
    ↪base-v2));
```

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。

CLIP `clip_local`

Model identifier	Default	Description
<i>openai/clip-vit-base-patch32</i>	OK	テストされた唯一の CLIP バリエント。ジョイントテキスト/画像の埋め込み。これは、デフォルトの `clip` モデルが使用するものです。

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。

T5 `t5_local`

Model identifier	Default	Description
<i>t5-small</i>	OK	最速、最小の T5 バリエント。これは、デフォルトの `t5` モデルが使用するものです。
<i>t5-base</i>	—	デフォルトより大きい - 品質は高くなりますが、速度は遅くなります。
<i>t5-large</i>	—	最大の T5 バリエント—最高の品質、ほとんどのリソースを使用します。

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.generate_text()` の両方をサポートしています。

ラマ llama_instruct_local

Model identifier	Default	Description
<i>TinyLlama/TinyLlama-1.1B-Chat-v1.0</i>	OK	小規模で高速な指示追従のチャットモデル。これは、デフォルトの `llama` モデルが使用するものです。
<i>HuggingFaceTB/SmolLLM-135M-Instruct</i>	—	非常に小さくて速い - これらのバリエーションの低品質、最低のリソース使用量。
<i>HuggingFaceTB/SmolLLM-360M-Instruct</i>	—	Small — 135M バリエーションからの品質のステップアップ。
<i>HuggingFaceTB/SmolLLM-1.7B-Instruct</i>	—	これらのバリエーションの中で最大のもの — リソースの使用量が多くなりますが、デフォルトよりも高品質です。

すべての Llama バリエーションは、テキスト生成 `aidb.generate_text()` をサポートしています。

!!!note 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ーションです。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ーションのみが EDB によってテストされています。

その他のテストされたバリエーション llama.cpp

リランキング llamacpp_reranking

Model identifier	Default	Architecture	Description
<i>gpustack/bge-reranker-v2-m3-GGUF</i>	—	Cross-encoder	多言語クロスエンコーダーのリランカー。分類ヘッドを介して各クエリ/ドキュメントペアを直接スコアリングします。
<i>ggml-org/Qwen3-Reranker-0.6B-Q8_0-GGUF</i>	—	Decoder-only	Qwen3-Reranker ファミリー。モデルの yes/no 判定トークンロジットを読み取ることにより、各ペアのスコアを獲得します。 llamacpp_reranking を参照してください。

どちらのバリエーションもデフォルトモデルとして事前登録されていません。 `aidb.create_model()` を呼び出すときに、 `jsonb_build_object()` でいずれかの識別子を使用します — [llamacpp_reranking](#) を参照してください。

6.3.2 リモートモデル

リモートモデルは、Postgres ホストで実行されるのではなく、ネットワークを介して呼び出されます。OpenAI および Anthropic の NVIDIA NIM マイクロサービスとチャット/エージェントモデル - ネイティブの `openai_responses` および `anthropic_messages` アダプターを含む複数のプロバイダーを介して到達可能。

NVIDIA NIM

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。登録方法については、[NVIDIA NIM](#) を参照してください。ハイブリッドマネージャーのモデルサービングによって事前展開されていないモデルは、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。

EDB ハイブリッドマネージャーのモデルサービングによって事前展開されるものを含む、プロバイダーごとにサポートされている NIM モデルの完全なリスト、埋め込み、CLIP、OCR、再ランキングについては、[GPU recommendations for default NIM models](#) を参照してください。

OpenAI と Anthropic モデル

以下の各モデル識別子は、複数の AIDB プロバイダーに登録できます。どれを選択するかは、どのモデルであるかではなく、モデルの使用方法によって異なります。

- ```openai_responses``` (+ `openai_responses_azure`) および ```anthropic_messages``` (+ `anthropic_messages_azure` / `anthropic_messages_bedrock`) — AIDB のネイティブ、エージェント対応プロバイダー。直接呼び出しと [Managing agents](#) で作成されたエージェントの両方で、実際の `tools / tool_choice / response_format` サポート。
- ```openai_completions``` OpenAI の古い Chat Completions API。同じ OpenAI モデルで動作し、プレーンテキスト生成 `aidb.generate_text()`、要約、パイプラインステップに最適な選択ですが、`tools / tool_choice / response_format` を直接拒否し、エージェントをバックアップする際にプロンプトインジェクションを介して `_simulates_` ツール呼び出しのみを行います。
- ```openrouter_chat``` OpenRouter 独自のモデルスラッグたとえば `openai/gpt-4o` または `anthropic/claude-opus-4-6` を介した OpenAI、Anthropic、および他のベンダーのモデル以下のすべてを含む単一のゲートウェイ。`openai_completions` と同じネイティブではなくシミュレートされたツール呼び出し。

完全な比較および登録例については、[OpenAI Responses API](#) および [Anthropic Messages API](#) を参照してください。以下の表は、各ベンダーが提供するすべてのモデルの完全なカタログではありません。これらは開始点として一般的に使用されるファミリーを呼び出し、現在のフラッグシップ/効率的な層と参照として長く確立されたバリエーションの両方をカバーしています。完全な現在のラインナップと正確な機能については、ベンダー独自のモデルドキュメントを確認してください。

OpenAI

Model identifier	Native provider(s)	Also usable via	Cloud host support (native)	Description
<i>gpt-5.6-sol-terra / -luna</i>	<i>ope-nai_responses, ope-nai_responses,</i>	<i>ope-nai_completion, open-router_chat</i>	Direct, Azure AI Foundry	GPT-5.6 ファミリーは、機能、レイテンシー、コストの 3 階層のトレードオフで提供されます。層の比較については、OpenAI のドキュメントを確認してください。
<i>gpt-5.4-mini</i>	<i>ope-nai_responses, ope-nai_responses,</i>	<i>ope-nai_completion, open-router_chat</i>	Direct, Azure AI Foundry	GPT-5.4 世代の小規模でコスト効率の高いティア—主力コストの数分の 1 で確かなツール呼び出し精度。
<i>gpt-4o</i>	<i>ope-nai_responses, ope-nai_responses,</i>	<i>ope-nai_completion, open-router_chat</i>	Direct, Azure AI Foundry	老舗のマルチモーダルモデル。最新の GPT-5.x 層が必要ない場合の安定した有名なアンカー。

人間性

Model identifier	Native provider(s)	Also usable via	Cloud host support (native)	Description
<i>claude-opus-4-6</i>	<i>an-thropic_messa, an-thropic_messa, an-thropic_messa,</i>	<i>open-router_chat</i>	Direct, Azure AI Foundry, AWS Bedrock	最も要求の高い推論およびエージェントツール呼び出しタスクのためのフラッグシップクロードモデル。
<i>claude-sonnet-5</i>	<i>an-thropic_messa, an-thropic_messa, an-thropic_messa,</i>	<i>open-router_chat</i>	Direct, Azure AI Foundry, AWS Bedrock	バランスの取れた中層モデル— エージェントおよび日常のチャット/完了のための強力な汎用選択。
<i>claude-haiku-4-5</i>	<i>an-thropic_messa, an-thropic_messa, an-thropic_messa,</i>	<i>open-router_chat</i>	Direct, Azure AI Foundry, AWS Bedrock	最速で最もコスト効率の高い現在のクロード階層-大容量またはレイテンシーの影響を受けるツール呼び出しに適しています。

次のページに続く

表 29 – 前のページからの続き

Model identifier	Native provider(s)	Also usable via	Cloud host support (native)	Description
<i>claude-3-5-haiku</i>	<i>an-thropic_messa</i> , <i>an-thropic_messa</i> , <i>an-thropic_messa</i>	<i>open-router_chat</i>	Direct, Azure AI Foundry, AWS Bedrock	長く確立され、広く使用されているバリエーション。最新のクロードティアが必要ない場合の、安定した有名なアンカー。

Anthropic には、OpenAI が `openai_completions` があるような古いコンプリートスタイルのプロバイダーがありません。 `anthropic_messages` は、 `openrouter_chat` ゲートウェイオプションと並んで、クロードの唯一の直接 AIDB プロバイダーです。

!!!note Azure AI Foundry または AWS Bedrock でのモデルの可用性は、特に最近リリースされた層の場合、ベンダー独自の直接 API でのリリースより遅れる場合があります。 `openai_responses_azure` または `anthropic_messages_bedrock` が動作すると予想されるモデルを拒否した場合、各クラウドプロバイダーに現在の可用性を確認してください。

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、外部ストレージを保存場所として Postgres にマップします。AIDB は、各保存場所を名前でパイプライン参照する **ボリューム** でラップします。ボリュームは **foreign table** であるため、標準の SQL でオブジェクトをリスト、読み取り、書き込み、および削除できます。

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 ストレージ場所のコンテンツを行として公開する外部テーブル。データフォーマット Text、Image、Pdf とオプションのサブパスを保存します。 `aidb.create_volume()` で作成されました。

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

!!!note 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。 `url` 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、リージョンなどのプロバイダー固有の設定を保持します。 `credentials` JSONB オブジェクトは、アクセスキーなどの認証シークレットを伝送します。

!!!note `credentials` パラメーターの明示的な資格情報は、認証の唯一の方法ではありません。これらを省略すると、S3、Azure、および GCS ストレージの場所は、各クラウドプロバイダーの独自のネイティブ資格情報解決 IAM ロール、管理対象/ワークロード ID、インスタンスメタデータ、アプリケーションのデフォルト資格情報にフォールバックします。Postgres が既にクラウドで実行されている場合に役立ちます ID が

アタッチされているため、保存するシークレットはまったくありません。詳細は、PGFS documentation を参照してください。

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,
  url       => s3://my-bucket/my-folder,
  options   => {"region": "us-east-1"},
  credentials => {"access_key_id": "<key>", "secret_access_key": "<secret>"}
  -- or omit the credentials argument to use the cloud providers native credential
  →chain (IAM role, managed identity, etc.) instead
);

- - Public S3 bucket (no credentials required)
SELECT pgfs.create_storage_location(
  name      => my_public_bucket,
  url       => 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,
  url  => file:///tmp/pgfs/
);
```

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

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

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

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

パラメーター

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

!!!note `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 パス処理

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

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

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

7.3.2 例 S3 の PDF

```
SELECT pgfs.create_storage_location(
  pdf_bucket,
  s3://my-docs-bucket,
  options      => {"region": "us-east-1"},
  credentials => {"access_key_id": "<key>", "secret_access_key": "<secret>"}
  -- or omit the credentials argument to use the cloud providers native credential_
  →chain instead
);

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

7.3.3 例ローカルディレクトリ内の画像

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

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

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 ボリュームのリスト

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

```
SELECT * FROM aidb.volumes;

__OUTPUT__
schema | volume | storage_location | path
-----+-----+-----+-----
```

(次のページに続く)

(前のページからの続き)

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

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

7.5.2 ボリュームの削除

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

!!!note 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。

7.6 SQL を使用してボリュームコンテンツにアクセスする

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

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

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

7.6.1 ボリューム内のオブジェクトのリスト

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

(次のページに続く)

(前のページからの続き)

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

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

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

7.6.2 単一オブジェクトの読み取り

正確なキーをフィルタリングしてオブジェクト本体を取得します。

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

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

7.6.3 オブジェクトの書き込み、上書き、および削除

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

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

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

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

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

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

第8章 Agents

AIDB を使用すると、データベース内でエージェントを構成および実行できます。彼らはデータへの安全なローカルアクセスを持っており、モデルが実行される場所によっては、データが Postgres を離れる必要はありません。

エージェントは、名前付けの再利用可能な構成指示、モデル、および呼び出すことができるツールセットです。AIDB は、ReAct スタイルの推論ループを介して実行します。モデルは考え、オプションで1つ以上のツールを呼び出し、結果を観察し、答えがあるまで繰り返します。

!!!note これは、[Agent Factory in Hybrid Manager \(HM\)](#) とは異なる明確なデータベース内機能です。AIDB のエージェントは、Postgres データベース内で SQL ファンクション呼び出しとして完全に実行されます。

エージェントは通常の Postgres オブジェクトであり、SQL で作成および照会されます。以下の例では、小さな text-to- SQL エージェントをエンドツーエンドで構築します。データベーススキーマを事前に知らず、*catalog discovery tools* で検出し、データに対して実際のクエリーを実行することにより質問に回答します。

8.1 例 text-to- SQL エージェント

8.1.1 セットアップ

サンプルデータを作成し、モデルを登録し、エージェントを作成します。

```
CREATE TABLE public.foods (
  id SERIAL PRIMARY KEY,
  name TEXT NOT NULL,
  category TEXT NOT NULL,
  calories INTEGER NOT NULL,
  price_usd NUMERIC(6,2) NOT NULL,
  is_vegetarian BOOLEAN NOT NULL DEFAULT false
);

INSERT INTO public.foods (name, category, calories, price_usd, is_vegetarian) VALUES
  (Apple, Fruit, 95, 0.50, true),
  (Banana, Fruit, 105, 0.30, true),
  (Grilled Chicken Breast, Meat, 165, 3.50, false),
  (Salmon Fillet, Fish, 208, 6.00, false),
  (Broccoli, Vegetable, 55, 1.20, true),
  (Cheddar Cheese, Dairy, 402, 2.80, true),
  (Brown Rice, Grain, 216, 0.80, true),
  (Bacon, Meat, 541, 4.20, false),
```

(次のページに続く)

```

(Almonds,           Nuts,      579, 5.50, true),
(Dark Chocolate,    Snack,     546, 3.00, true);

SELECT aidb.create_model(
  gpt_5_6_terra_azure,
  openai_responses,
  config => aidb.openai_responses_config(
    model => gpt-5.6-terra,
    url   => https://my-ai-service.example.com/openai/v1/responses
  ),
  credentials => jsonb_build_object(
    api_key, <your-api-key>
  )
);

SELECT aidb.create_agent(
  name           => text2sql_demo,
  instructions => You are a helpful, read-only database assistant that answers
    questions by
    || querying the database. You do not know this databases schema
    in
    || advance -- before answering, use the catalog_list_objects
    tool to
    || discover which tables exist, then use catalog_get_object_
    details to
    || inspect the columns of any table you intend to query. Never
    guess a
    || table or column name -- always verify it via these catalog
    tools first.
    || Once you know the relevant table and columns, use the run_sql_
    query
    || tool to fetch the actual data needed to answer the question --
    never
    || state a specific result as fact unless it came from a
    successful
    || run_sql_query call. You only have read-only access and cannot
    modify
    || any data. Once a query has given you enough information to
    answer,
    || respond immediately -- do not repeat a tool call you have
    already made
    || with the exact same arguments.,
  model          => gpt_5_6_terra_azure,
  tools          => ARRAY[run_sql_query, catalog_list_objects, catalog_get_object_
    details]

```

(前のページからの続き)

```
);
```

8.1.2 会話

データに関するエージェントに質問します。バックグラウンドでは、使用可能なテーブルをリストし、foods 列を検査し、クエリーを実行し、結果から回答します。

```
SELECT * FROM aidb.agent_converse(text2sql_demo, Which vegetarian food item has the
fewest calories?);
__OUTPUT__
      conversation_id      | message
-----+-----
08571a03-0758-4cda-8595-607e2a69c8ce | Broccoli has the fewest calories among vegetarian items, with 55 calories .
(1 row)
```

推論ループの個々のツール呼び出しと結果を監視するには、`debug => true` で再実行します。 *Invoking agents* を参照してください。

8.1.3 比較

エージェントの推論ループとツールを使用せずに、同じモデルに直接尋ねられた同じ質問には、モデルのトレーニングデータからのみ回答できます。foods テーブルの存在を知りません。

```
SELECT aidb.generate_text(gpt_5_6_terra_azure, Which vegetarian food item has the
fewest calories?);
__OUTPUT__
      generate_text
-----+-----
Among common solid vegetarian foods, raw watercress is one of the lowest-
calorie options, at about 11 calories per 100 g .
[...]
(1 row)
```

エージェントは実際のデータに基づいて回答を根拠にしました。裸のモデル呼び出しは推測することしかできませんでした。

エージェントの tools 配列は、AIDB の統合ツールカタログつまりビルトイン AIDB オペレーション、登録するカスタムパラメーター化 SQL クエリー、または外部 MCP サーバーからインポートされたオペレーションから取得します。完全なカタログについては、 *ツールとデリゲート* を参照してください。

8.2 ドキュメントマップ

Page	What it covers
<i>Managing agents</i>	<i>aidb.create_agent()</i> , updating and deleting agents, roles, structured output
<i>Invoking agents</i>	<i>aidb.agent_converse()</i> , conversations and sessions, read-only mode, debug mode, budgets and limits
<i>Delegation</i>	Presenting delegates to the model, continuing delegate conversations, what propagates across a handoff, delegation depth

完全なパラメーターテーブルと戻り型については、*Agents reference* を参照してください。

8.2.1 Managing agents

エージェントは `aidb.create_agent()` で作成された `aidb.agents` の行です。 *Invoking agents* を呼び出すまでは何も実行されません。エージェントの作成は、構成を保存するだけです。

```
SELECT aidb.create_agent(
  name           => db_helper,
  instructions    => You are a helpful assistant that answers questions about_
→this database. Be concise.,
  model          => my_gpt,
  tools          => ARRAY[run_sql_query], -- optional; tool names from aidb.
→tools. Default: NULL (no tools)
  delegates      => NULL,                -- optional; names of other_
→agents to delegate to. Default: NULL (no delegation)
  role           => NULL,                -- optional; Postgres role tool_
→calls run as. Default: NULL (run as the calling role)
  output_type    => NULL,                -- optional; structured output_
→schema from aidb.output_type(). Default: NULL (plain-text answer)
  input_token_budget => NULL,            -- optional; input token cap per_
→agent_converse call. Default: NULL (no limit)
  output_token_budget => NULL,          -- optional; output token cap_
→per agent_converse call. Default: NULL (no limit)
  max_iterations => NULL,                -- optional; reasoning-round cap_
→per call. Default: NULL (a hard ceiling of 25 always applies)
  timeout        => NULL,                -- optional; wall-clock seconds_
→cap per call. Default: NULL (300 seconds applied at runtime)
  budget_strategy => attempt_complete,  -- optional; behavior when a_
→budget is exceeded. Default: attempt_complete
  preset         => NULL                 -- optional; reserved for future_
→use, has no effect today. Default: NULL
);
```

成功した場合は行を返しません。失敗した場合たとえば、名前が既に使用されている場合、`error` 列セットを含む単一の行を返します。これは発生しないため、`create_agent` 呼び出しのバッチは途中で中断で

きません。

```
SELECT * FROM aidb.create_agent(db_helper, Be helpful., my_gpt);
```

__OUTPUT__

error

agent db_helper already exists

(1 row)

モデルの選択

model は、テキストを生成し、指示に従うことができる登録済みの **モデルの選択** を指定する必要があります。埋め込み、リランキング、OCR、およびマルチモーダル埋め込みプロバイダー bert_local、openai_embeddings、nim_reranking、clip_local などエージェントをバックすることができず、t5_local は、指示フォローまたはシステムプロンプトのサポートがないため、完全に拒否されます。他のすべてのテキスト生成プロバイダーは動作し、ツール呼び出しの品質によって2つの層に分割されます。

優先—これらは実際のプロバイダーネイティブのツール呼び出し要求を送信します。

- openai_responses + openai_responses_azure
- anthropic_messages + anthropic_messages_azure / anthropic_messages_bedrock
- llamacpp_generate

サポートされている—これらは動作しますが、AIDB は、プロンプトテキストで使用可能なツールを説明し、モデルのプレーンテキストの応答からツールコールバックを解析することにより、ツール呼び出しをシミュレートします。

- openai_completions
- completions 他の OpenAI-Chat-Completions 互換エンドポイント
- nim_completions
- openrouter_chat
- gemini
- llama_instruct_local

優先/サポートされている区別の詳細については、**ツール呼び出しと構造化出力** を参照してください。create_agent を呼び出すときにモデルはまだ存在する必要はありません。agent_converse のみが解決可能である必要があります。

ツールとデリゲート

tools は、エージェントが呼び出すことができる *Tools reference* aidb.tools からの名前の配列です。delegates は、*Delegation* ことができる他のエージェントの名前の配列です。どちらも create_agent / update_agent 時間に aidb.tools / aidb.agents に対して検証されます。不明な名前は、agent_converse 内で後で表面化するのではなく、すぐに拒否されます。ツールまたはデリゲートがないエージェントの場合は、いずれかを省略します。

```
SELECT aidb.create_agent(
  name          => help_desk,
  instructions => Help users with general questions, delegating data questions to a
  ↳specialist.,
  model         => my_gpt,
  tools         => ARRAY[run_sql_query],
  delegates     => ARRAY[sql_specialist]
);
```

アクセス許可

デフォルトでは、エージェントのツールは、Postgres ロールが `agent_converse` を呼び出すときに `runs` を呼び出します。代わりに `role` を渡してエージェントを特定のロールに固定します—たとえば、呼び出したアプリケーションユーザーが持っているよりも狭いテーブル権限を与えます。呼び出し者は、既にそのロール `pg_has_role(..., 'MEMBER')` のメンバーである必要があります。 `create_agent` / `update_agent` 属していないロールで構成されたエージェントを拒否します。

```
SELECT aidb.create_agent(
  name          => readonly_reporter,
  instructions => Answer questions about sales data. Never modify data.,
  model         => my_gpt,
  tools         => ARRAY[run_sql_query],
  role          => reporting_role
);
```

構造化された出力

デフォルトでは、`agent_converse` の回答はプレーンテキストです。代わりに構造化 JSON を取得するには、`aidb.output_type()` および `aidb.output_field()` を使用してスキーマをビルドします。

```
SELECT aidb.create_agent(
  name          => sentiment_tagger,
  instructions => Classify the sentiment of the given text.,
  model         => my_gpt,
  output_type   => aidb.output_type(
    aidb.output_field(sentiment, TEXT, One of: positive, negative, neutral),
    aidb.output_field(confidence, FLOAT)
  )
);
```

`aidb.output_field()` の `description` 引数はオプション。 `agent_converse` の呼び出し時 `output_type` 引数は、その 1 つのコールのみエージェントの構成済みスキーマをオーバーライドします。

予算、イテレーション、およびタイムアウト

`input_token_budget`、`output_token_budget`、`max_iterations`、および `timeout` 秒は、単一の `agent_converse` 呼び出しのコストと実行時間を制限します。`budget_strategy` は、予算を超過した場合の動作を制御します。すべてはオプション。これはエージェントの静的構成のプロパティではなく、会話の実行方法のプロパティであるため、完全な動作については[予算と制限](#)を参照してください。

```
SELECT aidb.create_agent(
  name           => cost_capped_helper,
  instructions    => Answer questions about the orders table.,
  model          => my_gpt,
  input_token_budget => 20000,    -- INTEGER; input tokens allowed per agent_
  ↳converse call
  output_token_budget => 2000,    -- INTEGER; output tokens allowed per agent_
  ↳converse call
  max_iterations  => 10,         -- INTEGER; reasoning rounds allowed per call (a_
  ↳hard ceiling of 25 always applies)
  timeout        => 60,         -- INTEGER; wall-clock seconds allowed per call
  budget_strategy => error      -- TEXT; one of ignore, error, summarize, attempt_
  ↳complete
);
```

エージェントの更新と削除

`aidb.update_agent()` は `create_agent` と同じフィールドを取ります。すべてオプション。渡したフィールドのみが変更します。その他はすべて現在の値を維持します。

```
SELECT aidb.update_agent(db_helper, model => my_new_gpt, max_iterations => 15);
```

専用の「list agent」ファンクションはありません。`aidb.agents` は通常のテーブルであるため、`SELECT * FROM aidb.agents` はそれらをリストします。完全な列リストについては、[Agents reference](#) を参照してください。

`aidb.delete_agent()` は、名前でエージェントを削除します。

```
SELECT aidb.delete_agent(db_helper);
```

エージェントに会話履歴がある場合、これはデフォルトで失敗します内部タスクレコードは外部キーによって参照されます。`force => true` を渡して、エージェントとともにその内部履歴を削除します。会話トランスクリプト自身は、いずれの場合も変更されず、それを生成したエージェントが削除された後でも[会話の取得](#)で読み取り可能なままです。

```
SELECT aidb.delete_agent(db_helper, force => true);
```

!!!note `create_agent` / `update_agent` は、`preset` 引数たとえば `'text_to_sql'` などを受け入れます。これはエージェントに保存されますが、現在その動作は変更しません。将来の使用のために予約されています。

8.2.2 Invoking agents

`aidb.agent_converse()` は、プロンプトをエージェントに送信し、回答が生成されるか、制限に到達するまで、その推論ループを実行します-考え、オプションでツールを呼び出し、観察、繰り返します。

```
SELECT * FROM aidb.agent_converse(
  agent_name      => db_helper,
  prompt          => How many rows are in the orders table?,
  conversation_id => NULL, -- optional; continue an existing conversation by id.
  ↳Default: NULL (starts a new conversation)
  read_only       => NULL, -- optional; force read-only mode on/off. Default: NULL
  ↳(auto: on for a read replica, off otherwise)
  output_type     => NULL, -- optional; override the agents structured output
  ↳schema for this call. Default: NULL (use the agents own configuration)
  debug           => NULL -- optional; also emit every action as a NOTICE.
  ↳Default: NULL (false)
);
```

message	conversation_id	error
The orders table has 4,213 rows.	8f14e45f-ceea-467e-9575-a3d1a2c1a123	

(1 row)

Managing agents と同様に、これはエージェントロジックの障害不明なエージェント、ループ途中の推論の障害、超過した予算に対して発生することはありません。`error` は代わりに返された行に値が設定されるため、呼び出しのトランザクションおよび既にログに記録されているすべてその試行) はまだコミットします。

`output_type` を渡して、このコールのみにエージェントの構成済み出力スキーマをオーバーライドします。ビルドの方法については、[構造化された出力](#) を参照してください。

会話

すべての `agent_converse` コールは、`conversation_id` によって識別される会話に属します。最初の呼び出しでそれを省略すると、AIDB が生成します。エージェント自分自身の以前の履歴をコンテキストとして、後の呼び出しに同じ ID を渡して会話を続行します。

```
SELECT * FROM aidb.agent_converse(db_helper, And how many are still pending?,
  conversation_id => 8f14e45f-ceea-467e-9575-a3d1a2c1a123);
```

最初のプロンプトを送信する前に ID を用意しておきたい場合たとえば、UI セッションとともにログに記録する場合、`aidb.start_agent_session()` で前もって 1 つ作成します。

```
SELECT * FROM aidb.start_agent_session(db_helper);
SELECT * FROM aidb.agent_converse(db_helper, How many rows are in the orders table?,
  conversation_id => <the id start_agent_session returned>);
```

`start_agent_session` はエージェント名のみを検証し、新しい ID を返します。会話は、それを使用する最初の `agent_converse` 呼び出しまで、実際には存在しません。

会話の取得

`aidb.get_conversation()` は、会話のユーザープロンプトと最終的な回答を順番に結果ます。ツール呼び出しまたは中間推論ステップ、これらは「会話」ではなくアクティビティの詳細です。

```
SELECT * FROM aidb.get_conversation(8f14e45f-ceea-467e-9575-a3d1a2c1a123);
```

__OUTPUT__

message_id	task_id	action_type	role	sender_id	contents
...	...	user_prompt	user	postgres	How many rows are in the orders table?
...	...	answer	agent	db_helper	The orders table has 4,213 rows.

(2 rows)

`aidb.get_message()` は、`message_id` によって 1 つのメッセージのプレーンテキストコンテンツを取得します。どちらも `aidb.conversation_log` ビューによってサポートされており、直接照会することもできます。`aidb.conversations` は、会話ごとに 1 行メッセージ数、ステータス、最後のメッセージに集計します。完全な列リストについては、[Agents reference](#) を参照してください。

読み取り専用モード

`read_only => true` を渡して、何も永続化せずにエージェントを実行します。会話履歴は書き込まれ、`conversation_id` は常に NULL として返されるため、読み取り専用の実行は後で再開できません。データベースがリードレプリカの場合、これは自動です明示的に渡す必要はありません。

読み取り専用の実行は、一度に 2 つのことを強制します。すべてのツール呼び出しはそのステートメントの `SET LOCAL transaction_read_only = on` の下で実行され、読み取り専用ではないツールは、そもそもエージェントが表示または呼び出すことができるものから除外されます。MCP ツールは、分類方法に関係なく、常に読み取り専用モードで除外されます。外部サーバー自身の読み取り/書き込み動作は、AIDB が検証できるものではありません。読み取り専用の実行中に *Delegation* が発生した場合、デリゲート自分自身のサブ呼び出しも強制的に読み取り専用になります。

デバッグモード

`aidb_internal.action_log` に書き込まれる通常の履歴に加えて、代わりにではなく、すべてのアクションモデルの要求/応答、ツール呼び出しとその結果、推論ステップを Postgres NOTICE として出力するためにも `debug => true` を渡します。これは、エージェントが対話的に動作するのを監視したり、`psql` はデフォルトで NOTICE をターミナルに出力したり、そのパスを選択した理由をデバッグしたりするのに役立ちます。

```
SELECT * FROM aidb.agent_converse(db_helper, How many rows are in the orders table?,
↳ debug => true);
```

```
NOTICE: [tool_call] run_sql_query({"query": "SELECT count(*) FROM orders"})
```

```
NOTICE: [tool_response] run_sql_query -> [{"count": 4213}]
```

```
NOTICE: [answer] The orders table has 4,213 rows.
```

各 NOTICE には、そのアクションの要求/応答ペイロードプロンプト、ツール引数、および結果が含まれます。切り詰められますが、**編集されません**。これらの詳細には、独自のツールから返されたデータが含まれる場合があります、クライアント出力またはサーバーログを表示できる人に表示されないコンテキストでは、`debug => true` を避けてください。これは、`run` が行うデリゲートのサブコールにも適用されます。

予算と制限

各 `agent_converse` コールは 4 つのバジェットによって制限され、すべてオプションで `promoteb.create_agent` に設定されます。

Parameter	What it caps
<code>input_token_budget</code>	Input tokens for the call.
<code>output_token_budget</code>	Output tokens for the call.
<code>max_iterations</code>	Reasoning rounds — one model call plus any tools it calls in that round, not individual tool calls. A hard ceiling of 25 always applies, regardless of configuration, as a backstop against a runaway agent.
<code>timeout</code>	Wall-clock seconds for the call. Defaults to 300 seconds if unset.

予算を超過した場合に何が起こるかは、`budget_strategy` によって異なります。

Strategy	Behavior
<code>attempt_complete</code> (default)	予算を最初に超過すると、3 つの追加の推論ラウンドを付与、モデルの時間が残りわずかであることを伝えて、ラップアップします。次に、まだ終了していない場合はエラーで停止します。
<code>error</code>	エラーですぐに停止します。
<code>summarize</code>	すぐに停止し、モデルにこれまでに行ったことの簡単なステータスレポートを求め、それをエラーではなく回答として返します。
<code>ignore</code>	1 回限りの警告を Postgres <code>WARNING</code> としてログに記録し、制限なしで続行します。

コンテキストの長さ

長い会話は自動的に制限されます。タスクの組み立てられたプロンプトが内部で追跡されるサイズの推定値を超えると、AIDB はモデルに履歴の古い部分を要約するように求め、元のターンの代わりにその概要を続行します。これは、長時間実行される会話がモデルのコンテキストウィンドウを吹き飛ばさないようにするための基本的な自動バウンディングです。構成するものは何ともありません。これが発生すると、アクティビティログに `compaction` エントリが表示されます。会話の背後にある完全なアクティビティ履歴に

については、[デバッグモード](#) を参照するか、`aidb_internal.action_log` を直接照会します。すべてのモデルの要求/応答、およびツール呼び出し/応答、`aidb.get_conversation()` が返すユーザー向けメッセージだけでなく。

8.2.3 Delegation

エージェントは、ツールを直接呼び出す代わりに、またはそれとともに、要求の一部を別のエージェントに引き渡すことができます。[Managing agents](#) を行うときに、`delegates` に他のエージェントの名前をリストします。

```
SELECT aidb.create_agent(
  name          => sql_specialist,
  instructions => You answer questions about the orders and customers tables.
  ↳precisely.,
  model         => my_gpt,
  tools         => ARRAY[run_sql_query]
);

SELECT aidb.create_agent(
  name          => help_desk,
  instructions => You help users with general questions, delegating data questions.
  ↳to the SQL specialist.,
  model         => my_gpt,
  delegates     => ARRAY[sql_specialist]
);

SELECT * FROM aidb.agent_converse(help_desk, How many orders came in last week?);
```

モデルにデリゲートを提示する

構成された各デリゲートは、デリゲートエージェントにちなんで名前付けられた通常のファンクション呼び出しツールになり、2つの引数、`prompt` 必須-ハンドオフの要求と `conversation_id` オプション-以下の [デリゲートの会話を継続する](#) を参照してください。デリゲート自分自身の `instructions` はそのツールの説明になるため、委任モデルはそれらを「このデリゲートの目的」と見なします。委任されることが予想される場合は、その2つの目的を念頭に置いてエージェントの指示を作成します。

デリゲート名はエージェント自分自身の `tools` のいずれかと衝突してはならず、`tools` と同様に、`create_agent` /`update_agent` 時間に現在登録されているエージェントであることが検証されます。

デリゲートの会話を継続する

デリゲートを呼び出すと、`answer` および `conversation_id` を含むオブジェクトが返されます。モデルは、その `conversation_id` を同じまたは将来の `agent_converse` 実行での後のデリゲート呼び出しに戻して、毎回新しく開始するのではなく、独自の以前の履歴でその特定のデリゲートの会話を続行できます。それを省略すると一般的な場合、呼び出しのたびにデリゲートとの新しい会話が始まります。

ハンドオフを介した動作の伝播

- **読み取り専用モードが伝播します。**委任の実行が **読み取り専用モード** の場合、デリゲート自分のサブ呼び出しも強制読み取り専用です。読み取り専用エージェントは委任を使用して、別のエージェントをライトスルーに到達することはできません。読み取り専用のデリゲート呼び出しは常に `conversation_id: null` を返し、続行できません。
- **デバッグモードが伝播します。**トップレベル呼び出しが `debug => true` を渡した場合、デリゲート自分のアクションは NOTICE メッセージとしても出力されます。
- **ロールは伝播しません。**デリゲートは、委任エージェントのロールではなく、**独自の構成済み role** または現在のユーザーがない場合は実行されます。
- **予算は伝播しません。**各デリゲート呼び出しは、独自の構成から独自の新しいトークン/反復バジェットとタイムアウトを取得します。親の予算は、デリゲートができる量を制限しません。

委任の深さ

委任チェーンは、10 レベルの深さ (各エージェントが次のエージェントに委任するエージェントのチェーン) で制限されます。これは、主に Postgres のスタック深さチェックがバックエンドを中止するまで再帰するのではなく、サイクル—1 つのエージェントが自分自身に委任する、または 2 つのエージェントが前後に委任する—でフェイルファストするために存在します。制限に到達すると、タスクがエラーですぐに停止します。通常のツールの障害が発生する場合とは異なり、再試行されません。

第9章 Tools

ツールとは、エージェントが名前呼び出すことができるものです。AIDB が知っているすべてのツールビルトイン、カスタム、または MCP サーバーからインポートは、1 か所の `aidb.tools` にリストされています。

```
SELECT name, tool_type, read_only FROM aidb.tools;
```

__OUTPUT__

name	tool_type	read_only
chunk_text	native_tool	t
create_agent	native_tool	f
...		

ツールは 3 つの異なる方法で呼び出すことができます。

1. AIDB の内部エージェントによってネイティブに。
2. MCP を介した外部エージェントによる。
3. テストと開発またはその他のカスタムユースケースのための SQL を介して。

```
SELECT aidb.run_tool(catalog_list_objects, {"schema": "public", "object_type": "table"
→});
```

__OUTPUT__

```
run_tool
-----
{"comment": null, "truncated": false, "object_name": "my_table", "object_type":
→"table", "schema_name": "public"}, ...
```

AIDB には 3 種類のツールがあります。

- `native_tool` - AIDB 独自のビルトインカタログ。いつでも利用できます。追加、削除、または再構成はできません。 [Native tools](#) を参照してください。
- `sql_tool` - 自分で登録するパラメーター化された SQL クエリー。 [Custom SQL tools](#) を参照してください。
- `mcp_tool` - 外部 MCP サーバーからインポートされたツール。 [MCP tools](#) を参照してください。

`promoteb.create_agent` の `tools` 配列に名前をリストすることにより、エージェントにツールへのアクセスを付与します。

9.1 ドキュメントマップ

Page	What it covers
<i>Native tools</i>	AIDB's built-in tool catalog: ad hoc SQL, catalog discovery, semantic knowledge bases, models, volumes, embedding and document processing, agent management, pipelines
<i>Custom SQL tools</i>	<code>aidb.create_sql_tool()</code> , naming, parameter placeholders, read-only enforcement, removing tools
<i>MCP tools</i>	<code>aidb.import_mcp_tools()</code> , authentication, filtering, refreshing, removing servers, exposing AIDB's tools over MCP

完全なパラメーターテーブルと戻り型については、[Tools reference](#) を参照してください。

9.1.1 Native tools

AIDB には、ネイティブツールの大規模なビルトインカタログが同梱されています。ほとんどが既存の AIDB 機能の薄いラッパーであるため、エージェントは SQL から直接操作するのと同じ方法でパイプライン、ナレッジベース、モデル、ボリュームを操作できます。これは、AIDB がサポートする 3 つのツールタイプの 1 つです。[Custom SQL tools](#) および [MCP tools](#) を含む完全なカタログについては、[Tools reference](#) を参照してください。`sleep` は、推論ループ自身の一時停止と再開の機能をサポートするため、バックギンモデルが要求で伝送できるツールの数を制限し、エージェントの他の設定済みツールが既にそのキャップを満たしている場合を除き、設定された `tools` リストに関係なく、すべてのエージェントが常に利用できます。他のすべてのネイティブツールを使用するには、`tools` に明示的にリストする必要があります。

ツールの正確なパラメーターは `SELECT params FROM aidb.tools WHERE name = '<name>'` で常に可視されます。以下のテーブルに、それぞれの 1 行の説明と読み取り専用ステータスを示します。

アドホック SQL

Tool	Read-only	Description
<code>run_sql_query</code>	Yes	データベースに対して読み取り専用の SQL クエリーを実行し、その行を JSON として結果ます。

カタログディスカバリ

データベースカタログのイントロスペクションこれらのツールを使用すると、エージェントは `information_schema/pg_catalog` クエリ自分自身を構成せずに、どのようなスキーマ、テーブル、ビュー、およびファンクションが存在するかを検出できます。

これらは以下の [Semantic knowledge bases](#) に関連していますが、2 つは互換性のあるものではなく、補完的なものです。

- **カタログ検出ツール** は、Postgres カタログに対してライブで正確なルックアップを行います。セットアップは必要なく、常にデータベースの現在の状態を反映し、正確な名前または SQL LIKE パターンで一致します。

- **セマンティックナレッジベースツール**は、スキーマのクロールされたスナップショットに対してセマンティックベクター検索を実行します。事前に構築されたオプティムのナレッジベースと、構造の変更を検出するための再クロールが必要ですが、「顧客の連絡先情報を保持しているテーブルはどれですか?」のような自然言語の質問に答えることができます。その正確なカタログ検索ではできません。

一般的なエージェントは、あいまいな説明から適切なテーブルを見つけるためのセマンティック検索と、その正確な現在の列と外部キーを取得するためのカタログディスカバリの両方を使用します。

4つのツールはすべて特権認識であり、呼び出したユーザーが実際に見ることができるオブジェクトのみを結果ます。リストは、Postgres と AIDB の内部スキーマ、およびインストールされた拡張機能のスキーマとオブジェクトをデフォルトで非表示にします `catalog_get_object_details` は意図的に拡張機能オブジェクトを非表示にします。明示的に名前を付けることがそれを求めています。リストの行数は 200 行に制限されています。すべての行には `truncated` フラグがあり、それ以上の行が存在する場合は `true` であり、`schema` または `name_pattern` で検索を絞り込むように呼び出し元に通知します。オプションのテキストパラメーターは、空の文字列を省略された引数と同じように扱います。

Tool	Read-only	Description
<code>catalog_list_schemas</code>	Yes	現在のデータベースのスキーマをリストします。 `include_system` が設定されていない限り、Postgres / AIDB 内部スキーマとインストールされた拡張機能のスキーマを非表示にします。
<code>catalog_list_objects</code>	Yes	リストテーブル、ビュー、マテリアライズドビュー、外部テーブル、および/またはファンクション。 <code>schema</code> 、 <code>object_type</code> 、および/または <code>name_pattern</code> で絞り込みます。
<code>catalog_get_object_details</code>	Yes	名前によって 1 つのオブジェクトの完全な詳細を取得します。リレーションの列、プライマリ/一意/外部キー。ファンクションの引数と戻り値の型。
<code>catalog_list_relations</code>	Yes	外部キー関係のリスト — スキーマ内のすべての FK、または特定のテーブルに接触するエッジのみ、いずれかの方向に。

セマンティック知識ベース

スキーマ検索ナレッジベースにより、テーブル、列、およびその他の DB オブジェクトのセマンティック検索ベースの検出が可能になります。パイプラインが作成できるユーザーデータのナレッジベースとは異なります。

Tool	Read-only	Description
<code>create_semantic_kb</code>	No	自然言語スキーマ検索用に 1 つ以上のスキーマに対してセマンティックナレッジベースを作成します。

次のページに続く

表 4 – 前のページからの続き

Tool	Read-only	Description
<i>delete_semantic_kb</i>	No	セマンティックナレッジベースを名前で削除します。
<i>list_semantic_kbs</i>	Yes	登録されているすべてのセマンティックナレッジベースをリストします。
<i>get_semantic_kb</i>	Yes	登録されたセマンティックナレッジベースの構成を名前で取得します。
<i>refresh_semantic_kb</i>	No	セマンティック知識ベースのスキーマを再クロールして、構造の変更を検出します。
<i>update_semantic_kb_auto_</i> <i>get_column_definitions</i>	No	セマンティック知識ベースの自動処理モードを変更します。
<i>get_metadata</i>	Yes	セマンティックナレッジベースを検索して、一致する列/テーブル定義を見つけます。
<i>get_entity_definitions</i>	Yes	セマンティックナレッジベースを検索して、一致するテーブル/ビュー定義を見つけます。
<i>semantic_kb_search</i>	Yes	ナレッジベースで複合セマンティック検索を実行し、ソースタイプの識別子と一致するテーブル、ビュー、列、およびエイリアスのランク付けリストを 1 つ結果ます。
<i>search_by_comment</i>	Yes	セマンティックナレッジベースのスキーマ/テーブル/列コメントを類似性によって検索します。
<i>semantic_kb_stats</i>	Yes	セマンティックナレッジベースの行/エンティティ数を取得します。

モデル

Integrating models を参照してください。

Tool	Read-only	Description
<i>list_models</i>	Yes	登録されているすべてのモデルをリストします。
<i>get_model</i>	Yes	登録されたモデルの構成を名前で取得します。
<i>create_model</i>	No	新しいモデルをプロバイダーと構成に登録します。
<i>delete_model</i>	No	登録されたモデルを名前で削除します。
<i>sync_hcp_models</i>	No	使用可能なモデルを、構成済みのハイブリッドマネージャー HM からモデルレジストリに同期します。
<i>list_hcp_models</i>	Yes	HM から既に同期されたモデルをリストします。
<i>get_hcp_models</i>	Yes	HM から利用可能なモデルをリストします、同期されているかどうか。

次のページに続く

表 5 – 前のページからの続き

Tool	Read-only	Description
<i>create_hcp_model</i>	No	HM からローカル名でモデルを登録します。

ボリューム

[Configuring external storage](#) を参照してください。

Tool	Read-only	Description
<i>list_volumes</i>	Yes	登録されているすべてのボリュームをリストします。
<i>create_volume</i>	No	オブジェクトストアによってバックアップされた新しいボリュームを登録します。
<i>delete_volume</i>	No	名前でボリュームを削除します。
<i>list_volume_content</i>	Yes	ボリュームに保存されているファイルをリストします。
<i>read_volume_file</i>	Yes	ボリュームからファイルの内容を読み取ります。
<i>write_volume_data</i>	No	ボリューム内のファイルにデータを書き込みます。
<i>delete_volume_file</i>	No	ボリュームからファイルを削除します。

埋め込みとドキュメント処理

[SQL functions](#) を参照してください。

Tool	Read-only	Description
<i>encode_text</i>	Yes	ドキュメント/検索インデックスストレージのテキスト埋め込みベクトルを計算します。
<i>encode_text_query</i>	Yes	クエリー対インデックス用に調整されたテキスト埋め込みベクトルを計算します。
<i>encode_image</i>	Yes	画像埋め込みベクトルを計算します。
<i>generate_text</i>	Yes	言語モデルからテキストを生成します。
<i>get_model_description</i>	Yes	モデルの人が判読可能な説明を取得します。
<i>list_model_providers</i>	Yes	モデルの登録に使用できる外部サーバーバックアップのモデルプロバイダーをリストします。
<i>rerank_text</i>	Yes	リランカーモデルを使用して、クエリーに対して候補テキストのリストを再ランク付けします。
<i>get_adapter_embedding</i>	Yes	テキストまたは画像入力のモデルの出力埋め込み次元を取得します。
<i>chunk_text</i>	Yes	テキストを埋め込みに適したチャンクに分割します。

次のページに続く

表 7 – 前のページからの続き

Tool	Read-only	Description
<i>parse_html</i>	Yes	HTML をプレーンテキストまたはマークダウンに解析します。
<i>parse_pdf</i>	Yes	PDF からページごとに 1 行でテキストを抽出します。
<i>pdf_to_image</i>	Yes	PDF のページを画像にレンダリングします、ページごとに 1 行。
<i>perform_ocr</i>	Yes	画像またはスキャンしたドキュメントで OCR を実行します。
<i>summarize_text</i>	Yes	言語モデルを使用してテキストを要約します。

エージェント管理

Managing agents を参照してください。エージェントに **他の** エージェント自分自身を含むを作成または再構成させることは、強力な自己変更機能です。これらのツールに意図的に付与します。

Tool	Read-only	Description
<i>list_agents</i>	Yes	登録されているすべてのエージェントをリストします。
<i>create_agent</i>	No	新しいエージェントを作成します。`error` 列成功した場合は `NULL` を含む単一の行を結果ます。
<i>update_agent</i>	No	既存のエージェントの構成を更新します。明示的に渡されたフィールドのみが変更されます。
<i>delete_agent</i>	No	名前でエージェントを削除します。`error` 列成功した場合は `NULL` を含む単一の行を結果ます。
<i>sleep</i>	Yes	推論を数秒間一時停止し、自動的に再開します。モデル自身のツールカウントキャップがエージェントの他の構成済みツールによって既に埋められている場合を除き、設定されたツールリストに関係なく、すべてのエージェントが常に利用できます。

パイプライン

AI pipelines を参照してください。

Tool	Read-only	Description
<i>search_pipelines</i>	Yes	登録されているすべてのデータパイプラインをリストします。
<i>get_pipeline</i>	Yes	登録されたデータパイプラインの構成を名前取得します。

次のページに続く

表 9 – 前のページからの続き

Tool	Read-only	Description
<code>delete_pipeline</code>	No	登録されたデータパイプラインを名前で削除します。
<code>run_pipeline</code>	No	パイプラインを1回手動で実行し、保留中のソース行を処理します。
<code>get_pipeline_metrics</code>	Yes	パイプラインの処理ステータスと行数を取得します。
<code>get_error_logs</code>	Yes	パイプラインのレコードレベルの処理エラーをリストします。オプションでフィルタリング。
<code>clear_error_logs</code>	No	IDに基づいて特定のパイプラインエラーログエントリを削除します。
<code>re-queue_pipeline_errors</code>	No	特定のパイプラインエラーのソース行をダーティとしてマークして、次回の実行で再試行します。
<code>get_error_log_summary</code>	Yes	パイプラインのエラー数をステップ、オペレーション、カテゴリごとにグループ化して取得します。
<code>get_all_error_summary</code>	Yes	すべてのパイプラインのエラー数をステップ、オペレーション、カテゴリごとにグループ化して取得します。

!!!note パイプラインオーサリング `aidb.create_pipeline/aidb.update_pipeline` にはネイティブツールがありません。これらのファンクションは、ステップ/ステップオプションのペア全体で最大 30 のパラメーターを取得するため、単一の汎用ツール呼び出しには非現実的であるため、現在マルチステップのパイプラインオーサリングはエージェントが自分自身で実行できるものではありません。

9.1.2 Custom SQL tools

SQL ツールは、エージェントが名前呼び出すことができる保存されたパラメーター化クエリです。これを使用して、Rust を書いたり、ネイティブ AIDB ファンクションを公開したりせずに、エージェントに独自のテーブルとビジネスロジックへのアクセスを許可します。

```
SELECT aidb.create_sql_tool(
  name           => orders_by_customer,
  description     => Look up recent orders for a customer by id.,
  sql_statement   => SELECT id, status, total FROM orders WHERE customer_id =
  ↳ ${customer_id} ORDER BY created_at DESC LIMIT ${limit},
  params         => aidb.tool_params(
    aidb.tool_param(customer_id, INT, The customers id.),
    aidb.tool_param(limit, INT, Maximum number of orders to return.)
  ),
  read_only      => true, -- optional; enforced, not just descriptive (see
  ↳ below). Default: true
  return_type_hint => NULL -- optional; documents the returned row shape, same
  ↳ aidb.tool_params() shape as `params`. Default: NULL
```

(次のページに続く)

(前のページからの続き)

);

Parameter	Type	Default	Description
<i>name</i>	TEXT	Required	ツールの一意の名前。
<i>description</i>	TEXT	Required	ツール定義の一部としてモデルに表示されます。
<i>sql_statement</i>	TEXT	Required	実行するクエリー。宣言されたパラメーターごとに`\${name}`プレースホルダーを使用します。
<i>params</i>	JSONB	Required	`aidb.tool_params()`で構築されたツールのパラメーター。
<i>read_only</i>	BOOLEAN	<i>true</i>	`true`の場合、読み取り専用であることが証明されていない限り、登録時にステートメントを拒否します。
<i>return_type_hint</i>	JSONB	<i>NULL</i>	クエリーの結果の形状を同じ`aidb.tool_params()`形状で文書化します。現在モデルには表面化されていません。

`aidb.tool_param(name, type, description)` は 1 つのパラメーターをビルドします。 `aidb.tool_params(...)` は、いくつかを組み合わせます。引数なしの `aidb.tool_params()` は、パラメーターをとらないツールを宣言します。

他の方法と同じようにエージェントにツールを登録します。

```
SELECT aidb.create_agent(sales_helper, Answer questions about customer orders., my_
→gpt,
    tools => ARRAY[orders_by_customer]);
```

ネーミング

ツールの名前は、すべてのツールタイプで一意である必要があります。ビルトイン *Native tools* または *MCP tools* から既にインポートされたツールと衝突してはなりません。 `aidb.create_sql_tool()` は、その場合に呼び出しを前もって拒否します。

パラメータープレースホルダー

`sql_statement` で宣言された各パラメーターを`\${name}`として参照する — AIDB は、その呼び出しにモデルが提供した値、実際のクエリパラメーター文字列補間されていないとしてバインドされた値にそれぞれを置き換えます。ステートメントで複数回使用されるパラメーターは、1 回だけ宣言する必要があります。 `params` は必須です。パラメーターをとらないツールの場合、引数なしで `aidb.tool_params()` を渡します。

読み取り専用ツール

`read_only` のデフォルトは `true` です。書き込む必要があるツールに `read_only => false` を渡します。そして、それは両方向に説明的なだけでなく適用されます。デフォルトの `true` のままでは、`sql_statement` が認識された書き込みパターン `INSERT`、`UPDATE`、`DELETE` などと一致する場合、登録はすぐに失敗します。誤ったラベルのツールを通過させるのではなく、デフォルトの `read_only => true` のままになっているツールは、**読み取り専用モード** で実行されているエージェントでも使用できます。`read_only => false` と明示的にマークされたツールは、未分類のネイティブまたは MCP ツールと同様に、読み取り専用の実行から除外されます。

`sql_statement` は単一のステートメントである必要があり、`EXPLAIN` や `COPY` のようなユーティリティステートメントはできません。後続の空白と後続の `;` は、登録時に自動的に削除されます。

!!!note `create_sql_tool` は、`params` と同じ `aidb.tool_params()` 形状の `return_type_hint` 引数も受け入れ、クエリーが返す行の形状を説明します。これはツールとともに保存されますが、現在モデルまたは `aidb.tools` には表示されません。今のところ、その唯一の使用は、`ailedb.sql_tool_registry` を直接検査する人のためのドキュメントとしてです。

ツールの削除

```
SELECT aidb.delete_tool(orders_by_customer);
```

`aidb.delete_tool()` は、渡した名前が SQL ツールではなくに属している場合、*MCP tools* も削除しますが、ビルトインネイティブツールは削除できません。

9.1.3 MCP tools

MCP モデルコンテキストプロトコルは、サーバーが呼び出し可能なツールセットをアダプタイズする標準の方法です。`aidb.import_mcp_tools()` は、外部 MCP サーバーを登録し、アダプタイズするツールをインポートするため、エージェントは AIDB 独自のネイティブツールおよび SQL ツールとともにこれら呼び出すことができます。

```
SELECT aidb.import_mcp_tools(
  name      => weather,
  url       => https://weather.example.com/mcp,
  transport => streamable_http, -- optional; streamable_http or sse (see below).
  ↳Default: streamable_http
  headers   => NULL,           -- optional; request headers to send, for
  ↳example, an auth token. Mutually exclusive with headers_env. Default: NULL
  tool_filter => NULL,         -- optional; import only these tool names.
  ↳Default: NULL (import everything the server advertises)
  headers_env => NULL          -- optional; env var holding the headers JSON
  ↳instead of storing it. Mutually exclusive with headers. Default: NULL
);

SELECT aidb.create_agent(trip_planner, Help plan trips, checking weather when useful,
  ↳ my_gpt,
  tools => ARRAY[get_forecast]); -- a tool the weather server advertised
```

Parameter	Type	Default	Description
<i>name</i>	TEXT	Required	サーバー登録の一意の名前。
<i>url</i>	TEXT	Required	サーバーの MCP エンドポイント URL。
<i>transport</i>	TEXT	'streamable_http'	'streamable_http' または 'sse' — 現在 'streamable_http' のみが完全にサポートされています。
<i>headers</i>	JSONB	NULL	送信する要求ヘッダーたとえば {"Authorization": "Bearer ..."} 'headers_env' と相互排他的。
<i>tool_filter</i>	TEXT	NULL	これらのツール名のみをインポートします。サーバーがアダプタイズするすべてのインポートを省略します。
<i>headers_env</i>	TEXT	NULL	ヘッダー JSON を保持する環境変数の名前。保存するのではなく、使用するたびに新しく読み取られます。'headers' と相互排他的。

`import_mcp_tools` は、何かを登録する前に `tools/list` エンドポイントを呼び出すことによりサーバーを検証します。到達不能であるか、`tool_filter` と一致するものをアダプタイズしない場合、何も保存されません。返されるツールはすぐにキャッシュされるため、別の更新後ではなく、すぐに `aidb.tools` に表示されます。

現在、`streamable_http` トランスポートデフォルトのみが実際にエンドツーエンドで動作します。`sse` は値として受け入れられますが、まだ完全には実装されていません。

MCP サーバーへの到達は、アウトバウンドネットワーク呼び出しであり、モデルプロバイダー呼び出しおよび HuggingFace のダウンロードと同じ `aidb.egress_allowlist` 制限の対象となります。

認証

特定の要求ヘッダー最も一般的には `Authorization` ベアラートークンを必要とするサーバーに `headers` を渡します。

```
SELECT aidb.import_mcp_tools(
  name    => internal_crm,
  url     => https://crm.internal.example.com/mcp,
  headers => {"Authorization": "Bearer sk-..."}::JSONB
);
```

`headers` は `aidb.mcp_registry` に保存されますが、`aidb_users` にはその列への `SELECT` アクセスがありません。AIDB 自身の内部呼び出し検証、リフレッシュ、またはサーバーのツールの呼び出しのみがそれを読み取ることができます。

!!!note シークレットをテーブルにまったく保存したくないですか？代わりに `headers_env` を渡します。`headers` が取るのと同じ形状で、ヘッダーの JSON オブジェクトを保持する環境変数名前付けます呼び出

しごとに Postgres プロセス自分の環境から新しく読み取られます。`headers` および `headers_env` は相互排他的です。変数の名前は、`aidb.env_var_allowed_prefix` デフォルトでは `AIDB_` で構成されたプレフィックスで始まる必要があります。これは、モデル資格情報に `credentials_env` で使用される同じメカニズムと GUC です。

インポートするツールのフィルタリング

デフォルトでは、サーバーがアダプタイズするすべてのツールがインポートされます。`tool_filter` を渡して、特定のもののみを名前でもインポートします。

```
SELECT aidb.import_mcp_tools(
    name      => weather,
    url       => https://weather.example.com/mcp,
    tool_filter => ARRAY[get_forecast, get_alerts]
);
```

ツールリストを最新の状態に保つ

AIDB は、すべてのクエリーでサーバーのツールリストを再取得しません。`aidb.tools` は、過去 60 分以内にキャッシュされたエントリのみを表示するため、`SELECT` にネットワーク副作用が発生することはありません。`aidb.refresh_mcp_tools()` を呼び出して、オンデマンドでキャッシュを再取得および更新しますたとえば、サーバーが新しいツールを追加した後。

```
SELECT aidb.refresh_mcp_tools(weather);
```

現在キャッシュされているツールの数を返します `tool_filter` 適用後サーバーの登録—その `url`、`transport`、`headers`、または `tool_filter` を再実行する `import_mcp_tools` 同等の設定を介して変更することも、キャッシュを自動的に無効にします。

サーバーの削除

```
SELECT aidb.delete_tool(weather);
```

MCP サーバーを削除すると、それに伴ってアダプタイズされたすべてのツールが削除されます。ツールの 1 つだけを個別に削除する方法はありません。`tool_filter` を使用して、代わりに公開されるツールをキュレートします。

MCP を介して AIDB 独自のツールを公開する

`aidb.get_mcp_tools()` は、`aidb.tools` に現在あるすべてのツールネイティブ、SQL、およびインポートされた MCP ツールを同様に返し、MCP `tools/list` 行項目として再形成します。他のシステムが MCP 独自の記述子形式で AIDB のツールカタログを必要とする場合に役立ちます。

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

これは、AIDB のツールカタログを MCP の記述子形状に変換するだけです。AIDB は、外部 MCP クライアントが接続し、ネットワーク経由でこれらのツールを実際に呼び出すことができる独自の MCP サーバーエンドポイントを実行しません。

第10章 AI pipelines

パイプラインは、セマンティック検索および RAG アプリケーション用のデータを取り込み、準備、およびベクトル化できます。[pgvector 反復インデックススキャン](#) 検索機能を統合し、ナレッジベースをソースデータと自動的に同期し続けることができます。

10.1 例 RAG のナレッジベース

このウォークスルーでは、内部サポート記事を介してナレッジベースを構築し、各取得ステージを独立して示します。完全なバージョンについては、[エンドツーエンドの例 RAG のナレッジベース](#) を参照してください。これは、モデルの登録、クロスエンコーダーが間違ったベクトル検索結果を修正するリランキングデモ、および検証された出力を含む回答を取得、リランク、および生成する単一のステートメントを追加します。

10.1.1 パイプラインのセットアップ

ソースは、データベース操作に関する 16 の短い記事を保持する `support_articles` テーブルです。完全な例の [source data step](#) から `CREATE TABLE` および `INSERT` ステートメントを取得します。パイプラインは、各記事の本文を `bge-small-en-v1.5-f16`、ビルトイン `llama.cpp` 埋め込みモデルの 1 つで埋め込みます。事前登録され、ローカルで実行され、最初に使用するときにモデルファイルが自動的にダウンロードされます。

```
SELECT aidb.create_pipeline(
  name          => support_kb,
  source        => support_articles,
  source_key_column => id,
  source_data_column => body,
  step_1        => KnowledgeBase,
  step_1_options => aidb.knowledge_base_config(
    model      => bge-small-en-v1.5-f16,
    data_format => Text
  ),
  auto_processing => Disabled
);
```

10.1.2 パイプラインを実行する

auto_processing => 'Disabled' では、あなたが言うまで何も実行されません。既存のすべての行を 1 回処理します。

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

10.1.3 Retrieve セマンティック検索

すべての記事本文が埋め込まれ、インデックスが作成されます。生のエラーテキストをセマンティック検索に貼り付けます。最上位の一致は解像度であり、キーワードではなく意味によって見つけられます。

```
`sql SELECT key, value, distance::numeric(5,3) AS distance FROM aidb.
retrieve_text('support_kb', 'application logs show: FATAL: sorry,
too many clients already', 3); __OUTPUT__ key | value | distance
-----+-----
# AI Pipelines`
```

<i>Pipeline steps</i>	Available step types: ChunkText, ParseHtml, ParsePdf, PerformOcr, Summarize-Text, KnowledgeBase.
<i>Orchestration</i>	Auto-processing modes, background workers, observability, and error handling.
<i>AI pipelines</i>	Full API reference for pipeline types, views, CRUD functions, and config helpers.
<i>Examples</i>	End-to-end worked examples.

Creating pipelines

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

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

Parameter group	Parameters	More information
Source	source, source_key_column, source_data_column	This page
Steps	step_1 ... step_10, step_N_options	Pipeline steps
Orchestration	auto_processing, background_sync_interval, batch_size	Orchestration

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
Destination	<i>destination</i>	<i>destination_key_column</i> and <i>destination_data_column</i> are not inputs — they're reported back in the function's return value, derived automatically from the destination type and steps

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)
);
```

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,
  url       => s3://my-bucket/my-folder,
  options   => {"region": "us-east-1"}
);
```

ステップ 2 ボリュームを作成する

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

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

ステップ 3 ボリュームをパイプラインソースとして参照する

source パラメーターをボリューム名に設定します。

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

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

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

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。

中間ストレージ

すべてのステップは、自分の出力を次のステップまたは最後のステップの場合、パイプラインの最終宛先に渡すことに加えて、自分自身の出力を専用のテーブルに永続化できます。これは、マルチステップパイプラインのデバッグ、またはパイプラインの外部で中間結果チャンクテキストなどを再利用する場合に役立ちます。デフォルトではオフになっています。

そのステップの `step_N_options` の `intermediate_destination` キーで設定します。ステップ構成ヘルパー `aidb.chunk_text_config()`、`aidb.pdf_parse_config()` などにはまだパラメーターがないため、ヘルパー自分の出力とともにプレーン JSON として追加し、`||` 演算子とマージします。

```
step_1_options => aidb.chunk_text_config(desired_length => 100) || jsonb_build_object(
  intermediate_destination, jsonb_build_object(enabled, true)
)
```

または、`jsonb_build_object()` を使用してオプションオブジェクト全体を直接ビルドします。

```
step_1_options => jsonb_build_object(
  desired_length, 100,
  intermediate_destination, jsonb_build_object(enabled, true)
)
```

Key	Type	Default	Description
<i>enabled</i>	BOOLEAN	Required	このステップの中間ストレージをオンにします。
<i>destination</i>	TEXT	NULL	ステップの出力を保存するテーブル。省略すると <code>`pipeline_<name>_step_<N>`</code> として自動生成されます。

最も単純なフォームは、それをオンにして、AIDB にテーブルに名前を付けます。

```
{"enabled": true}
```

または、宛先テーブルに明示的に名前を付けます。

```
{"enabled": true, "destination": "my_custom_chunk_storage"}
```

宛先テーブルは、他のパイプライン宛先テーブル `source_id`、`part_ids`、およびステップの出力を保持する `value` 列と同じ形状です。

パイプラインの **last** ステップで `intermediate_destination` を設定しても、テーブルを作成してそれに書き込みますが、`promoteb.retrieve_text()` の `intermediate_steps` 列がそれを表面化することはありません。そのルックアップは、最後の「次のステップ」がないため、最後のステップの前のステップのみを考慮します。自分の出力が中間になる前に。読み取る必要がある場合は、代わりに宛先テーブルを直接照会します。これは KnowledgeBase にとって最も重要であり、これは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。したがって、中間ストレージを有効にする理由はほとんどありません。

!!!note `intermediate_destination` は、`enabled` キーおよびオプションで `destination` を持つオブジェクトである必要があります。他の JSON タイプは、パイプラインの作成時にエラーを発生します。

```
ERROR: Invalid options for step 1: intermediate_destination must be an object with
↳keys enabled and optionally destination; got a string
```

ChunkText

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

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

Parameter	Type	Default	Description
<code>desired_length</code>	<code>INTEGER</code>	Required	ターゲットチャンクサイズ。単位は `strategy` に依存します。
<code>max_length</code>	<code>INTEGER</code>	<code>NULL</code>	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。
<code>overlap_length</code>	<code>INTEGER</code>	<code>NULL</code>	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。
<code>strategy</code>	<code>TEXT</code>	<code>NULL</code>	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。

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 <code>'StructuredPlaintext'</code> デフォルト、または階層を保持する場合は <code>'StructuredMarkdown'</code> 。

```
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)
);
```

結果

ParseHtml は結果のネストを解除しません。各ソース行は、元の行の対応関係が保持された状態で、抽出されたテキストを含む出力行を 1 つだけ生成します。ChunkText とペアにして、上記の例のように、抽出されたテキストをより小さなセグメントに分割します。

ParsePdf

非標準または複雑なファイルを処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>	<code>'Structured'</code> 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>	<code>'true'</code> デフォルトの場合、個々のページでエラーが発生した場合に解析を続行します。

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

```

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

```

結果

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

PdfToImage

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

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

オプション形式 オプションを JSON リテラルとして渡します。

Parameter	Type	Default	Description
<i>dpi</i>	<i>INTEGER</i>	<i>300</i>	各ページのレンダリングに使用される 1 インチあたりのドット数の解像度。値が高いほど、より鮮明な画像が生成されますが、ファイルサイズが大きくなります。
<i>first_page</i>	<i>INTEGER</i>	<i>NULL</i>	レンダリングする最初のページ 1 ベース。省略すると最初からレンダリングされます。
<i>last_page</i>	<i>INTEGER</i>	<i>NULL</i>	レンダリングする最後のページ。1 ベースを含みます。省略すると最後までレンダリングされます。

次のページに続く

表 7 – 前のページからの続き

Parameter	Type	Default	Description
<i>max_pages</i>	<i>INTEGER</i>	<i>NULL</i>	レンダリングするページの最大数。`first_page`および`last_page`に関係なく、安全キャップとして機能します。
<i>render_annotations</i>	<i>BOOLEAN</i>	<i>true</i>	レンダリングされた出力にPDFの注釈コメントやフォームフィールドなどを含めるかどうか。
<i>format</i>	<i>JSONB</i>	<i>{"type": "png"}</i>	出力画像フォーマット。受け入れられる値 <i>{"type": "png"}</i> 、 <i>{"type": "jpeg"}</i> 。出力を `PerformOcr` に渡す場合は、 <i>{"type": "png"}</i> を使用します。

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

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

```
SELECT aidb.create_model(
  my_paddle_ocr_model,
  nim_paddle_ocr,
  credentials => {"api_key": "<NVIDIA_NIM_API_KEY>"}::JSONB -- or credentials_env =>
  ↳to read this from an environment variable instead
);
```

次に、パイプラインを作成します。

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

結果

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

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 -- or credentials_env_
→to read this from an environment variable instead
);
```

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。

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。

ヘルパーファンクション `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するか、 <code>`reduce`</code> はそれらを反復圧縮します。
<i>reduction_factor</i>	<i>INTEGER</i>	<i>NULL</i>	<code>`reduce`</code> ストラテジーで使用されます。反復ごとにテキストをどの程度積極的に削減するかを制御しますデフォルトは3です。
<i>inference_config</i>	<i>JSONB</i>	<i>NULL</i>	オプションの実行時推論設定 <code>`aidb.inference_config()`</code> から。

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

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
<code>model</code>	<code>TEXT</code>	Required	埋め込みモデルの名前。
<code>data_format</code>	<code>aidb.PipelineDataFor</code>	Required	<code>'Text'</code> または <code>'Image'</code> 。
<code>distance_operator</code>	<code>aidb.DistanceOperator</code>	<code>NULL</code>	類似度メトリック <code>L2</code> デフォルト、 <code>'InnerProduct'</code> 、 <code>'Cosine'</code> 、 <code>'L1'</code> 、 <code>'Hamming'</code> 、または <code>'Jaccard'</code>
<code>vector_index</code>	<code>JSONB</code>	<code>NULL</code>	<code>'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。</code>

```
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)
  )
);
```

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

宛先テーブル

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

Column	Type	Description
<i>id</i>	BIGSERIAL	主キー。
<i>pipeline_id</i>	INT	元のパイプラインへの参照。
<i>source_id</i>	TEXT	元のソースレコードの ID。
<i>part_ids</i>	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。
<i>value</i>	VECTOR	pgvector 埋め込み。

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

完全なエンドツーエンドのワークフローと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。

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。

- **** 自動処理を有効にする **** パイプラインはバックグラウンドで実行するか、データと同期したままにすることができます。バッチサイズとトリガーを構成し、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。
- **** Background workers **** 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、文書解析、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** 彼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でもアプリの高速性が保証されます。
- **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 などが一時的に停止している場合に自動的に再試行します。ダウン。
- **** モニタリングとオブザーバビリティ **** 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.
- **** Pipeline error log **** 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。SQL サーフェスから検査、クリア、および再キューイングします。

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。

Auto-processing

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

主な機能

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

自動処理モード

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

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

構成

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

パイプライン作成時のモードの設定

```
SELECT aidb.create_pipeline(
    name                => my_pipeline,
```

(次のページに続く)

(前のページからの続き)

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

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。

Observability

すべての自動処理モードの詳細なステータスと進行状況の出力を確認します。

2つのビューがオンデマンドの可観測性のニーズのほとんどをカバーします。それぞれがクエリ時に数値を計算します。

- `promoteb.pipeline_metrics` エイリアス `aidb.pipem` — バックログ、ソース/宛先カウント、ステータス、エラーカウントを含むパイプラインごとに 1 行。これは、処理の進行状況を追跡するためのプライマリビューです。

- `helpb.Knowledge_base_metrics` エイリアス `aiddb.kbm` —パイプライン数、埋め込み数、および接続されているすべてのパイプラインの集計ステータスを含むナレッジベースごとに 1 行。

オンデマンドメトリック

パイプラインごとの進行状況

```
SELECT * FROM aidb.pipem;

__OUTPUT__

      pipeline      | auto processing | table: unprocessed rows | volume: scans
───────────────────+──────────────────+──────────────────+──────────────────
→ completed | count(source records) | count(destination records) | Status
→ count(record errors) | count(blocking errors) | last run completed
──────────────────+──────────────────+──────────────────+──────────────────
→
→
feedback_na_pipe    | Live           | 0 |
                    | 412           | 412 | UpToDate |
0 |                0 | 2026-07-21 10:14:58.001+00
product_docs_pipe   | Background     |
6 |                1842 | 1842 | UpToDate |
0 |                0 | 2026-07-21 10:10:02.774+00

(2 rows)
```

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

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

ナレッジベースの概要

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

__OUTPUT__

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

(2 rows)

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

つけます。

マテリアライズされたメトリック

`aidb.pipeline_metrics` および `aidb.knowledge_base_metrics` は、照会時にオンデマンドで数値を計算します。外部モニタリング-たとえば、EDB Postgres AI エージェントを介した Prometheus スクレイピングの場合、一定の間隔でこれらのビューをポーリングすることは、すべてのスクレイピングですべてのパイプラインの統計を再計算することを意味し、各パイプライン所有者の基になるテーブルに到達するにはスクレイピングロールが必要です。

`aidb.pipeline_metrics_collection` は両方のコストを回避します。バックグラウンドワーカーは、すべてのパイプラインから同じ統計のスナップショットを作成し、そのパイプラインの所有ロールとして照会し、15 秒ごとにこのテーブルに書き込みます。テーブルへのアクセスが付与されたロール—`pg_monitor` はデフォルトで `SELECT` を取得します—は、何も再計算せず、パイプラインごとの権限を必要とせずに、事前計算されたメトリックを読み取ることができます。

```
SELECT * FROM aidb.pipeline_metrics_collection;
__OUTPUT__
  pipeline      | auto processing | table: unprocessed rows | volume: scans_
→completed | count(source records) | count(destination records) | count(record_
→errors) | count(blocking errors) | Status | owner_role | collected_at
-----+-----+-----+-----+-----+-----+-----+-----+
→-----+-----+-----+-----+-----+-----+-----+
→-----+-----+-----+-----+-----+-----+-----+
feedback_na_pipe | Live           |                          | 0 |
      |              412 |                          412 | 0 |
      0 | UpToDate | app_owner | 2026-07-21 10:15:03.221+00
product_docs_pipe | Background     |                          |
6 |              1842 |                          1842 | 0 |
      0 | UpToDate | app_owner | 2026-07-21 10:15:03.402+00
(2 rows)
```

列は `aidb.pipeline_metrics` に一致するものに加えて、

- `owner_role` パイプラインの所有ロール、`aidb.pipelines.owner_role` と同じ。
- `collected_at` この行が最後に更新された日。

削除されたパイプラインの行は、次の収集サイクルで自動的に削除されます。

コレクターの有効化

コレクターはオプションであり、デフォルトで無効になっているため、`aidb.pipeline_metrics_collection` は、オンにするまで空のままです。インスタンス全体で有効にします。

```
ALTER SYSTEM SET aidb.enable_background_metrics_collector = on;
```

または、1 つのデータベースにスコープを設定し、そのデータベースのインスタンス全体のデフォルトをオーバーライドします。

```
ALTER DATABASE my_db SET aidb.enable_background_metrics_collector = on;
```

!!!note Postgres を再起動して、新しい構成を適用します。

失敗したレコードの検査

レコードごとの障害の詳細失敗したステップ、オペレーション、基になるエラーメッセージ、および再試行状態は *Pipeline error log* にあります。

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。7.2.0 より前に作成されたパイプラインは、アップグレード中にテーブルを取得します。
- 書き込みは失敗したステップのトランザクション内で発生するため、成功したステップが孤立したログ行を残すことはなく、障害がレコードを失うことはありません。
- ログに記録された各エラーは、Postgres **WARNING** も生成します。 `promoteb.pipeline_error_warnings` で切り替えます; エラーは関係なくログに残ります。
- `query` および `summary` ファンクションは、エラーログテーブルが欠落しているパイプラインに対して空の結果を返します。
- エラーログテーブルを手動で削除しないでください。7.4.0 の時点で、ソース **DELETE** ハンドラーとステップ後クリーンアップは、テーブルが見つからない場合にエラーを発生させます。
- エラーは、4 つの `error_category` 値のいずれかを取得します `helpb.ErrorBlocking` を参照してください。

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。

次のページに続く

表 14 – 前のページからの続き

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

レコードレベルの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続けます。パイプラインレベルのエラーは、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。具体的には **RecordTemporary** は現在モデルレート制限エラーの場合にのみ起動し、**RecordPermanent** は不正な入力データたとえば、破損した PDF、不正な HTML、無効な UTF-8 をカバーし、**PipelineTemporary** は一時的な外部障害たとえば、オブジェクトストアの停止、モデルサービスタイムアウトをカバーします)、**PipelinePermanent** は、一般的な構成の誤りたとえば、削除されたモデル、無効なステップ構成などです。

エラーのクエリー

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

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

__OUTPUT__

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

(3 rows)

ソースレコード、ステップ番号、またはエラーカテゴリでフィルタリングします。

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

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

完全なパラメーターと戻り型の詳細については、[Operations reference](#) および [pro-](#)

`moteb.get_all_error_summaries` を参照してください。

失敗したレコードの再キューイング

7. 5.0 の新機能。 `aidb.requeue_pipeline_errors()` は、ターゲットのレコードレベルのエラー行を削除し、パイプラインのステートサーフェスでソースレコードをダーティとしてマークします。次の通常の実行で再処理されます。呼び出し自分自身は処理を行いません。

```
SELECT * FROM aidb.requeue_pipeline_errors(my_pipeline, ARRAY[1, 2, 99]);
```

__OUTPUT__

error_id	action
1	requeued
2	requeued
99	not_found

- - - - -

(3 rows)

結果は `error_id` の昇順で並べられます。各行は、3つのアクションのいずれかをレポートします。

- **requeued** レコードレベルの行が削除され、ソースがダーティとしてマークされました。テーブルソースは状態イベントをエンキューします。ポリウムソースは、状態行にワンショットの `retry_requested_at` マーカーを設定します。
- **skipped_pipeline_level** エントリーには NULL `source_id` 以下を参照してください、そして適切な場所に残されます。
- **not_found** ID がエラーログになかったか、パイプラインにエラーログテーブルがありません古いパイプラインまたは Semantic KB のような Empty ソース。

バックグラウンドパイプラインは、次のワーカティックに再キューイングされた行を自動的にドレインします。無効またはライブパイプラインを実際に再処理するには、フォローアップの `aidb.run_pipeline('my_pipeline')` が必要です。

リキューイングは遅延マークダーティ操作であり、同期再試行ではありません。呼び出し自分自身は処理を行わないため、レコードごとの成功または失敗は、フォローアップの実行後にのみ表面化します。再処理後に何が起ったのかを確認するには、`aidb.get_error_logs` を再照会します。

重複する ID のコンカレント・リキューは、行ごとの FOR UPDATE ロックでシリアル化されます。重複しない ID セットは並列的に処理されます。Requeue は、`aidb.run_pipeline` またはバックグラウンドワーカーをブロックしません。

パイプラインレベルのエラー

パイプラインレベルのエラー NULL `source_id` は、特定のレコードに関連付けられていないバッチ障害を記録します。レコードごとに再度キューイングすることはできないため、`aidb.requeue_pipeline_errors` は `action='skipped_pipeline_level'` を返し、ログにエントリを残します。

```
SELECT * FROM aidb.requeue_pipeline_errors(my_pipeline, ARRAY[7]);
```

__OUTPUT__

(次のページに続く)

```

error_id |          action
-----+-----
      7 | skipped_pipeline_level
(1 row)

```

根本的な問題たとえば、モデルの可用性、構成、削除されたモデルなどを修正し、パイプラインを再実行します。次のステップがエラーなしで実行されると、NULL 行は自動的にクリアされます。再実行せずに終了するには、`aidb.clear_error_logs` を使用します。

無効モードのボリュームに関する注意事項

ボリュームソースを使用する無効モードのパイプラインでは、`aidb.requeue_pipeline_errors` はマークされたファイルに `retry_requested_at` マーカーを設定しますが、ドレインをスケジュールできないため、`aidb.run_pipeline` を指す NOTICE を発行します。手動 `aidb.run_pipeline` は、マークされたファイルのみではなくボリューム全体を再読み取ります。これにより、大規模なボリュームのフルボリュームスキャンがコストがかかります。

大規模な無効モードボリュームの場合、パイプラインをバックグラウンドモードに切り替え、`aidb.requeue_pipeline_errors` を呼び出し、ワーカーにマークされたファイルをドレインさせて、パイプラインを無効に戻します。

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。決済は ID ベースです。 `get_error_logs()` を使用してエントリを削除する前に確認してください。

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。

自動クリーンアップ

エラーログエントリは、3つの場合、`aidb.clear_error_logs` または `aidb.requeue_pipeline_errors` なしでクリアされます。

- ソースレコードが削除されました。ライブ DELETE トリガー、バックグラウンドワーカー DELETE 処理、およびボリュームの古い状態のクリーンアップはすべて、宛先行とともに一致するエラーログエントリを削除します。

- ソースエンベロープは、後の実行で宛先に到達します。パイプラインエグゼキューターは、(source_id, part_ids) によってスコープされるエントリをクリアするため、部分的粒度の成功によって兄弟エラーがワイプされません。
- そのステップが次に正常に実行された後、ステップのパイプラインレベルの行 NULL source_id。その実行中に同時に失敗したエグゼキューターによって書き込まれたエラーは保存されます。

パイプラインを削除すると、エラーログテーブル全体が削除されます。

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。

エンドツーエンドの例

3つのドキュメントソースを使用した短いウォークスルー。1つの行には破損したPDFがあり、1つは解析しますが、特大のチャンクを生成し、3番目は埋め込みステップでモデルレート制限に達します。

パイプラインを実行します。3つの障害はいずれもバッチを中止しません。次に、結果のエラーを検査します。

```
SELECT aidb.run_pipeline(my_pipeline);

SELECT id, source_id, error_category
FROM aidb.get_error_logs(my_pipeline);
__OUTPUT__
id | source_id | error_category
-- +-----+-----
3 | doc_200   | RecordTemporary
2 | doc_42    | RecordPermanent
1 | doc_7     | RecordPermanent
(3 rows)
```

パイプラインは **PartialErrors** を報告し、数を表示します。

```
SELECT pipeline, "Status", "count(record errors)", "count(blocking errors)"
FROM aidb.pipeline_metrics WHERE pipeline = my_pipeline;
__OUTPUT__
pipeline | Status | count(record errors) | count(blocking errors)
-- +-----+-----+-----
my_pipeline | PartialErrors | 3 | 0
(1 row)
```

上流のレート制限の問題を修正した後、レート制限された行を再度キューに入れます。これは、ソースがダーティであることを示します。自分自身は再処理しません。

```
SELECT * FROM aidb.requeue_pipeline_errors(my_pipeline, ARRAY[3]);
__OUTPUT__
error_id | action
-----+-----
      3 | requeued
(1 row)
```

再度キューに入れられた行をドレインします。バックグラウンドパイプラインがこれを自動的に取得します。無効またはライブパイプラインには明示的な呼び出しが必要です。トリアージ後に 2 つの永続的な障害を無視します。

```
SELECT aidb.run_pipeline(my_pipeline);

SELECT aidb.clear_error_logs(my_pipeline, ARRAY[1, 2]);
__OUTPUT__
clear_error_logs
-----
                2
(1 row)
```

ステータスは UpToDate に戻ります。

```
SELECT pipeline, "Status" FROM aidb.pipeline_metrics WHERE pipeline = my_pipeline;
__OUTPUT__
pipeline | Status
-----+-----
my_pipeline | UpToDate
(1 row)
```

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 全体で使用する型と列挙型、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。

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,
```

(次のページに続く)

(前のページからの続き)

```
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。
<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>Unknown</i>	ステータスは特定できません。
<i>PartialErrors</i>	一部のレコードはレコードレベルのエラーで失敗しました。それ以外の場合、パイプラインは動作しています。
<i>BlockingErrors</i>	パイプラインレベルのエラーにより処理が停止されました。問題を解決し、再実行して、エントリをクリアします。

helpb.ErrorBlocking

ログに記録されたエラーがパイプライン処理に与える影響を分類します。2次元スコープレコード対パイプラインと時間性一時的対永続的。

```
CREATE TYPE ErrorBlocking AS ENUM (
    RecordTemporary,
    RecordPermanent,
    PipelineTemporary,
    PipelinePermanent
);
```

Value	Description
<i>RecordTemporary</i>	一時的なレコードレベルの障害。他のレコードの処理は継続されます。現在、モデルレート制限エラーの場合にのみ発生します。
<i>RecordPermanent</i>	永続的なレコードレベルの障害たとえば、不正な入力データ。他のレコードの処理は継続されます。
<i>PipelineTemporary</i>	一時的なパイプラインレベルの障害たとえば、ネットワークタイムアウト、サービスの停止など。ステップをブロックします。再試行で解決される場合があります。
<i>PipelinePermanent</i>	永続的なパイプラインレベルの障害たとえば、削除されたモデル、無効な構成。ステップをブロックします。

helpb.DistanceOperator

ベクトル類似性検索に使用される距離メトリックを指定します。

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

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

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。

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>pro-</i> <i>moteb.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>pro-</i> <i>moteb.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。
<i>auto_processing</i>	text	現在の自動処理モード。

次のページに続く

表 24 – 前のページからの続き

Column	Type	Description
<i>table: unprocessed rows</i>	bigint	テーブルソースの場合 まだ処理されていない行数。
<i>volume: scans completed</i>	bigint	ボリュームソースの場合 完了したフルスキャンの数。
<i>count(source records)</i>	bigint	ソース内のレコードの合計数。
<i>count(destination records)</i>	bigint	宛先のレコードの合計数。
<i>Status</i>	text	現在の パイプラインステータスの統合 。
<i>count(record errors)</i>	bigint	このパイプラインでログに記録されたレコードレベルのエラーの数 <i>RecordTemporary</i> plus <i>RecordPermanent</i> 。
<i>count(blocking errors)</i>	bigint	このパイプラインでログに記録されたパイプラインレベルのエラーの数 <i>PipelineTemporary</i> plus <i>PipelinePermanent</i> 。
<i>last run completed</i>	timestamptz	パイプラインの最後の実行が終了したとき。7.6.0で導入されました。

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) | last run completed
-----+-----+-----+-----+-----
→-----+-----+-----+-----+-----
→-----+-----+-----+-----+-----
pipeline__7471a      | Background      |                          | 0 |
      |                | 5 |                          | 5 | UpToDate |
      0 |                | 0 | 2026-07-21 10:15:03.221+00
pipeline__7471b      | Background      |                          | 0 |
      |                | 5 |                          | 5 | UpToDate |
      0 |                | 0 | 2026-07-21 10:15:03.402+00
animal_facts_kb      | Disabled        |                          | 0 |
      |                | 99 |                          | 372 | UpToDate |
      0 |                | 0 | 2026-07-20 09:02:11.010+00
animal_facts_kb_bert | Disabled        |                          | 0 |
      |                | 99 |                          | 372 | UpToDate |
      0 |                | 0 | 2026-07-20 09:02:11.010+00
mpkb_pipe_int        | Disabled        |                          | 0 |
      |                | 2 |                          | 4 | UpToDate |
      0 |                | 0 | 2026-07-19 16:40:55.884+00
```

(次のページに続く)

(前のページからの続き)

mpkb_pipe_text	Disabled	0	
	2	4	UpToDate
0	0	2026-07-19 16:40:55.884+00	
(6 rows)			

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
name	TEXT	Required	パイプラインの名前。最大 46 文字。
source	TEXT	Required	ソーステーブルまたはボリュームの名前。
step_1	helpb.PipelineStepOp	Required	最初のパイプラインステップのオペレーション。
source_key_column	TEXT	NULL	ソーステーブルの一意のキー列。
source_data_column	TEXT	NULL	処理するデータを含む列。
destination	TEXT	NULL	宛先テーブルまたはボリュームの名前。
auto_processing	helpb.PipelineAutoPr モード	NULL	自動処理モード。
batch_size	INT	NULL	バッチごとに処理するレコードの数。
back-ground_sync_interval	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。
owner_role	TEXT	NULL	このパイプラインを所有および実行するロール。

次のページに続く

表 25 – 前のページからの続き

Parameter	Type	Default	Description
<i>step_1_options</i>	JSONB	NULL	ステップ 1 の構成適切な パイプラインステップ構成ヘルパー を使用します。どのステップでもここで <code>`intermediate_destination`</code> を設定して、出力を永続化することもできます。 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 を参照してください。
<i>step_2</i> ... <i>step_10</i>	<i>helpb.PipelineStepOp</i>	NULL	ステップ 2～10 のオペレーション。
<i>step_2_options</i> ... <i>step_10_options</i>	JSONB	NULL	手順 2～10 の構成。

返品

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

例

```

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

```

(次のページに続く)

(前のページからの続き)

```

name          => pdf_to_kb,
source        => pdf_volume,
destination    => my_kb,
step_1        => ParsePdf,
step_1_options => aidb.pdf_parse_config(),
step_2        => KnowledgeBase,
step_2_options => aidb.knowledge_base_config(my_model, Text),
auto_processing => Background,
background_sync_interval => 60 seconds
);

```

promoteb.update_pipeline

既存のパイプラインの自動処理設定を更新します。

パラメーター

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

例

```

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

```

promoteb.delete_pipeline

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

パラメーター

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

例

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

promoteb.run_pipeline

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

パラメーター

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

例

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

promoteb.get_pipeline_metrics

パイプラインの現在の処理統計を返します。

パラメーター

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

返品

Column	Type	Description
<i>name</i>	text	パイプラインの名前。
<i>auto_processing</i>	text	現在の自動処理モード。
<i>count_events</i>	bigint	テーブルソースの場合 未処理の変更イベントの数。
<i>count_source</i>	bigint	ソース内のレコードの合計テーブルソースの場合。
<i>count_known_objects</i>	bigint	既知のオブジェクトの合計ボリュームソースの場合。
<i>last_full_run_id</i>	bigint	最後に完了したフルスキャンの ID ボリュームソースの場合。

[次のページに続く](#)

表 31 – 前のページからの続き

Column	Type	Description
<i>count_destination</i>	bigint	宛先の合計レコード。
<i>status</i>	text	現在の パイプラインステータスの統合 。
<i>count_errors_record</i>	bigint	このパイプラインでログに記録されたレコードレベルのエラーの数 <i>RecordTemporary</i> plus <i>RecordPermanent</i> 。
<i>count_errors_pipeline</i>	bigint	このパイプラインでログに記録されたパイプラインレベルのエラーの数 <i>PipelineTemporary</i> plus <i>PipelinePermanent</i> 。
<i>last_run_completed</i>	timestampz	パイプラインの最後の実行が終了したとき。7.6.0 で導入されました。

例

```
SELECT * FROM aidb.get_pipeline_metrics(my_chunker);
```

エラーログファンクション

障害が発生したパイプラインレコード `aidb.get_error_logs`、`aidb.clear_error_logs`、`aidb.get_error_log_summary`、`aidb.get_all_error_summaries`、`aidb.requeue_pipeline_errors` を検査、クリア、および再キューイングするためのファンクションは、[Operations reference](#) に文書化されています。エンドツーエンドのガイドスタイルのワークフローについては、[Pipeline error log](#) を参照してください。

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。それらには、`intermediate_destination` のパラメーターがありません。ステップのオプションに追加する方法については、[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](#) を参照してください。

`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。

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>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> '。

例

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

helpb.summarize_text_config

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

パラメーター

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

例

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

promoteb.ocr_config

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

パラメーター

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

例

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

promoteb.html_parse_config

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

パラメーター

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

例

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

promoteb.pdf_parse_config

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

パラメーター

Parameter	Type	Default	Description
<i>method</i>	TEXT	NULL	使用する解析方法。`NULL` の場合、デフォルトの方法を使用します。
<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、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>	helpb.PipelineDataFormat	Required	埋め込まれるデータのフォーマット。
<i>distance_operator</i>	helpb.DistanceOperator	NULL	類似検索のための距離機能。デフォルト `L2`
<i>vector_index</i>	JSONB	NULL	ベクトルインデックス構成 Vector 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>	helpb.PipelineDataFormat	Required	埋め込まれるデータのフォーマット。

例

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

helpb.inference_config

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

パラメーター

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

例

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

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。
<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。

!!!note HNSW は、最大 2000 の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();
```

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。

構成

AI パイプラインの動作を制御するセッション設定可能な GUC。

promoteb.pipeline_error_warnings

Property	Value
Type	boolean
Default	<i>true</i>
Context	<i>SUSET</i>

スーパーユーザーは、セッションごと、ロールごと、データベースごと、または `postgresql.conf` でこの設定を変更できます。他のロールは、`SET` を行う前に、明示的な `GRANT SET ON PARAMETER aidb.pipeline_error_warnings` が必要です。

有効にすると、パイプラインのエラーログテーブルに記録された各エラーも Postgres WARNING を生成します。エラーは、この設定に関係なくログに残ります。永続性に影響を与えることなく警告をサイレントにして無効にします。

```
SET aidb.pipeline_error_warnings = false;
```

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。

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

複数の操作を連鎖させることができます。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,
```

(次のページに続く)

(前のページからの続き)

```
    step_2 => ChunkText
);
```

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
    )
);
```

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,
```

(次のページに続く)

(前のページからの続き)

```

    auto_processing => Live
);

```

モニタリングと削除

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

```

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

```

```

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

```

エンドツーエンドの例 RAG のナレッジベース

この例では、内部サポートナレッジベースを介して、完全に SQL で完全にローカルな、完全な検索拡張世代 RAG フローを構築します。そして、世代モデルは、取得された記事に基づいて回答を合成します。この例の縮小バージョンは、AI pipelines landing page にあります。

ステップ 1 モデルを登録する

各ステージでは、異なる種類のモデルを使用します。埋め込みモデル `bge-small-en-v1.5-f16` は、AIDB インストールのたびに事前登録されるため、リランキングモデルと生成モデルのみが登録が必要です。`aidb.create_model()` は、Hugging Face から各 GGUF ファイルをダウンロードして検証しますリランカーは～420 MB、Llama-3.2-3B～3.4 GB です。revision ピンにより、例が再現可能になります。それらがないと、モデルリポジトリの変更可能な main ブランチが使用されます。

```
-- Reranking model (cross-encoder)
SELECT aidb.create_model(
  name      => bge-reranker-v2-m3-Q4,
  provider => llamacpp_reranking,
  config    => jsonb_build_object(
    model,      gpustack/bge-reranker-v2-m3-GGUF,
    model_file, bge-reranker-v2-m3-Q4_K_M.gguf,
    revision,   3093af03b1a635e67b084b1d8c03c5f5e020fd05,
    n_ctx,      2048
  )
);

-- Generation model (instruct LLM)
SELECT aidb.create_model(
  name      => llama-3.2-3b-instruct-Q8_0,
  provider => llamacpp_generate,
  config    => jsonb_build_object(
    model,      unsloth/Llama-3.2-3B-Instruct-GGUF,
    model_file, Llama-3.2-3B-Instruct-Q8_0.gguf,
    revision,   e7d0997e49c9cb00d88b4c1a6a16aa894b0bbc31,
    n_ctx,      8192,
    temperature, 0.0
  )
);
```

!!!note 生成ステージのモデルサイジング 1B クラス生成モデル事前登録された llama-3.2-1b-instruct-Q8_0 を含むは、この例の生成ステージでは信頼できません。テストでは、質問のコンストレインと矛盾するアクションを推奨しました。Llama-3.2- 3B -指示は正しく回答し、まだ控えめなローカルモデルです。

ステップ 2 ソースデータを作成する

データベース操作に関する 16 の短いサポート記事。いくつかはボキャブラリーディスク領域、バキューム、バックアップを意図的に共有します。これが、後の再ランキング段階が目に見えて重要になるものです。

```
CREATE TABLE support_articles (
  id      INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
  title   TEXT NOT NULL,
  body    TEXT NOT NULL
```

(次のページに続く)

(前のページからの続き)

);

INSERT INTO support_articles (title, body) **VALUES**(Reclaiming disk **space with VACUUM FULL**,

VACUUM FULL rewrites a **table into a new file and returns** unused disk **space to the**
 ↳operating **system**. It takes an **ACCESS EXCLUSIVE lock**, so **all reads and writes to the**
 ↳**table are blocked until** it completes.),

(Reducing **table bloat without** downtime,

The pg_repack extension rebuilds bloated tables **and indexes online**. It holds **only**
 ↳brief locks, so applications can **continue reading and writing while space is**
 ↳reclaimed. It requires **free disk space** roughly equal **to the table size**.),

(Monitoring **database disk usage**,

Use pg_database_size() **and pg_total_relation_size()** to track how much disk **space**
 ↳databases **and tables occupy**. Configure alerts well **before storage** volumes reach
 ↳capacity.),

(Resolving "too many connections" errors,

The error "FATAL: sorry, too many clients already" means the max_connections **limit**
 ↳**is** reached. Find idle sessions **in pg_stat_activity and terminate** them **with pg_**
 ↳**terminate_backend()**, **or** introduce a **connection pooler**.),

(Connection pooling **with PgBouncer**,

PgBouncer multiplexes thousands **of** client connections onto a small pool **of database**
 ↳sessions. **Transaction pooling mode** provides the highest **connection density for**
 ↳typical web application workloads.),

(Diagnosing streaming replication lag,

Compare pg_current_wal_lsn() **on the primary with pg_last_wal_replay_lsn()** **on the**
 ↳standby **to** measure replication lag. Common causes include network saturation **and**
 ↳long-running queries **on the standby holding back WAL replay**.),

(Creating a hot standby **with pg_basebackup**,

Run pg_basebackup **to** clone the **primary server, then start** the clone **with** a standby.
 ↳signal file. The standby serves **read-only** queries while continuously replaying WAL
 ↳**from the primary**.),

(Replicating selected tables **with** logical replication,

Logical replication publishes **row changes for** selected tables **to subscriber**
 ↳databases **and** keeps them continuously synchronized. Unlike streaming replication,
 ↳it can replicate a subset **of tables and works across major versions**.),

(Point-in-time recovery **with** WAL archiving,

With continuous WAL archiving enabled, you can restore a base backup **and set**
 ↳recovery_target_time **to** roll the **database forward to a specific** moment, such **as**
 ↳just **before** an accidental **data deletion**.),

(Logical backups **with** pg_dump,

pg_dump exports a **database or** single tables **to** a portable archive that can be
 ↳restored **into** newer PostgreSQL **versions**. Dumps **are** consistent snapshots but offer
 ↳**no incremental or point-in-time** capability.),

(Reading query plans **with EXPLAIN ANALYZE**,

EXPLAIN ANALYZE executes a query **and reports the actual time** spent **in each** plan node.

(次のページに続く)

(前のページからの続き)

```

→ Look for row-estimate mismatches and sequential scans on large tables to find
→ missing indexes.),
(Finding and removing unused indexes,
  Query pg_stat_user_indexes for indexes with zero scans. Unused indexes slow down
→ every write and consume disk space; dropping them is a safe, immediate win.),
(Tuning work_mem for large sorts,
  When sorts or hash joins exceed work_mem, they spill to temporary files on disk and
→ slow down dramatically. Raise work_mem session-locally for reporting queries rather
→ than globally.),
(Restricting data access with row-level security,
  Row-level security policies filter which rows each role can see or modify. Policies
→ are enforced by the server itself, so they apply uniformly to every client and
→ application.),
(Auditing database activity with pgaudit,
  The pgaudit extension produces detailed audit logs of reads, writes, and DDL for
→ compliance requirements. Configure log classes narrowly to control log volume.),
(Upgrading major versions with pg_upgrade,
  pg_upgrade migrates a cluster to a new major version by relinking data files instead
→ of dumping and reloading. With the --link option, downtime is typically minutes
→ even for large clusters.);

```

ステップ 3 ナレッジベースパイプラインを作成して実行する

```

SELECT aidb.create_pipeline(
  name          => support_kb,
  source        => support_articles,
  source_key_column => id,
  source_data_column => body,
  step_1        => KnowledgeBase,
  step_1_options => aidb.knowledge_base_config(
    model      => bge-small-en-v1.5-f16,
    data_format => Text
  ),
  auto_processing => Disabled
);

SELECT aidb.run_pipeline(support_kb);

```

ステップ 4 取得

セマンティック検索は、クエリが共有するキーワードがほとんどまたはまったくない場合でも、適切な記事を見つけます。生のエラーテキストを貼り付け、解像度を取得します。

```

SELECT a.title, r.distance::numeric(5,3) AS distance
FROM aidb.retrieve_key(support_kb,

```

(次のページに続く)

(前のページからの続き)

```

application logs show: FATAL: sorry, too many clients already, 3) r
JOIN support_articles a ON a.id = r.key::int
ORDER BY r.distance;
__OUTPUT__

```

title	distance
Resolving "too many connections" errors	0.507
Tuning work_mem for large sorts	0.844
Connection pooling with PgBouncer	0.908

(3 rows)

または、回答と共通のツール名またはキーワードを持たない症状を説明します。

```

SELECT a.title, r.distance::numeric(5,3) AS distance
FROM aidb.retrieve_key(support_kb,
    A developer accidentally deleted rows from a table yesterday. How do we get
    ↳the data back?, 3) r
JOIN support_articles a ON a.id = r.key::int
ORDER BY r.distance;
__OUTPUT__

```

title	distance
Point-in-time recovery with WAL archiving	0.795
Reducing table bloat without downtime	0.806
Finding and removing unused indexes	0.846

(3 rows)

`aidb.retrieve_text()` は、結合を必要とせずに、記事本文を直接返します。出力テーブルでタイトルが読みやすいため、結合バリエーションが表示されています。

ステップ 5: 再ランキング

デモの質問「データベースのディスク領域が不足していますが、アプリケーションをオフラインにして修正することはできません。」

ベクトル検索だけでは、ポイントインタイムリカバリーディスク領域の問題には明らかに役に立たない— が 1 位にランクされ、正しい記事 *pg_repack* がほぼ同点で 2 位です。

```

SELECT a.title, r.distance::numeric(5,3) AS distance
FROM aidb.retrieve_key(support_kb,
    The database is running out of disk space, but we cannot take the application
    ↳offline to fix it., 5) r
JOIN support_articles a ON a.id = r.key::int
ORDER BY r.distance;
__OUTPUT__

```

title	distance
-------	----------

(次のページに続く)

(前のページからの続き)

```

-----+-----
Point-in-time recovery with WAL archiving | 0.808  ← wrong
Reducing table bloat without downtime      | 0.838  ← correct answer, near-tie
Tuning work_mem for large sorts            | 0.851
Monitoring database disk usage             | 0.865
Resolving "too many connections" errors    | 0.883
(5 rows)

```

クロスエンコーダーは、質問と各記事を一緒に読み取り、思い切って再注文します。aidb.rerank_text() は (text, logit_score, id) を返します。ここで、id は、入力配列内の 0 ベースの位置であり、より高い logit_score ほど関連性が高いことを意味します。id + 1 は、そのインデックスを 1 ベースの SQL 配列にマップします。

```

WITH candidates AS (
    SELECT array_agg(a.title ORDER BY r.distance) AS titles,
           array_agg(a.body ORDER BY r.distance) AS bodies
    FROM aidb.retrieve_key(support_kb,
        The database is running out of disk space, but we cannot take the
        ↪ application offline to fix it., 5) r
    JOIN support_articles a ON a.id = r.key::int
)
SELECT c.titles[r.id + 1] AS title, r.logit_score::numeric(7,3) AS logit_score
FROM candidates c,
     aidb.rerank_text(bge-reranker-v2-m3-Q4,
        The database is running out of disk space, but we cannot take the
        ↪ application offline to fix it.,
        c.bodies) r
ORDER BY r.logit_score DESC;
__OUTPUT__

          title                                     | logit_score
-----+-----
Reducing table bloat without downtime          | -2.742  ← promoted to #1, clear
↪ margin
Monitoring database disk usage                 | -3.116
Point-in-time recovery with WAL archiving      | -7.037  ← demoted
Tuning work_mem for large sorts                | -8.836
Resolving "too many connections" errors        | -9.216
(5 rows)

```

これは古典的な 2 段階の取得パターンです。ベクトルインデックスここでは 5、運用環境では 20~100 を使用して候補を安くオーバーフェッチし、より精度が高いがより高価なクロスエンコーダーに最終的な注文を選択させます。リランカーは、候補ごとに完全なモデルのフォワードパスを実行します。テーブル全体をリランクしないでください。

ステップ 6 生成

1つのステートメントで完全な RAG。5つの候補を取得し、再ランク付けしてベスト3を維持し、根拠のある回答を生成します。最初の2つの CTE は、上記のステップ4およびステップ5コマンドとまったく同じです。\\set は psql 機能です。他のクライアントでは、質問テキストを直接インラインします。

```

\\set question The database is running out of disk space, but we cannot take the
→application offline to fix it. What should we do?

WITH candidates AS (
    -- Stage 1: vector search over the knowledge base (over-fetch 5 candidates)
    SELECT array_agg(a.title ORDER BY r.distance) AS titles,
           array_agg(a.body ORDER BY r.distance) AS bodies
    FROM aidb.retrieve_key(support_kb, :question, 5) r
    JOIN support_articles a ON a.id = r.key::int
),
reranked AS (
    -- Stage 2: cross-encoder reranking, keep the best 3
    SELECT c.titles[r.id + 1] AS title,
           c.bodies[r.id + 1] AS body,
           r.logit_score
    FROM candidates c,
         aidb.rerank_text(bge-reranker-v2-m3-Q4, :question, c.bodies) r
    ORDER BY r.logit_score DESC
    LIMIT 3
),
context AS (
    SELECT string_agg(### || title || E\\n || body, E\\n\\n
                     ORDER BY logit_score DESC) AS articles
    FROM reranked
)
-- Stage 3: grounded answer generation
SELECT aidb.generate_text(
    llama-3.2-3b-instruct-Q8_0,
    E\\nKnowledge base articles:\\n\\n || context.articles ||
    E\\n\\nQuestion: || :question ||
    E\\n\\nAnswer the question using only the facts in the articles above. Recommend
→one course of action and mention its caveats.,
    {
        "system_prompt": "You are a database support assistant. Be accurate and
→concise; do not invent facts that are not in the provided articles.",
        "max_tokens": 250,
        "temperature": 0.0
    }::json
) AS answer
FROM context;

```

出力 temperature: 0.0 の決定性—繰り返し実行により同じ回答が生成されます。

Based on the provided articles, I recommend using the `pg\_repack` extension to reduce table bloat without downtime.

According to the article, `pg\_repack` rebuilds bloated tables and indexes online, holding only brief locks, allowing applications to continue reading and writing while space is reclaimed.

This approach does not require taking the application offline.

However, there is a caveat: `pg\_repack` requires free disk space roughly equal to the table size. This means that you need to ensure there is sufficient free disk space available before running `pg\_repack`, which may require some planning and preparation.

この回答を含む単一の記事はありません。モデルは、質問からの制約「オフラインにできない」を2つの競合する記事からの事実 VACUUM FULL ブロックの読み取りと書き込み。pg_repack はオンラインで動作し、空きディスク領域の警告を引き継ぎました。—逐語的な結果ではなく、合成。

同じステートメントは、ホストされたものを含む登録された世代モデルで動作します。モデル名のみが変更されます。ホストされた GPT-5.x 推論モデルの注意事項が1つあります。推論構成から"temperature"を省略します。彼らはパラメーターを拒否します。

ステップ 7: ティアダウン

```
SELECT aidb.delete_pipeline(support_kb);
DROP TABLE support_articles;
SELECT aidb.delete_model(llama-3.2-3b-instruct-Q8_0);
SELECT aidb.delete_model(bge-reranker-v2-m3-Q4);
```

エンベディングモデルは削除されません。bge-small-en-v1.5-f16 は、事前登録されたデフォルトの1つです。

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。デフォルトではオフであり、ステップ構成ヘルパー aidb.chunk_text_config() などにはそのパラメーターがあります。以下の両方の例は、ヘルパー独自のオプションとともにプレーンな JSON として追加します。クエリー時に deduplicate と相互作用する方法を含む完全なリファレンスについては、[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](#) を参照してください。

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;
```

(次のページに続く)

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

例 2 自動中間宛先

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

- 自動名前付けの中間ストレージを使用してパイプラインを作成します。

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

- 自動生成された中間テーブルにアクセスします。

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

エンドツーエンドの例 スキャンされた PDF を検索可能なナレッジベースに

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

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

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

前提条件

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

```
-- Register the OCR model
SELECT aidb.create_model(
  my_paddle_ocr_model,
  nim_paddle_ocr,
  credentials => {"api_key": "<NVIDIA_NIM_API_KEY>"}::JSONB -- or credentials_env =>
  -- to read this from an environment variable instead
);
```

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

ステップ 1 ソーステーブルを作成する

生の PDF データを保持するテーブルを作成します。

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

ステップ 2 PDF をテーブルにロードする

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

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

S3 互換ボリュームから PDF をロードすることもできます。[Configuring external storage](#) および [パイプラインの作成 ボリュームソース](#) を参照してください。

ステップ 3 パイプラインを作成する

```
SELECT aidb.create_pipeline(
  name          => scanned_pdf_kb,
  source        => scanned_pdfs,
  source_key_column => id,
  source_data_column => pdf_data,
  step_1        => PdfToImage,
  step_1_options => {"dpi": 150}::jsonb,
  step_2        => PerformOcr,
```

(次のページに続く)

(前のページからの続き)

```
step_2_options    => aidb.ocr_config(model => my_paddle_ocr_model),
step_3            => KnowledgeBase,
step_3_options    => aidb.knowledge_base_config(
  model           => bert,
  data_format      => Text,
  distance_operator => Cosine
),
auto_processing   => Live
);
```

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

ステップ 4 既存のデータでパイプラインを実行する

パイプラインを有効にする前行を挿入した場合、ここで処理します。

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

ステップ 5 ナレッジベースを照会する

パイプラインが完了したら、セマンティック検索を使用して結果を照会します。

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

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

第11章 Knowledge bases

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

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

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

11.1 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。

11.1.1 検索機能の流れ

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。

11.1.2 `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。
- **intermediate_steps** *intermediate_destination* で構成されたステップ、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 列たとえば、`ChunkText`。パイプライン内のステップで中間ストレージが有効になっていない場合は空。これが `deduplicate` とどのように相互作用するかについては、以下の [重複排除と中間結果](#) を参照してください。

11.1.3 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。

11.2 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。これは、`intermediate_steps` 列の手動代替であり、[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](#) が有効になっているステップで `aidb.retrieve_text()` が既に自動的に返しています。中間テーブルの追加列への結合など、`intermediate_steps` が提供しないものが必要な場合にのみ手動結合に手を伸ばします。

11.2.1 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,
```

(次のページに続く)

(前のページからの続き)

```

    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];

```

`intermediate_steps` と同様に、この結合は 1 つの行のみを参照します `deduplicate` 通過します。次のセクションを参照してください。

11.3 重複排除と中間結果

`deduplicate` `aidb.retrieve_key()` と `aidb.retrieve_text()` の両方のデフォルト `true` は、`ChunkText` のようなステップがレコードをいくつかの独立した埋め込み部分に分割し、それらの部分のうちの複数の部分がクエリーと一致する場合でも、取得結果をソースレコードごとに 1 行に折りたたみます。1 つの部分のみが「勝ち」クエリーに最も近いものであり、その `part_ids` および `intermediate_steps` は、そのソースレコードに対して返される唯一の部分です。

これは、チャンクまたは解析ステップで [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](#) を有効にするとすぐに重要になります。デフォルトの `deduplicate => true` では、複数のチャンクが一致した場合でも、ソースレコードごとに一致するチャンクの中間コンテンツは **one** のみが表示されます。一致するすべての部分とそれぞれ独自の `intermediate_steps` エントリーを表示するには、`deduplicate => false` で取得を呼び出します。

たとえば、パイプラインは、`intermediate_destination` を有効にして動物に関する一連の短い記事をチャンク化するため、各チャンクのテキストは `intermediate_steps` で利用できます。デフォルトの重複排除で `'ocean'` を照会すると、`whale` 記事の最も一致するチャンクが勝ち、2 番目のチャンクも一致した場合でも、他の `whale` チャンクは表示されません。

第12章 Knowledge bases

経験則としてソースレコードごとに1つの代表的な結果が必要な場合は `deduplicate => true` デフォルトのままで、同じソースレコードから一致するすべてのチャンクまたは部分が必要な場合は常に `deduplicate => false` を使用します。たとえば、長い文内のすべてのパッセージを表示する場合単一の最適なドキュメントだけでなく、クエリーにマッチしたドキュメント。

12.1 Multi-pipeline knowledge bases

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

これは、次のような場合に役立ちます。

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

12.1.1 既存の KB へのパイプラインのアタッチ

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

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

(次のページに続く)

(前のページからの続き)

```

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

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

```

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

```

SELECT key, value, pipeline_name
FROM aidb.retrieve_text(public.customer_feedback_kb, shipping was slow, topk => 3);
__OUTPUT__

```

key	value	pipeline_name
1042	Delivery was held up in customs for over a week	feedback_eu_pipe
318	Order took twice as long to arrive as estimated	feedback_na_pipe
1057	Tracking number never updated until it arrived	feedback_eu_pipe

さまざまなステップチェーンが1つのKBを共有できます。たとえば、あるパイプラインはKnowledgeBaseステップの前に `ChunkText` で長いドキュメントをチャンクし、別のパイプラインは短い行を直接埋め込みます。`aidb.retrieve_text()` の `intermediate_steps` JSONB 列は、[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](#) が有効になっているステップについて、パイプラインごとの中間結果を結果ます。`deduplicate` デフォルトで `on` は、KB 全体ではなく、パイプラインごとにスコープされます。[重複排除と中間結果](#) を参照してください。

12.1.2 互換性ルール

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

Constraint	Error if violated
Model	<i>Model mismatch: existing KB uses '<X>' but pipeline specifies '<Y>'</i>
Distance operator	<i>Distance operator mismatch: ...</i>
Vector index config	<i>Vector index mismatch: ...</i>

次のページに続く

表 1 – 前のページからの続き

Constraint	Error if violated
Embedding dimensions	<i>Embedding dimension mismatch: existing KB vector column has N dimensions but model '<X>' produces M</i>

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

12.1.3 削除セマンティクス

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

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

12.1.4 接続されたパイプラインの検査

```
SELECT name, model_name, distance_operator, pipeline_names
FROM aidb.knowledge_bases
WHERE name = public.customer_feedback_kb;
__OUTPUT__
      name          | model_name | distance_operator | pipeline_
→names
-----+-----+-----+-----
→-----
public.customer_feedback_kb | bert      | L2                | {feedback_eu_pipe,
→feedback_na_pipe}
```

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

12.1.5 ベクターテーブルスキーマ

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

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

__OUTPUT__

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

-----+-----+-----

1 | 318 | {0}

1 | 412 | {0}

2 | 1042 | {0}

2 | 1057 | {0}

12.2 Hybrid search

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

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

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

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

12.2.1 前提条件

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

pgvector 反復インデックススキャン

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

ネイティブ PostgreSQL FTS 拡張機能は必要ありません。

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

12.2.2 ボリュームソースの制限

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

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

12.3 Hybrid search

12.3.1 次のステップ

- *Setting up native PostgreSQL FTS* — `tsvector` 列、GIN インデックス、およびトリガーをソーステーブルに追加します。
- *Writing hybrid search queries* — すべての組み合わせの SQL パターンリレーショナルフィルター、逆数ランクフュージョン RRF、重み付けリニアフュージョン。サンプルデータと予想される出力を含む作業例が含まれています。
- *Handling embedding service outages* — クエリが正常に結果を返し続けるための 3 層フォールバックチェーン。
- *Pipelines API reference* — `aidb.kb_query_encode()`、`aidb.rerank_text()`、および `aidb.knowledge_bases` ビューの完全なリファレンス。

Setting up native PostgreSQL FTS

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

1. `tsvector` 列を追加します

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

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

1. 列に重み付けフィールドを設定します

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

```
UPDATE my_docs
SET search_vector =
    setweight(to_tsvector(english, coalesce(title, '')), A) ||
```

(次のページに続く)

```
setweight(to_tsvector(english, coalesce(summary, )), B) ||
setweight(to_tsvector(english, coalesce(body, )), C);
```

1. GIN インデックスを作成する

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

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

1. 列を最新の状態に保つトリガーを作成します

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

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

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

1. テストクエリーを実行する

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

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

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

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

次のステップ

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

Writing hybrid search queries

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

- [Setting up native PostgreSQL FTS](#)

ベクトル+関係フィルター

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

```
SET hnsw.iterative_scan = relaxed_order;

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

Vector + ネイティブ PostgreSQL FTS

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

```
SET hnsw.iterative_scan = relaxed_order;

SELECT s.id, s.title,
```

(次のページに続く)

(前のページからの続き)

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

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

重み付けリニアフュージョン

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

重み付けリニアフュージョンと RRF を使用する場合

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

ワークロードに基づいて重みを調整します。

Use case	Suggested weights
Concept/meaning is most important	0.7 semantic, 0.3 keyword
Exact terminology matters as much as meaning	0.5 semantic, 0.5 keyword

次のページに続く

表 4 – 前のページからの続き

Use case	Suggested weights
Keyword precision is critical (for example, code or rule search)	0.3 semantic, 0.7 keyword

ネイティブ PostgreSQL FTS の場合

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

```
WITH semantic AS (
    SELECT source_id,
           (2.0 - (value <=> aidb.kb_query_encode(public.pipeline_docs, connection_
→pooling)::vector)) / 2.0
           AS vec_score
    FROM public.pipeline_docs
    ORDER BY value <=> aidb.kb_query_encode(public.pipeline_docs, connection_
→pooling)::vector
    LIMIT 50
),
keyword AS (
    SELECT id::TEXT AS source_id,
           ts_rank_cd(search_vector, plainto_tsquery(english, connection pooling)) AS_
→fts_score
    FROM my_docs
    WHERE search_vector @@ plainto_tsquery(english, connection pooling)
    ORDER BY fts_score DESC
    LIMIT 50
)
SELECT
    COALESCE(s.source_id, k.source_id) AS id,
    0.6 * COALESCE(s.vec_score, 0) + 0.4 * COALESCE(k.fts_score, 0) AS fusion_score
FROM semantic s
FULL OUTER JOIN keyword k ON s.source_id = k.source_id
ORDER BY fusion_score DESC
LIMIT 10;
```

実用的な例リレーショナル述語フィルタリング

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

ソーステーブルとパイプラインの設定

1. ソーステーブルを作成します。

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

1. サンプルデータを入力します。

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

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

```
SELECT aidb.create_pipeline(
  name          => food_products,
  source         => products,
  source_key_column => id,
  source_data_column => product_name,
  step_1         => KnowledgeBase,
  step_1_options => aidb.knowledge_base_config(
    model      => bert,
    data_format => Text
  )
);
```

1. パイプラインを実行してエンベディングを生成します。

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

標準のセマンティッククエリーフィルタなし

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

```
SELECT * FROM aidb.retrieve_text(public.pipeline_food_products, fast food, topk => 4);
__OUTPUT__
key | value | distance | part_ids | pipeline_name | intermediate_
-> steps
```

(次のページに続く)

(前のページからの続き)

8	Veggie Burger	0.9955745765192575	{0}	food_products	[]
6	Pizza	1.0280680840689795	{0}	food_products	[]
4	Fries	1.0630385212203115	{0}	food_products	[]
7	Sandwich	1.0903594734917037	{0}	food_products	[]

リレーショナルフィルターを使用したカスタムクエリー

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

```
SET hnsf.iterative_scan = relaxed_order;

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

Handling embedding service outages

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

フォールバックチェーンの定義

推奨される劣化チェーン

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

フォールバックの実装

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

プライマリ: ハイブリッドベクター+ネイティブ FTS

```
WITH semantic AS (
    SELECT source_id,
           (2.0 - (value <=> aidb.kb_query_encode(public.pipeline_docs, :query)::vector)) / 2.0 AS vec_score
    FROM public.pipeline_docs
    ORDER BY value <=> aidb.kb_query_encode(public.pipeline_docs, :query)::vector
    LIMIT 50
),
keyword AS (
    SELECT id::TEXT AS source_id,
           ts_rank_cd(search_vector, plainto_tsquery(english, :query)) AS fts_score
    FROM my_docs
    WHERE search_vector @@ plainto_tsquery(english, :query)
    LIMIT 50
)
SELECT COALESCE(s.source_id, k.source_id) AS id,
       0.6 * COALESCE(s.vec_score, 0) + 0.4 * COALESCE(k.fts_score, 0) AS score
FROM semantic s
FULL OUTER JOIN keyword k ON s.source_id = k.source_id
ORDER BY score DESC
LIMIT 10;
```

フォールバック 1: FTS のみ埋め込みサービスなし

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

フォールバック 2: 正規表現インデックスなし

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

正規表現フォールバックは、完全なシークエンシャルスキャンを実行し、大規模なテーブルにスケーリン

グしません。緊急使用または小規模なデータセット用に予約し、アプリケーションにタイムアウトガードを追加して、長時間実行されるクエリを防止します。

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

Hybrid search reference

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

quietb.kb_query_encode()

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

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

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

quietb.rerank_text()

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

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

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

aidb.knowledge_bases view

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

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

次のページに続く

表 5 – 前のページからの続き

Column	Type	Description
<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;
```

12.4 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 パ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。

12.4.1 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"

                                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)
```

12.4.2 例 2 S3 バケット PGFS からの大量埋め込み

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

ステップ 1 外部ストレージとボリュームを構成する

S3 バケットの場所を定義し、AIDB ボリュームにマッピングします。

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

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

ステップ 2 ボリュームコンテンツをリストする

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

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

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

(4 rows)

ステップ 3 ボリュームパイプラインを作成して実行する

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

SELECT aidb.run_pipeline(volume_pipeline__6545);
```

ステップ 4 埋め込みを検証する

```
SELECT source_id, part_ids, value FROM pipeline_volume_pipeline__6545 ORDER BY source_
→id;
__OUTPUT__
```

(次のページに続く)

(前のページからの続き)

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13章 Semantic knowledge bases

セマンティックナレッジベースセマンティック KB は、データベースのスキーマテーブル、ビュー、列、およびそれらのコメントを検索可能なベクターストアにインデックス付けするため、あなたまたは *Agents reference* は、正確な名前ではなく意味で適切なリレーションと列を見つけることができます。これは、**text-to-SQL**の基礎です。自然言語の質問を、独自のスキーマに対する正しい SQL クエリーに変換します。

13.1 セマンティック KB とベクターナレッジベース

セマンティック KB と *Knowledge bases reference* は、どちらもエンベディングを保存し、類似性検索に回答しますが、それらは異なるものにインデックスを付け、異なる問題を解決します。

	Vector knowledge base	Semantic knowledge base
What it embeds	Your data (rows, documents, images)	Your schema (table, view, and column metadata)
What you search	Content, by meaning	Schema structure, by meaning
Created by	<i>aidb.create_pipeline()</i> with a <i>AI pipelines</i>	<i>aidb.create_semantic_kb()</i>
Typical use	Retrieval-augmented generation, semantic search	Natural-language schema discovery, agent tools, text-to-SQL

データベースの **内容**を検索する場合は、ベクトルナレッジベースを使用してください。データベースの **形状**を検索し、質問がマップするテーブルと列を見つけない場合は、セマンティック KB を使用します。

13.2 セマンティック KB インデックスとは

セマンティック KB はスキーマをクロールすると、構造要素ごとに1つのエントリを埋め込みます。

- **テーブル** は、テーブル定義とテーブルコメントを提供します。
- **ビュー** は、ビュー定義とビューコメントを提供します。
- **Columns** は、各列の定義名前、タイプ、NULL 可能性、デフォルトおよび列のコメントに貢献します。

各エントリには、Table、View、またはColumnの `entity_type` があります。コメントは重要です。COMMENT ON ステートメントの自然言語インテントは構造定義とともに埋め込まれ、直接照合できるため、適切にコメントされたスキーマは、より有用なセマンティック KB を生成します。

13.3 セマンティック KB を作成および管理する

`aidb.create_semantic_kb()` を使用して 1 つ以上のスキーマにセマンティック KB を作成します。渡されたモデルにスキーマメタデータを埋め込み、KB の内部ベクトルテーブルを構築し、すぐにスキーマをクロールします。

```
SELECT aidb.create_semantic_kb(
  name    => analytics_kb,
  model   => my_embedding_model,
  schemas => ARRAY[sales],
  auto_processing => Live
);
```

`model` は埋め込みモデルであり、常に必要です。セマンティック KB は、すべてのメタデータおよびそのメンバーである *Semantic aliases* に単一の埋め込みモデルを使用するため、検索は常に同じ空間内のベクトルを比較します。

管理機能は次のとおりです。

Function	Purpose
<code>aidb.create_semantic_kb(model, schemas, auto_processing, pass_triggers, vector_index)</code>	Create a KB over one or more schemas. Returns the KB name.
<code>aidb.list_semantic_kbs()</code>	List every KB with its model, schemas, and processing mode.
<code>aidb.refresh_semantic_kb()</code>	Re-crawl the KB's schemas from scratch to pick up structural changes.
<code>aidb.update_semantic_kb(auto_processing)</code>	Change how the KB keeps itself current.
<code>aidb.semantic_kb_stats(kb)</code>	Return entity counts (<i>total</i> , <i>tables</i> , <i>views</i> , <i>columns</i>) and the number of <i>pending</i> schema changes.
<code>aidb.delete_semantic_kb(n)</code>	Delete a KB and its indexed metadata.

13.3.1 シングル KB のショートカット

セマンティック KB が 1 つだけ存在する場合、すべてのファンクションから `name /kb_name` を省略でき、呼び出しはその KB に解決されます。作成時に名前を省略すると、KB は固定名 `default_semkb` で登録されます。2 番目の KB が存在すると、名前を省略した呼び出しがあいまいエラーで失敗するため、それ以降は名前を明示的に渡します。

13.4 KB を最新の状態に保つ

`auto_processing` モードは、スキーマの変更に KB が反応する方法を制御します。

Mode	Behavior
<i>Disabled</i> (default)	自動更新はありません。スキーマが変更されたときに <code>aidb.refresh_semantic_kb()</code> を呼び出します。
<i>Live</i>	DDL は、スキーマが変更されるとすぐに、影響を受けるリレーションを再ロールします。
<i>Background</i>	バックグラウンドワーカーはキューされた DDL イベントを処理するため、再インデックスは書き込みパス外で発生します。

内部的に、**Background** モードは AIDB のパイプライン機構によって駆動され、**SemanticKB** パイプラインステップがキューに入れられたスキーマ変更イベントを消費します。そのパイプラインを自分で作成または管理することはありません。 `aidb.create_semantic_kb()` が設定します。

13.5 セマンティック KB の使用方法

- **直接、SQL で**。 *Searching a semantic knowledge base* `aidb.semantic_kb_search()` およびその仲間たちを呼び出して、質問がマップするスキーマエンティティを見つけます。
- **エージェントから** KB 検索機能はネイティブの *agent tools* として登録されているため、AIDB エージェントはスキーマを検出し、例 *text-to- SQL エージェント* で質問に回答できます。
- **保存されたクエリーを介して**。繰り返しの質問を *Semantic aliases* としてキャプチャします。意味によって検索ができる自然言語の説明を含むパラメーター化 SQL。

13.6 このセクションでは

Page	What it covers
<i>Searching a semantic knowledge base</i>	The schema-search functions: <i>semantic_kb_search</i> , <i>get_metadata</i> , <i>comment search</i> , and filters
<i>Semantic aliases</i>	Named, parameterized SQL queries with vectorized descriptions, enabling reusable, governed text-to-SQL
例 <i>text-to- SQL エージェント</i>	How an AIDB agent uses the KB's search tools to discover schema and generate correct SQL
<i>Examples</i>	A worked end-to-end example over a sample schema

13.7 関連ドキュメント

- *Agents reference*、ツールとしてセマンティック KB 検索機能呼び出すデータベース内エージェントの構築をカバーします。
- *Semantic knowledge base tools*、KB 機能のネイティブツールカタログエントリをリストします。
- *Knowledge bases reference*、データのセマンティックおよびハイブリッド検索のためのベクター知識ベースをカバーします。

13.7.1 Searching a semantic knowledge base

セマンティック KB がスキーマをクロールしたら、自然言語クエリで検索します。KB は、スキーマのインデックス付けに使用したのと同じモデルでクエリを埋め込み、メタデータがクエリに最も近いテーブル、ビュー、および列を結果ます。すべての検索ファンクションは、KB 名を `kb_name` として取得します。これは、KB が 1 つだけ存在する場合に省略できます。

`promoteb.semantic_kb_search()`

`semantic_kb_search()` は、複合エントリーポイントです。ソースごとのランク付け検索スキーマメタデータと厳選された *Semantic aliases* を実行し、相互ランク融合 RRF で 1 つのランクリストに融合します。「スキーマの何がこれに関連していますか？」に対する単一のランク付けの回答が必要な場合に使用します。

```
SELECT source_type, entity_type, schema_name, relation_name, column_name, rank
FROM aidb.semantic_kb_search(
  query_text => which customers spent the most,
  kb_name    => analytics_kb,
  top_k      => 5
);
```

Parameter	Type	Default	Description
<code>query_text</code>	<code>text</code>	—	自然言語クエリー。
<code>kb_name</code>	<code>text</code>	<code>NULL</code>	検索する KB。1 つの KB のみが存在する場合、単一の KB をターゲットとすることを省略します。
<code>top_k</code>	<code>int</code>	<code>10</code>	返されるランク付けされた行の数。
<code>sources</code>	<code>text[]</code>	<code>NULL</code>	検索するソース。アクティブ <code>schema</code> 、 <code>alias</code> `すべて省略します。`history`、`relationship` は予約されています。
<code>entity_types</code>	<code>text[]</code>	<code>NULL</code>	含めるエンティティタイプ。アクティブ <code>Table</code> 、 <code>View</code> 、 <code>Column</code> 、 <code>Alias</code> 。すべて省略します。
<code>rrf_k</code>	<code>int</code>	<code>60</code>	ランクフュージョンの減衰定数。
<code>min_similarity</code>	<code>double precision</code>	<code>NULL</code>	オプションの類似度フロア。それより低いスコアの行はドロップされます。フロアがない場合は省略します。

一致ごとに 1 行を結果ます。

Column	Description
<code>source_type</code>	行を生成したソース `schema` または `alias` のいずれか。

次のページに続く

表 6 – 前のページからの続き

Column	Description
<i>entity_type</i>	<i>Table</i> 、 <i>View</i> 、 <i>Column</i> 、または <i>`Alias`</i>
<i>schema_name</i>	エンティティが属するスキーマ。
<i>relation_name</i>	テーブルまたはビューの名前。
<i>column_name</i>	列名テーブルレベルおよびビューレベルの一致の場合は空です。
<i>object_ref</i>	一致したオブジェクトへの修飾参照。
<i>definition</i>	埋め込まれた構造定義。
<i>comment</i>	エンティティの <i>`COMMENT ON`</i> テキスト存在する場合。
<i>score</i>	融合された RRF スコア。
<i>rank</i>	融合された結果セット内の 1 ベースのランク。
<i>components</i>	フュージョンをフィードしたソースごとのスコアの JSONB 詳細。

`semantic_kb_search()` は、スキーマエンティティと一致するエイリアスの両方を 1 つのランク付けリストで返すため、これは 例 *text-to- SQL エージェント* ワークフローでの自然な最初の呼び出しです。

スキーマメタデータファンクション

より狭いニーズの場合、4 つのファンクションはスキーマメタデータのみを検索します。これらは同じオプションの `kb_name` および `similarity` スコアコサイン類似度、高いほど近いを共有し、`min_similarity` はオプションのフロアとして、ページング用には `top_k / offset` を使用します。

`promoteb.get_metadata()`

これは最も広範なスキーマ検索であり、完全なメタデータとともに列、テーブル、およびビューを結果ます。

```
SELECT schema_name, relation_name, column_name, entity_type, similarity
FROM aidb.get_metadata(
    kb_name      => analytics_kb,
    query_text   => customer email address,
    top_k        => 10
);
```

パラメーター `kb_name`、`query_text`、`min_similarity`、`top_k` デフォルト 10、`offset` デフォルト 0。 `schema_name`、`relation_name`、`column_name`、`entity_type`、`definition`、`comment`、`definition_vector`、および `similarity` を返します。

`promoteb.get_entity_Definitions()`

テーブルとビュー のみに一致します。

```
SELECT schema_name, relation_name, entity_type, similarity
FROM aidb.get_entity_definitions(
    kb_name      => analytics_kb,
    query_text   => orders and line items,
    entity_types => ARRAY[Table, View]
);
```

パラメーター `kb_name`、`query_text`、`min_similarity`、`entity_types` デフォルト `ARRAY['Table', 'View']`、`top_k`、`offset`。 `schema_name`、`relation_name`、`entity_type`、`definition`、`comment`、`similarity` を返します。

`promoteb.get_column_definings()`

一致する列とテーブルの **定義**、最もリレーンな結果のみを返します、SQL 生成のコンパクトな基礎として役立ちます。

```
SELECT definition
FROM aidb.get_column_definitions(
  kb_name    => analytics_kb,
  query_text => order total amount
);
```

パラメーター `kb_name`、`query_text`、`min_similarity`、`top_k`、`offset` 単一の `definition` 列を結果返します。

`promoteb.search_by_comment()`

スキーマ、テーブル、列に添付された自然言語 `COMMENT ON` テキストのみと一致します。

```
SELECT relation_name, column_name, comment, similarity
FROM aidb.search_by_comment(
  kb_name    => analytics_kb,
  query_text => order status values
);
```

パラメーター `kb_name`、`query_text`、`min_similarity`、`top_k`、`offset` `schema_name`、`relation_name`、`column_name`、`entity_type`、`definition`、`comment`、`similarity` を返します。

これは、スキーマに既に関与したドキュメントを活用する最も直接的な方法です。たとえば、「*order lifecycle state: pending、confirmed、shipped、delivered、Cancelled*」とコメントされた列は、列自身自身の名前が `st` であっても、「*order statusvalues*」のようなクエリーに表示されます。コメント検索は十分に文書化されたオブジェクトに依存しているため、`COMMENT ON` ステートメントを最新の状態に保つことが、セマンティック KB の結果を向上させる最も効果的な方法です。

チューニング結果

- ```min_similarity``` は、再現率を犠牲にして精度を高めます。クエリが何も返さない場合、それを下げますたとえば `0.3`。結果が広範すぎる場合に上げます。
- ```top_k``` / ```offset``` 結果をページスルーします。
- ```entity_types``` 特定のフィールドをハンティングしているときに、`ARRAY['Column']` などの関心の種類に絞り込みます。

次のステップ

- よくある質問を *Semantic aliases* を使用して、再利用可能なパラメーター化クエリーとして保存します。
- 例 *text-to- SQL エージェント* ワークフローでエージェントにこれらの検索をエンドツーエンドで駆動させます。

13.7.2 Semantic aliases

セマンティックエイリアスは、自然言語の説明とペアになった名前付けのパラメーター化 SQL クエリーです。説明が埋め込まれているため、スキーマのメタデータと同じように、意味からエイリアスを見つけることができます。エイリアスは、text-to- SQL の再利用可能な管理レイヤーです。質問ごとに新しい SQL を生成する代わりに、繰り返しの質問は、既に作成およびレビューしたクエリに解決されます。

各エイリアスは、SQL が読み取るスキーマを所有する **すべての**セマンティック KB のメンバーです。1 つだけではなく、AIDB は、クエリから所有する KB を推論し、その KB のモデルを使用して KB ごとに 1 回エイリアスの説明を埋め込むため、各埋め込みはその KB のスキーマベクトルと比較可能なままです。2 つの KB が所有するスキーマは 2 つのエンベディングを生成し、エイリアスは `promoteb.semantic_kb_search()` を使用してどちらの KB からでも見つけられます。

!!!note セマンティックエイリアスは、以下の aidb SQL ファンクションで管理および実行されます。KB `search` ファンクションとは異なり、エイリアスファンクションは *agent tools* として公開されません**。製品の決定により、エージェントはスキーマを検出および読み取りますが、エイリアス自分自身を作成または実行しません。

エイリアスを作成する

`aidb.create_semantic_alias()` は、一意の名前、埋め込む説明、SQL クエリテキスト、およびそのパラメーター定義を取得します。クエリーのプレースホルダーは `${name}` 構文を使用します。クエリーは単一の読み取り専用の SELECT である必要があります。

```
SELECT aidb.create_semantic_alias(
  name      => monthly_revenue,
  description => Total revenue for a given month and year,
  query_text => $$
    SELECT SUM(amount) AS total
    FROM sales.orders
    WHERE EXTRACT(MONTH FROM order_date) = ${month}
    AND EXTRACT(YEAR FROM order_date) = ${year}
  $$,
  params => [
    {"name": "month", "param_type": "integer", "description": "Month number (1-12)"
→  },
    {"name": "year", "param_type": "integer", "description": "Four-digit year"}
  ]
);
```

Parameter	Type	Default	Description
<i>name</i>	<i>text</i>	—	一意のエイリアス名。
<i>description</i>	<i>text</i>	—	人間が読める説明 — これは検索のために埋め込まれるものです。
<i>query_text</i>	<i>text</i>	—	単一の読み取り専用の `SELECT` と、パラメーターの `\${name}` プレースホルダー。
<i>params</i>	<i>jsonb</i>	<i>NULL</i>	パラメーター定義の配列以下を参照してください。
<i>kb_name</i>	<i>text</i>	<i>NULL</i>	オプション。エイリアスをこの KB のみに絞り込みます。クエリのスキーマを所有するすべての KB への埋め込みを省略します。

params JSONB 配列の各エントリは次のとおりです。

Field	Required	Description
<i>name</i>	Yes	`query_text` の `\${name}` プレースホルダーと一致します。
<i>param_type</i>	Yes	PostgreSQL タイプ <i>text</i> 、 <i>integer</i> 、 <i>numeric</i> 、 <i>date</i> など。
<i>description</i>	No	人間が読めるパラメーターの説明。
<i>enum_values</i>	No	パラメーターがセットにコンストレインされている場合、許可される値。

クエリのスキーマが複数の KB にマッピングされている場合、エイリアスはそれぞれに 1 回埋め込まれます。作成があいまいでも失敗することはありません。kb_name を渡して単一の KB に絞り込みます。KB がまだ所有していないスキーマは、それをカバーする KB が作成されるか、後で採用されるまで、エイリアスが埋め込まれないままです。

エイリアスを見つける

`aidb.search_semantic_aliases()` は、自然言語クエリーによってエイリアスを検索します。kb_name がないと、`default_semkb` — 明示的な名前なしで作成されたときに KB が取得する名前 — その KB が存在する場合、またはそれ以外の場合は、保存されたエイリアス埋め込みを持つすべての KB を検索します。クエリは、これらの KB の間の個別のモデルごとに 1 回埋め込まれるため、検索はディメンションセーフなままで、結果はエイリアスごとに最も一致するように重複排除されます。

```
SELECT name, query_text, similarity
FROM aidb.search_semantic_aliases(
    query_text => how much money did we make last month,
    kb_name    => analytics_kb,
    top_k      => 5
);
```

Parameter	Type	Default	Description
<i>query_text</i>	<i>text</i>	—	自然言語クエリー。
<i>min_similarity</i>	<i>double precision</i>	<i>NULL</i>	オプションの類似度フロア。
<i>top_k</i>	<i>int</i>	<i>10</i>	最大の結果。
<i>offset</i>	<i>int</i>	<i>0</i>	ページングオフセット。
<i>kb_name</i>	<i>text</i>	<i>NULL</i>	オプション。検索をこの KB に絞り込みます。上記の優先順位に従って、`default_semkb` またはエイリアス埋め込みを含むすべての KB の解決を省略します。

`name`、`query_text`、および `similarity` を結果ます。エイリアスは `source_type = 'alias'` の `promoteb.semantic_kb_search()` 結果にも表示されるため、1 つの複合検索で一度にスキーマとエイリアスの両方が検出されます。

エイリアスを実行する

`aidb.execute_semantic_alias()` は、パラメーターを置き換えて、名前でエイリアスを実行します。それぞれが JSONB オブジェクトである `result` 行のセットを結果ます。

```
SELECT result
FROM aidb.execute_semantic_alias(
    alias_name => monthly_revenue,
    args       => {"month": 3, "year": 2025}
);
```

Parameter	Type	Default	Description
<i>alias_name</i>	<i>text</i>	—	実行するエイリアス。
<i>args</i>	<i>jsonb</i>	<i>NULL</i>	名前でキー設定されたパラメーター値のオブジェクト。
<i>execute_role</i>	<i>text</i>	<i>NULL</i>	クエリを実行する PostgreSQL ロール — 適切な `SET ROLE` 付与が必要です。

`execute_role` を使用して、接続しているユーザーではなく最小権限のレポートロールでエイリアスを実行します。すべてのエイリアスは作成時に単一の読み取り専用の `SELECT` として検証され、実行時に再検証されるため、エイリアスを使用して書き込みを実行することはできません。

エイリアスの管理

Function	Purpose
<code>aidb.get_semantic_aliases()</code>	List all aliases with <i>name</i> , <i>description</i> , <i>query_text</i> , and <i>param_count</i> .
<code>aidb.get_semantic_alias(ai)</code>	Get one alias's <i>name</i> , <i>description</i> , <i>query_text</i> , and <i>params</i> .
<code>aidb.update_semantic_alias(ai, query_text, kb_name)</code>	Update an alias; changing <i>query_text</i> or <i>kb_name</i> re-infers its owning KB(s) and reconciles embeddings — adding KBs it newly belongs to and dropping ones it no longer does. Changing <i>description</i> re-embeds it in every KB it's already in.
<code>aidb.delete_semantic_alias(ai)</code>	Delete an alias by name.

KB を削除すると、その KB の各エイリアスの埋め込みのみが削除されます。エイリアス定義、および他の所有する KB へのその埋め込みは残ります。同じスキーマをカバーする後の KB は、エイリアスが他の KB に既に埋め込まれている場合でも、そのエイリアスを採用し、その新しい埋め込みを追加します。

次のステップ

- エージェントがスキーマ検索とエイリアスをエンドツーエンドの [例 text-to- SQL エージェント](#) ワークフローに組み合わせる方法を確認します。
- エイリアスをウォークスルーし、[Examples](#) で一緒に検索します。

13.7.3 Text-to-SQL

Text-to- SQL は、「**前四半期に最も費やした顧客はどこですか?**」などの自然言語の質問を、独自のスキーマに対する正しいクエリーに変換します。セマンティック KB により、これが信頼できます。テーブル名と列名を推測するのではなく、質問は最初に KB のインデックス付けされたスキーマに基づいています。

2 つの補完的なパスが [Searching a semantic knowledge base](#) にビルドされます。

- **エージェント駆動**。Agents [reference](#) は、関連するスキーマを検出するツールとして KB の検索機能呼び出し、SQL を生成および実行します。オープンエンドの会話型分析に最適です。
- **アプリケーション駆動**アプリケーションは [エイリアスを見つける](#) および [エイリアスを実行する](#) を呼び出して、確認されたパラメーター化されたクエリーに質問を解決します。決定論的で管理された回答が必要な繰り返しの質問に最適です。

text-to- SQL エンジンとしてのエージェント

セマンティック KB 検索機能は、ネイティブ [agent tools](#) として登録されます。これらのツールを与えられたエージェントは、質問がマップするテーブルと列を自分自身で見つけ、それらに基づいて SQL を書くことができます。利用するツール

Tool	Role in text-to-SQL
AI pipelines	One ranked list of the schema entities (and any matching aliases) relevant to the question.
AI pipelines / AI pipelines / AI pipelines	Narrower schema lookups when the agent needs only tables, only columns, or lean definitions.

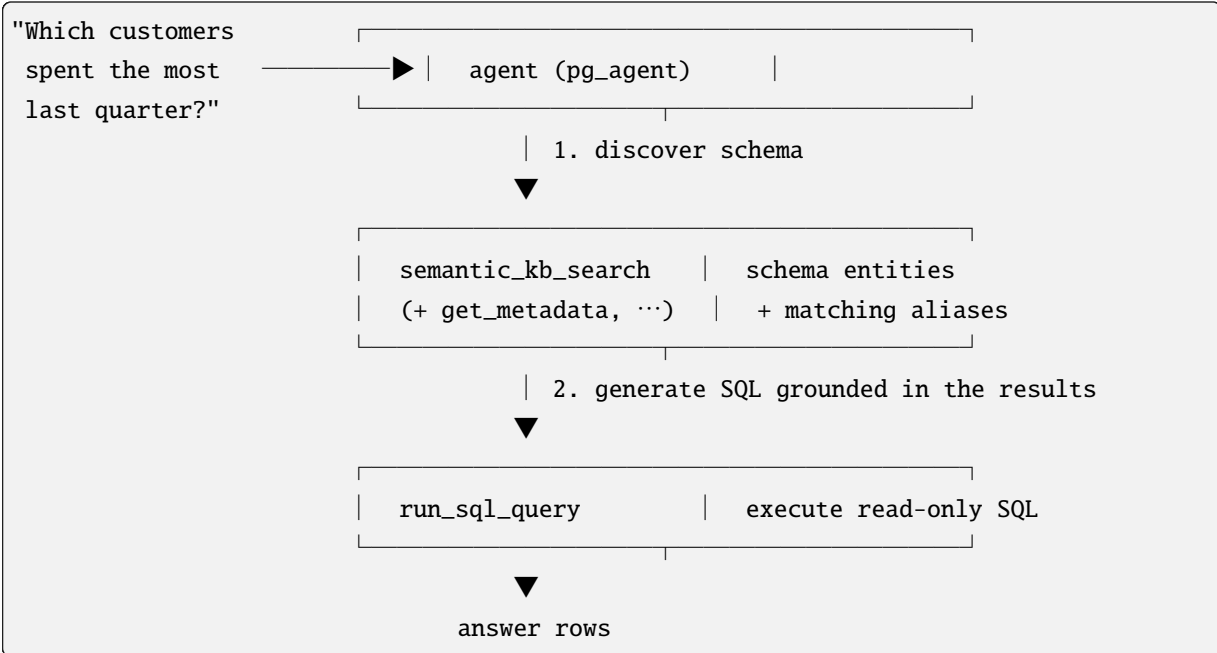
[次のページに続く](#)

表 12 – 前のページからの続き

Tool	Role in text-to-SQL
<i>AI pipelines</i>	Finds entities through the intent captured in <i>COMMENT ON</i> text.
<i>run_sql_query</i>	Runs the read-only SQL the agent generates and returns the rows.

すべての検索ツールは読み取り専用であるため、エージェントは書き込みのリスクなしで自由にスキーマを探索できます。

ワークフロー



1. **Discover.** エージェントは `semantic_kb_search` および必要に応じてより狭いファンクションを呼び出して、質問がマップする正確なテーブル、列、コメントに加えて、既に一致する *Semantic aliases* を取得します。これらは `source_type = 'alias'` で結果に表示されます。
2. **Generate, grounded.** エージェントは、取得した定義に対して SQL を書き込みます。実際のスキーマコンテキスト実際の列名、タイプ、コメントに対する生成は、テーブル名のみに対して生成するよりもはるかに正確です。
3. **Run.** エージェントは `run_sql_query` で SQL を実行し、行を結果ます。

このようなエージェントを構築するには、作成時にセマンティック KB ツールを提供します。 *Managing agents* および *tool catalog* を参照してください。

代わりにエイリアスに手を伸ばす場合

エイリアスは、エージェント生成の決定論的な対応物です。 *Semantic aliases* は、レビューされたパラメータ化された `SELECT` であるため、検証する要求ごとの生成なしで信頼できる結果を返します。次の場合にエイリアスを使用します。

- 同じ質問が繰り返し発生し、1つの正規クエリーに値します。

- エイリアスが単一の読み取り専用の `SELECT` として実行され、`execute_role` を介して最小特権ロールで実行できるガバインド実行が必要です。

エイリアスは `semantic_kb_search` 結果内の表面に表示されるため、エージェントは質問にキュレートされたクエリが存在することを確認できます。ただし、エイリアスの実行は、`aidb.execute_semantic_alias()` を介して直接実行されます。エイリアス機能は意図的にエージェントツールではありません。

raw 生成との比較

- **根拠あり、推測ではありません。** モデルは、関連するリレーションの実際の定義とコメントを参照するため、存在する列を適切なタイプで参照します。
- **信頼できる回答が発生します。** エイリアスとしてキャプチャしたすべての質問は、生成の必要がなくなった質問であり、モデルが間違った SQL を生成する可能性のあるサーフェスを縮小します。
- **構築による読み取り専用。** スキーマ検索ツールとエイリアスはすべて読み取り専用であるため、検出パスと回答パスはデータを変更できません。

次のステップ

- *Examples* のサンプルスキーマのフロー全体を説明します。
- *Searching a semantic knowledge base* の検索機能と *Semantic aliases* のエイリアス機能を確認します。

13.7.4 Example

この例では、小さな `sales` スキーマを介した完全なセマンティックナレッジベースのワークフローを説明します。スキーマの文書化、インデックス付け、意味による検索、および再利用可能なエイリアスとしての繰り返しの質問のキャプチャ。

ステップ 1 エンベディングモデルを登録する

セマンティック KB は、テキスト埋め込みモデルを使用してスキーマメタデータを埋め込みます。最初に 1 つ登録します

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

ステップ 2 スキーマを作成および文書化する

コメントは構造メタデータとともに埋め込まれるため、スキーマを直接文書化すると検索品質が向上します。

```

CREATE SCHEMA sales;

CREATE TABLE sales.customers (
  id      INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  email   TEXT NOT NULL,
  region  TEXT
);

CREATE TABLE sales.orders (
  id          INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  customer_id INT NOT NULL REFERENCES sales.customers(id),
  amount      NUMERIC(12,2) NOT NULL,
  status      TEXT NOT NULL,
  order_date  DATE NOT NULL
);

COMMENT ON TABLE  sales.orders      IS One row per customer order, with its
→monetary amount and lifecycle status.;
COMMENT ON COLUMN  sales.orders.amount IS Order total in USD.;
COMMENT ON COLUMN  sales.orders.status IS Order lifecycle state: pending, confirmed,
→shipped, delivered, cancelled.;
COMMENT ON COLUMN  sales.customers.email IS Customer contact email address.;

```

ステップ 3 セマンティック KB を作成する

sales スキーマを介して KB を作成します。model は必須です。Live 自動処理は、スキーマの変更に応じてインデックスを最新の状態に保ちます。

```

SELECT aidb.create_semantic_kb(
  name    => analytics_kb,
  model   => sales_embeddings,
  schemas => ARRAY[sales],
  auto_processing => Live
);
__OUTPUT__
create_semantic_kb
-----
analytics_kb
(1 row)

```

ステップ 4 インデックスされた内容を確認する

aidb.semantic_kb_stats() は、エンティティ数と保留中のスキーマの変更をレポートします。

```
SELECT * FROM aidb.semantic_kb_stats(kb_name => analytics_kb);
```

__OUTPUT__

total	tables	views	columns	pending
10	2	0	8	0

(1 row)

ステップ 5 意味で検索する

名前が使用されている単語と一致しない場合でも、コンセプトにマップするスキーマエンティティを見つけます。semantic_kb_search() は、スキーマエンティティと一致するエイリアスを 1 つのランク付けリストで結果を返します。

```
SELECT source_type, entity_type, schema_name, relation_name, column_name, rank
FROM aidb.semantic_kb_search(
  query_text => customer email address,
  kb_name    => analytics_kb,
  top_k      => 3
);
```

__OUTPUT__

source_type	entity_type	schema_name	relation_name	column_name	rank
schema	Column	sales	customers	email	1
schema	Table	sales	customers		2
schema	Column	sales	orders	customer_id	3

(3 rows)

コメント検索は、名前だけでなく文書化したインテントを介してエンティティを見つけます。

```
SELECT relation_name, column_name, comment
FROM aidb.search_by_comment(
  query_text => order status values,
  kb_name    => analytics_kb,
  top_k      => 1
);
```

__OUTPUT__

relation_name	column_name	comment
orders	status	Order lifecycle state: pending, confirmed, shipped, delivered...

(1 row)

ステップ 6 定期的な質問をエイリアスとして保存する

「特定の月にいくらの収益がありましたか?」という質問は繰り返し発生するため、毎回 SQL を再生成するのではなく、*Semantic aliases* としてキャプチャします。AIDB は、エイリアスが `analytics_kb` のメンバーであると推論し、クエリーが読み取る `sales` スキーマを所有する KB

```
SELECT aidb.create_semantic_alias(
  name      => monthly_revenue,
  description => Total revenue for a given month and year,
  query_text => $$
    SELECT SUM(amount) AS total
    FROM sales.orders
    WHERE EXTRACT(MONTH FROM order_date) = ${month}
    AND EXTRACT(YEAR FROM order_date) = ${year}
  $$,
  params => [
    {"name": "month", "param_type": "integer", "description": "Month number (1-12)"
    → },
    {"name": "year", "param_type": "integer", "description": "Four-digit year"}
  ]
);
__OUTPUT__
create_semantic_alias
-----
monthly_revenue
(1 row)
```

ステップ 7 エイリアスを見つけて実行する

検索は自然言語の質問をエイリアスに解決します。

```
SELECT name, similarity
FROM aidb.search_semantic_aliases(
  query_text => how much money did we make last month,
  kb_name    => analytics_kb,
  top_k      => 1
);
__OUTPUT__
      name      | similarity
-----+-----
monthly_revenue |         0.83
(1 row)
```

パラメーターを使用して実行します。

```
SELECT result
FROM aidb.execute_semantic_alias(
```

(次のページに続く)

(前のページからの続き)

```
alias_name => monthly_revenue,  
args       => {"month": 3, "year": 2025}  
);  
__OUTPUT__  
      result  
-----  
{"total": 48250.00}  
(1 row)
```

エージェントからの使用

上記はすべて *Agents reference* でも利用できます。エージェントにセマンティック KB *search tools* および *run_sql_query* を与えると、自由形式の質問に自分自身で回答できます。*semantic_kb_search* を呼び出して関連するテーブルと列を検出し、これらの定義に基づいて SQL を生成、実行します- 例 *text-to- SQL エージェント* ワークフロー。

あなたが構築したもの

- 文書化された *sales* スキーマをインデックスするセマンティック KB *analytics_kb* Live モードで最新の状態に保たれます。
- 意味によってリレーションと列を見つける自然言語スキーマ検索。
- 繰り返しの質問の再利用可能なパラメーター化されたエイリアス、意味によって見つけれられ、パラメーターを使用して実行可能。

第14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](#) にあります。
- 推論ファンクション `encode_text`、`generate_text`、`rerank_text` は [Reference → Models](#) にあります。

Page	Functions	What it covers
Text chunking	<code>aidb.chunk_text()</code>	Splitting text into segments for LLM context windows
Data parsing	<code>aidb.parse_html()</code> , <code>aidb.parse_pdf()</code>	Extracting readable text from HTML and PDF inputs
Performing OCR	<code>aidb.perform_ocr()</code>	Extracting text from images using an OCR-capable model
Summarization cookbook	<code>aidb.summarize_text()</code> , <code>aidb.summarize_text_aggregate()</code>	Generating and aggregating AI-powered text summaries
AI inference functions	<code>aidb.encode_text()</code> , <code>aidb.generate_text()</code> , variants	Embedding text and images into vectors and generating text completions
Text reranking	<code>aidb.rerank_text()</code>	Scoring and reordering candidate passages by relevance

!!!note 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。

14.1 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)

```

14.1.1 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。

14.2 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。

14.2.1 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。

14.2.2 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。

[illegible]

- `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。

- `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。

- 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 のようなを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。

`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。

```
SELECT aidb.create_model(
    my_paddle_ocr_model,
    nim_paddle_ocr,
    credentials => {"api_key": "__NVIDIA_NIM_API_KEY__"}::JSONB -- or credentials_
    ↪env to read this from an environment variable instead
);
```

[illegible]

193

(前のページからの続き)

[illegible]

(次のページに続く)

(次のページに続く)

```

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

```

14.4 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。

14.4.1 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

14.5 Text summarization

```

Control model behavior at runtime using `inference_config`:

```
``` sql SELECT * FROM promoteb.summarize_text( input => '長い記事テキスト...', options => pro-
moteb.summarize_config( 'my_llm' , NULL, ' Create a resume_config( 温度=> 0.3、 max_tokens => 50、 seed
=> 42 ) ):json );
```

14.6 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 ```

14.7 Text summarization

<i>The requested adapter is not supported by the model provider: X</i>	Embedding-only provider used for summarization	Use <i>completions</i> , <i>openai_completions</i> , <i>t5_local</i> , or another language provider
<i>Invalid parameters: '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 greater than or equal to 'desired_length'</i>	<i>max_length</i> is smaller than <i>desired_length</i>	Set <i>max_length</i> $\geq$ <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 reduce strategy produced output longer than input failed to reduce size</i>		Increase chunk size (use $\geq$ 50 words or $\geq$ 800 chars)

14.7.1 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
```

(次のページに続く)

(前のページからの続き)

```
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 機能を手に入れます。

エンタープライズユースケース

小売と e コマース — 何百万ものレビューとフィードバックエントリにわたる GROUP BY 製品カテゴリ。好み、苦情、および新しいトレンドのカテゴリごとの概要を取得します。新しいレビューが到着すると、リアルタイムで更新されます。

金融サービス — 融資担当者ノート、借り手とのコミュニケーション、および引受コメントにわたる GROUP BY リスクセグメント。量的モデルが見逃す表面パターン—デフォルトの前にあるフリーテキストフィールドのナラティブシグナル。

ヘルスケアとライフサイエンス — GROUP BY 患者のフィードバック、臨床ノート、および退院概要にわたる GROUP BY 施設、部門、または状態。PHI を外部サービスに公開せずに、ケアの質の変動を特定します。

法とコンプライアンス — GROUP BY 数千のコンプライアンス文書、監査結果、および規制対応にわたる GROUP BY 管轄または規制タイプ。数週間の手動レビューを単一のクエリーに削減します。

営業とカスタマーサクセス — GROUP BY CRM ノート、通話のトランスクリプト、およびメールの概要全体の営業担当者、地域、または取引ステージ。すべての CRM の非構造化テキストから勝敗パターンとコーチングの機会を抽出します。

製造とサプライチェーン — GROUP BY 調達通信、品質レポート、および出荷ノートにわたる GROUP BY サプライヤーまたはコンポーネント。配信メトリックに表示される前に、ナラティブシグナルからサプライチェーンリスクを検出します。

HR とタレント — GROUP BY 退職インタビューの記録と従業員アンケートの回答にわたる部門、在職期間、またはロールレベル。人々が去った本当の理由を表面化します。

その仕組み

AIDB は、データベースのネイティブ状態蓄積フレームワークを使用して、真の PostgreSQL 集計を実装します。

- **State ファンクション** — グループ内の各行からテキストを蓄積し、メモリ内のグループごとの状態を維持します
- **ファイナライズファンクション** - 蓄積されたテキストを構成済みの LLM に送信し、結果を返します
- **チャンキングエンジン** - 大きなテキストグループを管理可能なチャンクに自動的に分割します。ワード数または文字数で構成可能、状況保存のためのオーバーラップあり
- **削減戦略** — `append` はチャンクサマリーを連結します。`reduce` は、出力が収束するまで繰り返し再要約します。データベース内で実行される真のマップ-リデュースパターン
- **カスタムプロンプト** — 各クエリーは独自の指示を指定できます。要約、翻訳、センチメントの抽出、分類、トレンドの特定—同じ機能を無限に柔軟にする
- **モデルに依存しない** — クエリーごとに構成可能な OpenAI 互換エンドポイント GPT-4o、Llama、Mistral、オンプレミスモデルで動作します

すごい瞬間

顧客設定データのテーブルから始めます - 20 の顧客と 5 つの製品カテゴリにわたる 126 行。生データは、非構造化テキストの壁です。

次のクエリーを実行します。

```
SELECT name,
       aidb.summarize_text_aggregate(
         preferences,
         aidb.summarize_text_config(gpt-4o)::json
       ) AS style_profile
FROM customer_preferences
GROUP BY name;
```

数秒で、126 行は 20 行になります。それぞれは、その顧客のすべてのデータポイントから合成されたリッチで一貫したスタイルのプロファイルです。パイプラインはありません。エンジニアリングチームはありません。データベースからデータは残りませんでした。

次に、プロンプトを変更します。

```
SELECT category,
       aidb.summarize_text_aggregate(
         preferences,
         aidb.summarize_text_config(
           gpt-4o,
           prompt => Identify the 3 strongest purchasing signals and recommend
           → inventory actions
         )::json
       ) AS buying_signals
FROM customer_preferences
GROUP BY category;
```

同じデータ。同じ機能。別の質問。異なる洞察力。アナリストは文字列を 1 つ変更し、まったく新しい分析を取得しました。コードの展開、パイプラインの変更、チケットのファイルはありません。

それは、カチッとした瞬間です。アナリストが既に知っているツールを使用して、組織全体の非構造化データが自然言語で照会可能になります。

チャンクとカスタムプロンプトを使用した例

大規模なテキストグループの場合、チャンクとターゲットを絞ったプロンプトを組み合わせます。

```
SELECT product_category,
       aidb.summarize_text_aggregate(
         review_text,
         aidb.summarize_text_config(
           local-llama,
           aidb.chunk_text_config(120, 120, 15, words),

```

(次のページに続く)

(前のページからの続き)

```

        Identify the top 3 quality issues and suggest improvements,
        reduce
    )::json
) AS quality_analysis
FROM product_reviews
GROUP BY product_category;

```

競争力のあるポジション

Capability	Traditional ETL + LLM API	AIDB in-database aggregate
Data movement	Required (security risk)	None (data stays in place)
User interface	Python / custom app	SQL (universal)
Infrastructure	Airflow + queues + compute	PostgreSQL (existing)
Time to first insight	Days to weeks	Minutes
Ad-hoc queries	Requires new pipeline	Change the SQL
Access control	Separate system	PostgreSQL RBAC / RLS
Audit trail	Must be built	PostgreSQL native logging
Compliance	Requires separate review	Inherits DB compliance posture
Model flexibility	Locked to provider	Any OpenAI-compatible endpoint

14.7.2 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,
       aidb.summarize_text_aggregate(
           sn.note_text,
           aidb.summarize_text_config(my-llm)::json
       ) AS summary

```

(次のページに続く)

(前のページからの続き)

```
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,
```

(次のページに続く)

(前のページからの続き)

```

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
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,
         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
```

(次のページに続く)

(前のページからの続き)

```

        )::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 $$
  SELECT aidb.summarize_text_config(
    p_model,
    prompt => Analyze sentiment. Respond ONLY in this format:
OVERALL: [positive/negative/neutral/mixed]
SCORE: [1-10]
  )::json;
$$ LANGUAGE sql;
```

(次のページに続く)

(前のページからの続き)

```

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
WHERE so.status IN (lost, closed_lost)
GROUP BY product_name;

```

- 構成ラッパー機能は、一貫したインターフェイスを提供し、クエリを変更せずにプロンプトを調整できます。

- 戻り値の型は `summarize_text_aggregate` のパラメーター型と一致するように、JSONB ではなく JSON である必要があります。
- 「プリアンブルなし」、「のみ」、「このフォーマットで応答」は効果的なステアリングフレーズです。

9. 要約前のテキストのスクラブ

使用する場合テキストデータには、LLM トークンを無駄にするフィラーワード、挨拶、署名、またはボーラプレートが含まれています。

クリーンアップファンクションを作成し、集計の前に適用します。

```
CREATE OR REPLACE FUNCTION clean_note_text(p_text TEXT)
RETURNS TEXT AS $$
BEGIN
    RETURN regexp_replace(
        regexp_replace(
            regexp_replace(
                regexp_replace(p_text,
                    -- Remove email signatures
                    E(Best regards|Sincerely|Thanks|Cheers), ?\s * \n. * $,
                    , gi),
                -- Remove greeting lines
                E^(Hi|Hello|Dear|Hey)\s+[A-Za-z]+, ?\s * \n,
                , gi),
            -- Remove filler phrases
            E\b(basically|essentially|actually|literally|you know|I mean|sort of|kind
of)\b,
            , gi),
        -- Collapse whitespace
        E\s+, , g);
END;
$$ LANGUAGE plpgsql IMMUTABLE;

-- Use it:
SELECT product_name,
       aidb.summarize_text_aggregate(
           clean_note_text(note_text),
           aidb.summarize_text_config(my-llm)::json
       ) AS summary
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
GROUP BY product_name;
```

- IMMUTABLE PostgreSQL は、繰り返しの呼び出しを最適化します。
- データの正規表現パターンたとえば、内部専門用語、標準ヘッダーなどをカスタマイズします。

- 一般的な削減トークンが 15～30%減少します。これは、チャンクが減り、処理が高速になることを意味します。

レシピの組み合わせ 完全な最適化スタック

最大の速度と品質を得るには、スクラビング+フィルタリング+構成ラッパーを組み合わせます。

```
SELECT p.product_name,
       aidb.summarize_text_aggregate(
         clean_note_text(sn.note_text),
         analyze_top_reasons_config(Why are we losing these deals?, 3)
       ) AS loss_reasons
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
JOIN products p ON so.product_id = p.product_id
WHERE so.status IN (lost, closed_lost)
GROUP BY p.product_name;
```

このパイプライン 関連する行 失われた取引にフィルタリングします。clean_note_text() を介して行ごとにフィ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
2,915 notes	98s	3,200 chars	131K トークンを超えるコンテキストウィンドウ以下を参照してください
3,248 notes	91s	1,700 chars	部分的なデータのみが処理されました

チャンク戦略の比較 50 ノート

Config	Est. LLM calls	Duration	ms/call	Result chars	Status
No chunking	1	12s	11,947	1,443	PASS
words/50/append	32	152s	4,759	12,159	PASS
words/100/append	16	77s	4,832	7,021	PASS
words/200/append	8	48s	6,044	4,781	PASS
words/50/reduce rf=2	63	217s	3,452	545	PASS
words/100/reduce rf=2	31	102s	3,280	475	PASS
words/200/reduce rf=2	15	51s	3,367	767	PASS
words/50/reduce rf=3	50	170s	3,396	410	PASS
words/100/reduce rf=3	25	72s	2,897	867	PASS
words/200/reduce rf=3	12	44s	3,661	535	PASS
chars/200/append	40	246s	6,154	16,355	PASS
chars/400/append	20	137s	6,835	11,793	PASS
chars/800/append	10	66s	6,584	5,897	PASS
chars/200/reduce rf=2	81	971s	11,988	—	FAIL
chars/400/reduce rf=2	41	368s	8,967	441	PASS
chars/800/reduce rf=2	21	144s	6,842	850	PASS

戦略効率の概要

Strategy	Avg LLM calls	Avg total time	Avg ms/call	Avg output
No chunking	1	12s	11,947	1,443 chars
Append	21	121s	5,868	9,668 chars
Reduce rf=2	34	176s	5,182	616 chars
Reduce rf=3	29	95s	3,318	604 chars

予想される規模

Scenario	100 notes	500 notes	1,000 notes
No chunking	28s (1 call)	~60s (1 call)	~100s (1 call, partial data*)
words/100/append	~77s (16 calls)	~385s (80 calls)	~770s (160 calls)
words/100/reduce rf=3	~72s (25 calls)	~180s (62 calls)	~240s (125 calls)

- 1,000 以上のノートでのノーチャンクは、コンテキストウィンドウを超える場合があります。完全を期するためにチャンクを使用します。

推奨される構成

Use case	Config	Why
Fastest possible (< 500 notes)	No chunking	Single LLM call, stays within context window
Fast + detailed	words/200/append	Fewest calls for append, good detail
Balanced detail	words/100/append	More granular, moderate speed
Executive summary	words/100/reduce rf=3	Compressed output, efficient reduce
Maximum compression	words/50/reduce rf=3	Smallest output, most LLM calls

一般的なガイドンスチャンクなしで開始します。グループが~500 ノートを超える場合にのみチャンクを追加します。常に、文字ベースよりもワードベースのチャンクを優先します。Reduce の場合、rf=2 よりも rf=3 を優先します。同様の出力品質で呼び出しの合計数が少なくなります。

既知の制限

コンテキストウィンドウのオーバーフローサイレント

入力テキストの合計がモデルのコンテキストウィンドウを超えると、テキストはサイレントに切り捨てられます。エラーは発生しません。LLM は、適合するものを処理し、残りを無視します。

例えば granite4:1b には、131,072 トークンのコンテキストウィンドウがあります。3,248 ノート 806K 文字、~201K トークンのグループは、これを 53%超えます。結果は有効な概要ですが、データの一部にすぎません。

安全な制限チャンキングなし

Avg note size	Safe note count	Total tokens
250 chars	~500 notes	~33K tokens
500 chars	~250 notes	~33K tokens
1,000 chars	~125 notes	~33K tokens

回避策これらの制限を超える可能性のあるグループにはチャンクを使用します。チャンク化は、コンテキストウィンドウのサイズに関係なく、すべてのデータを処理します。

```
- - Safe for large groups:
aidb.summarize_text_config(
    my-llm,
    aidb.chunk_text_config(200, 200, 20, words),
    NULL,
    append
)::json
```

小さなチャンクでの障害の削減

reduce ストラテジーでは、各要約ラウンドが入力よりも短い出力を生成する必要があります。チャンクサイズが小さい場合、LLM は入力よりも長い概要を生成することが多く、次の原因が発生します。

Text summarization failed to reduce size. The summarized output **is** larger than **or** equal to the **input**.
Consider adjusting the chunking parameters.

障害しきい値

Chunk type	Size	Result
chars/200/reduce	200 chars	Always fails
chars/400/reduce	400 chars	Unreliable
chars/800/reduce	800 chars	Reliable
words/50/reduce	50 words	Reliable
words/100/reduce	100 words	Reliable
words/200/reduce	200 words	Reliable

回避策 **reduce** でワードベースのチャンクを使用しますテストされたすべてのサイズで 100 の合格率。char ベースを使用する必要がある場合は、**>= 800 chars** を使用します。

温度、**max_tokens**、およびシステムプロンプトの制御

デフォルトでは、LLM はプロバイダーのデフォルト温度通常 0.7~1.0 を使用します。これにより、実行ごとに出力に大きな分散が発生する可能性があります。**inference_config** を使用して、コールごとにこれらのパラメーターを制御します。

```
-- Control temperature and max_tokens for more consistent, bounded output
SELECT p.product_name,
       aidb.summarize_text_aggregate(
         sn.note_text,
         aidb.summarize_text_config(
           my-llm,
           NULL,
           Summarize the key points,
           NULL,
           NULL,
           aidb.inference_config(
             temperature => 0.3,
             max_tokens  => 500
           )
         )::json
       ) AS summary
FROM sales_notes sn
JOIN sales_orders so ON sn.order_id = so.order_id
```

(次のページに続く)

(前のページからの続き)

```

JOIN products p ON so.product_id = p.product_id
GROUP BY p.product_name;

-- Override the system prompt for non-summarization tasks
SELECT aidb.summarize_text(
  input    => Customer complained about slow response times and billing errors.,
  options => aidb.summarize_text_config(
    my-llm,
    NULL,
    NULL,
    NULL,
    NULL,
    aidb.inference_config(
      system_prompt => You are a customer support analyst. Classify issues by
→severity.,
      temperature   => 0.2
    )
  )::json
);

```

!!! ヒント 最良の結果を得るには、`inference_config` と制約されたプロンプトを組み合わせます。たとえば、`max_tokens => 200` を設定し、`'Give exactly 3 bullet points, max 1 sentence each.'` のようなプロンプトを使用します。トークン制限はハードキャップとして機能し、プロンプトがフォーマットをガイドします。

エラーなしの空の結果

場合によっては、LLM は処理後に空の文字列を返す場合があります。AIDB は、これをエラーなしの成功結果として保存します。アプリケーションコードで `length(result)` を常に確認してください。

```

SELECT product_name,
  CASE
    WHEN length(summary) = 0 THEN ERROR: Empty result
    WHEN length(summary) < 50 THEN WARNING: Suspiciously short
    ELSE summary
  END AS validated_summary
FROM (
  SELECT p.product_name,
    aidb.summarize_text_aggregate(
      sn.note_text,
      aidb.summarize_text_config(
        my-llm,
        prompt => Your prompt here
      )::json
    ) AS summary

```

(次のページに続く)

(前のページからの続き)

```

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、センテメントラベルは、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ファンクションを使用します。要約する前にスクラブテキスト Recipe 9 を使用してノイズを削減します。

最適化パターン

1. テキストのスクラブ

要約する前に、つなぎ言葉、挨拶、および定型文を削除します。Recipe 9 を参照してください。影響トークンが 15~30%削減され、それに比例して処理が高速化します。

2. コンストレインされたプロンプト

特定の短いフォーマットの出力を生成するように LLM に指示すると、トークン生成時間各呼び出しの最も遅い部分が削減されます。

```

- - Slow (open-ended, LLM may generate 2,000+ chars):
Summarize these notes.

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

```

3. 抽出プレフィルタリング

非常に大規模なデータセットの場合、LLM に送信する前に SQL を使用して最も関連性の高いテキストにプレフィルタリングします。

```

- - Only send notes containing relevant keywords
SELECT aidb.summarize_text_aggregate(
  note_text,
  aidb.summarize_text_config(
    my-llm,
    prompt => What are the pricing concerns?

```

(次のページに続く)

(前のページからの続き)

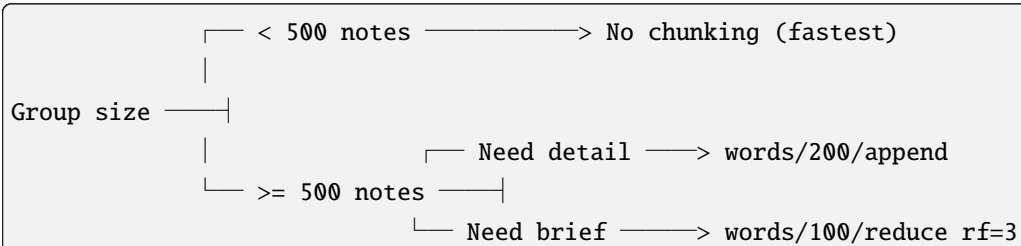
```

)::json
)
FROM sales_notes
WHERE note_text ILIKE %price%
      OR note_text ILIKE %cost%
      OR note_text ILIKE %budget%
      OR note_text ILIKE %expensive%
GROUP BY product_id;

```

インパクト 処理する行が大幅に減少。LLM は関連するテキストのみを認識するため、速度と品質の両方が向上します。

4. 適切な戦略の選択



5. パフォーマンスコストモデル

支配的なコストは LLM 呼び出し数です。入力サイズに応じて、各呼び出しには 3～12 秒かかります。

Step	Time
Network overhead	~50ms
Prompt processing	~1-3s
Token generation	~2-8s
Response parsing	~10ms
Total per call	~3-12s

プロンプト処理は入力長に応じてスケーリングしますが、トークン生成は出力長に応じてスケーリングします。サマリーは入力量に関係なくほぼ固定サイズであるため、コンテキストウィンドウの制限までは、多数の小さな呼び出しよりも、より少なくて大きな呼び出しの方が効率的です。これが、ノーチャン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。

14.8 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。

14.8.1 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(model_name => my_embedder, input => What is pgvector?);
```

__OUTPUT__

encode_text
[0.021,-0.043,0.117, ...]

(1 row)

完全なパラメーターのリファレンスについては、[Models reference](#) を参照してください。

14.8.2 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(
  model_name => my_model_name,
  input      => ARRAY[What is pgvector?, How does RAG work?, Explain embeddings.]
);
```

__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。

14.8.3 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(
  model_name => my_model_name,
  input      => pg_read_binary_file(/path/to/image.png)::BYTEA
);
```

(次のページに続く)

(前のページからの続き)

__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。

14.8.4 quietb.generate_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.generate_text(
  model_name => my_model_name,
  input      => Explain what a knowledge base is in two sentences.
);
```

__OUTPUT__

generate_text

→ -----

A knowledge base is a structured repository of information used to answer queries or

→ ...

(1 row)

生成動作を制御するには、`aidb.inference_config()` によってビルドされた `inference_config` オブジェクトを渡します。呼び出しが `aidb.generate_text(TEXT, TEXT, json)` 署名と一致するように、JSONB 結果を `json` にキャストします。

```
SELECT aidb.generate_text(
  model_name      => my_model_name,
  input           => Summarize the following text: ...,
  inference_config => aidb.inference_config(
    system_prompt => You are a concise technical writer.,
    temperature   => 0.2,
    max_tokens    => 256
  )::json
);
```

完全なパラメーターのリファレンスについては、[Models reference](#) を参照してください。

14.8.5 quietb.generate_text_batch()

単一のバッチ要求で複数のプロンプトのテキスト応答を生成します。

```
SELECT aidb.generate_text_batch(
  model_name => my_model_name,
  input      => ARRAY[
    What is a vector database?,
    What is retrieval-augmented generation?
  ]
);
```

__OUTPUT__

generate_text_batch
A vector database stores high-dimensional vectors ...
Retrieval-augmented generation (RAG) combines a ...

(2 rows)

完全なパラメーターのリファレンスについては、[Models reference](#) を参照してください。

14.8.6 quietb.inference_config()

推論構成オブジェクトをビルドして、言語モデル生成動作を制御します。結果を `aidb.generate_text()` または `aidb.generate_text_batch()` の `inference_config` 引数として、またはパイプラインステップ内の `aidb.summarize_text_config()` の一部として渡します。

```
SELECT aidb.generate_text(
  model_name      => my_llm,
  input           => Write a haiku about PostgreSQL.,
  inference_config => aidb.inference_config(
    temperature => 0.9,
    max_tokens  => 64,
    seed        => 42
  )::json
);
```

完全なパラメーターのリファレンスについては、[Models reference](#) を参照してください。

!!!note 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。

14.9 Text reranking

`aidb.rerank_text()` フังก์ションを使用して、クエリーとの関連性によって一連の候補テキストパッセージを並べ替えます。再ランキングは通常、取得後のステップとして適用され、候補の初期セットがナレッジベースから取得された後、最も関連性の高い結果を上部に表示します。

ステップ 1 リランキングモデルを登録する

```
- - Remote: NVIDIA NIM
SELECT aidb.create_model(
  my_reranker,
  nim_reranking,
  config => aidb.nim_reranking_config(
    api_key => nvapi-..., -- or use create_models own credentials_env parameter.
    instead, to avoid storing this
    model    => nvidia/nv-rerankqa-mistral-4b-v3
  )
);

- - Local: llama.cpp (GGUF), no API key required
SELECT aidb.create_model(
  my_reranker,
  llamacpp_reranking,
  config => jsonb_build_object(
    model, gpustack/bge-reranker-v2-m3-GGUF,
    model_file, bge-reranker-v2-m3-Q4_K_M.gguf,
    n_ctx, 2048
  )
);
```

ステップ 2: 候補セットを再ランク付けする

```
SELECT * FROM aidb.rerank_text(
  model_name => my_reranker,
  query      => What is retrieval-augmented generation?,
  input      => ARRAY[
    RAG combines retrieval and generation to produce grounded answers.,
    Vector databases store high-dimensional embeddings for similarity search.,
    RAG systems retrieve relevant documents before generating a response.
  ]
) ORDER BY logit_score DESC;
```

__OUTPUT__

text	logit_score	id
RAG systems retrieve relevant documents before generating ...	0.97	2
RAG combines retrieval and generation to produce grounded ...	0.91	0

(次のページに続く)

Vector databases store high-dimensional embeddings for ...	0.43	1
--	------	---

(3 rows)

id 列は、input 配列内の各文字列の 0 ベースのインデックスであるため、元のデータに結合して戻すことができます。

完全なパラメーターのリファレンスについては、[Models reference](#) を参照してください。

14.9.1 ナレッジベースの取得後に再ランキングを使用する

一般的なパターンは、ナレッジベースから初期候補セットを取得し、結果を再ランク付けすることです。

```
WITH candidates AS (
  SELECT key, value
  FROM aidb.retrieve_text(my_kb, What is RAG?, topk => 20)
)
SELECT c.key, c.value, r.logit_score
FROM candidates c
JOIN aidb.rerank_text(
  model_name => my_reranker,
  query      => What is RAG?,
  input      => ARRAY(SELECT value FROM candidates ORDER BY key)
) r ON r.text = c.value
ORDER BY r.logit_score DESC
LIMIT 5;
```

!!!note 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。たとえば、NVIDIA NIM リランキングエンドポイントの `aidb.nim_reranking_config()`。llamacpp_reranking にはまだ構成ヘルパーがありません。上記のように、代わりに `jsonb_build_object()` を使用して config をビルドします。[Local models](#) を参照してください。

!!!note logit_score のスケールはプロバイダーによって異なります。上記の例のように、固定範囲の正規化スコアを結果とすることができるものもありますが、llamacpp_reranking は生の境界なしのモデルロジットを結果とします。異なるプロバイダーまたは呼び出し間ではなく、1 つの `aidb.rerank_text()` 呼び出しとプロバイダー内のスコアを比較します。

第15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
Models reference	Catalog views, model CRUD (<code>create_model</code> , <code>get_model</code> , <code>delete_model</code>), HCP model functions, inference functions (<code>encode_text</code> , <code>generate_text</code> , <code>rerank_text</code>), and all model config helpers	Integrating models
Knowledge bases reference	Knowledge base views and stats, retrieval functions (<code>retrieve_key</code> , <code>retrieve_text</code>), volume functions, and auto-processing	Knowledge bases reference
Operations reference	Pipeline error log functions (<code>get_error_logs</code> , <code>clear_error_logs</code> , <code>requeue_pipeline_errors</code>)	Pipeline error log
Functions reference	Standalone SQL functions: <code>chunk_text</code> , <code>parse_html</code> , <code>parse_pdf</code> , <code>perform_ocr</code> , <code>summarize_text</code>	SQL functions
Pipelines reference	Pipeline types, views, CRUD functions, step config helpers, and vector index config helpers	AI pipelines
Agents reference	Agent CRUD, conversation functions, structured output helpers, and catalog views	Agents reference
Tools reference	Tool catalog view, SQL tool registration, MCP server management, and parameter helpers	Tools reference

15.1 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。

15.1.1 構成モデル

モデルの動作は 2 つの方法で制御され、特定の設定でどちらを使用するかが重要です。

- **Creation-time ``config``** — プロバイダー固有の **モデル構成ヘルパー** `aidb.bert_config()`、`aidb.completions_config()` などを使用して、`aidb.create_model()` を呼び出したときに設定されます。これは、モデルの永続的な保存された構成です。
- **Call-time ``inference\_config``** — オプションの呼び出しごとのオーバーライド。`quiteb.inference_config()` でビルドされ、推論ファンクションに直接渡されます。その 1 つの呼び

出しのみについてモデルに保存されている構成をオーバーライドします。オーバーライドしないものは、作成時に設定された値にフォールバックします。

一部の推論ファンクションのみが呼び出し時オーバーライドを受け入れます。残りはモデルの作成時の `config` によって完全に修正されます。

Function	Call-time override?
<code>aidb.generate_text()</code> , <code>aidb.generate_text_batch()</code>	Yes — pass <code>aidb.inference_config()</code> (or raw JSON) as the <code>inference_config</code> argument.
<code>aidb.summarize_text()</code> , <code>aidb.summarize_text_aggregate()</code>	Yes — via the <code>inference_config</code> field of <i>AI pipelines</i> .
<code>aidb.encode_text()</code> , <code>aidb.encode_text_batch()</code> , <code>aidb.encode_image()</code>	No. Behavior is fixed by <code>config</code> at creation time.
<code>aidb.rerank_text()</code>	No. Behavior is fixed by <code>config</code> at creation time.
<code>aidb.perform_ocr()</code>	No — its <code>options</code> parameter only selects which registered model to run, not inference settings.

この非対称は、各機能のニーズに従っています。テキスト生成には、呼び出しごとに調整する価値のあるランタイムノブがあります温度、システムプロンプトなど。埋め込み、リランキング、および OCR は行わないため、オーバーライドするものではありません。

15.1.2 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
<code>server_name</code>	name	モデルプロバイダーの名前
<code>server_description</code>	text	プロバイダーの説明
<code>server_options</code>	text[]	使用可能な構成オプション

`helpb.models`

デフォルトおよびユーザー作成のモデルを含む、レジストリ内のすべてのモデルをリストします。

Column	Type	Description
<code>name</code>	text	モデルのユーザー定義の名前
<code>provider</code>	text	モデルプロバイダー名
<code>options</code>	text[]	モデルの設定済みオプション
<code>functions</code>	text[]	モデルのプロバイダーがサポートする内部機能識別子たとえば <code>aidb-openai-text-embeddings</code> 、 <code>aidb-openai-text-completion</code>

例

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

__OUTPUT__

name	provider	options	functions
bert	bert_local	{ "config": {} }	{ aidb-openai-text-embeddings }
clip	clip_local	{ "config": {} }	{ aidb-openai-text-image-embeddings }
t5	t5_local	{ "config": {} }	{ aidb-openai-text-embeddings, aidb-openai-text-completion }

(3 rows)

15.1.3 モデル管理機能

promoteb.create_model

新しいモデルを AIDB モデルレジストリに登録します。

パラメーター

Parameter	Type	Default	Description
<i>name</i>	TEXT	Required	モデルの一意の名前。
<i>provider</i>	TEXT	Required	プロバイダー名 `aidb.model_providers`を参照してください。
<i>config</i>	JSONB	'{}'	プロバイダー固有の構成。 モデル構成ヘルパー でビルドします。
<i>credentials</i>	JSONB	'{}'	プロバイダーの資格情報たとえば { "api_key": "..."}。
<i>replace_credentials</i>	BOOLEAN	false	`true`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。
<i>validate</i>	BOOLEAN	true	`true`の場合、作成時にモデルをブローブして、機能していることを確認します。 :ref: 検証<検証> を参照してください。
<i>credentials_env</i>	TEXT	NULL	資格情報を保存する代わりに、読み取る環境変数の名前。 環境変数からの資格情報 を参照してください。 `credentials`と相互排他的。

環境変数からの資格情報

!!!note プロバイダーの API キーまたはパスワードをデータベーステーブルにまったく置いたくないですか、許可が制限されている場合でも。credentials の代わりに credentials_env を渡し、資格情報を読み取る Postgres 環境変数を名前付けます。変数の **名前**のみが保存されます。値自身は、モデルが使用されるたびに Postgres バックエンドプロセスの環境から新しく読み取られ、データベースに永続化されることはありません。

credentials オブジェクトの代わりに変数名を渡します。

```
SELECT aidb.create_model(
  name          => my_openai,
  provider      => openai_embeddings,
  config        => aidb.embeddings_config(model => text-embedding-3-small),
  credentials_env => AIDB_OPENAI_API_KEY
);
```

参照される変数は、ベアシークレット api_key として使用または JSON オブジェクト {"basic_auth": "..."} など credentials 自分自身が受け入れるのと同じ形状を保持できます。credentials および credentials_env は相互排他的です。両方を渡すとエラーが発生します。

セキュリティのため、環境変数の名前は *aidb.env_var_allowed_prefix* デフォルトでは AIDB_ で構成されたプレフィックスで始まる必要があります。これにより、モデルまたは MCP サーバー構成が Postgres ホストで任意の無関係の環境変数を命名し、その値を抽出するのを防ぎます。同じメカニズムと同じ GUC が MCP サーバー認証に適用されます。*headers_env* を参照してください。

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"
}
```

検証

デフォルト validate => true では、aidb.create_model() は登録がコミットする前に最小限のプロープ推論を実行し、モデルが機能していることを確認します。

- **ローカルモデル**がダウンロードされてロードされ、テスト推論が実行されるため、不正なモデル識別子またはロード不可の重みは、最初を使用するときではなく作成時に失敗します。
- **外部モデル**は小さなテスト要求を受信します。したがって、無効な API キー、間違った URL、または到達不能なエンドポイントがすぐにキャッチされます。

プロープが失敗した場合、aidb.create_model() はエラーを発生させ、何も登録しません。validate => false を渡してプロープをスキップします。たとえば、大規模なダウンロードを最初を使用するまで延期したり、エンドポイントまたは資格情報が利用できる前にモデルを登録したりします。モデルは後で aidb.validate_model() を使用して検証できます。

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>	text[]	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

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)

promoteb.validate_model

登録されたモデルをプローブして、最小限のテスト推論を実行して、機能していることを確認します。
 validate => false 以外の場合、aidb.create_model() によって自動的に呼び出されます。直接呼び出して、既存のモデルを再チェックすることもできます。

パラメーター

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

返品

TEXT — プローブされたモデル機能たとえば `text embedding`、`language`、`reranking`、`image embedding`、または OCR など。プローブが失敗した場合にエラーが発生します。

例

```
SELECT aidb.validate_model(my_bert);

__OUTPUT__
validate_model
-----
text embedding
(1 row)
```

15.1.4 HCP モデルファンクション

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

`promoteb.list_hcp_models`

HM で現在実行されているモデルをリストします。

返品

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

例

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

```
__OUTPUT__
```

```

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

promoteb.create_hcp_model

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

パラメーター

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

promoteb.sync_hcp_models

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

返品

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

15.1.5 推論ファンクション

!!!note `aidb.generate_text()` および `aidb.generate_text_batch()` の以前の名称は `aidb.decode_text()` および `aidb.decode_text_batch()` でした。古い名称はまだ機能しますが、非推奨であり、将来のバージョンで削除されます。新しいコードでは新しい名称を使用します。

helpb.encode_text

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

パラメーター

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

返品

real[] ー入力テキストの埋め込み。 **vector** にキャストして pgvector 演算子で使します。

例

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

helpb.encode_text_batch

1 回の呼び出しでテキスト文字列の配列をベクトルにエンコードします。

パラメーター

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

返品

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

例

```
SELECT * FROM aidb.encode_text_batch(bert_local, ARRAY[The quick brown fox, Lorem
↳ ipsum dolor sit amet]);
```

promoteb.encode_image

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

パラメーター

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

返品

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

例

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

promoteb.generate_text

指定されたモデルを使用して、プロンプトからテキスト応答を生成します。

パラメーター

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

返品

TEXT — 生成されたレスポンス

例

```

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

-- With inference configuration
SELECT aidb.generate_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
);

```

helpb.generate_text_batch

1 回の呼び出しで一連のプロンプトに対するテキスト応答を生成します。

パラメーター

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

返品

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

例

```

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

```

helpb.rerank_text

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

パラメーター

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

返品

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

15.1.6 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.generate_text()`、`aidb.generate_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。
<i>repeat_penalty</i>	REAL	NULL	トークンを繰り返す場合のペナルティ。`1.0` =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.generate_text(
  my_llama,
  What is machine learning?,
  aidb.inference_config(temperature => 0.5, max_tokens => 150)::json
);

-- Full configuration
SELECT aidb.generate_text(

```

(次のページに続く)

```

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

```

ツール呼び出しと構造化出力

さらに3つの推論設定が存在しますが、`aidb.inference_config()` の名前付けパラメーターとして公開されていません。代わりに生の JSON としてビルドし、他の設定も必要な場合は `||` を使用してヘルパーの出力とマージします。`llamacpp\_generate`、`openai\_responses` + `openai\_responses\_azure`、および `anthropic\_messages` + `anthropic\_messages\_azure` / `anthropic\_messages\_bedrock` のみがこれらを尊重します。他のすべてのプロバイダーはそれらを拒否します。

Field	Type	Description
<code>tools</code>	JSONB array	OpenAI 互換の `tools` 配列 — モデルが呼び出すことができるファンクション定義。
<code>tool_choice</code>	JSONB	OpenAI 互換 `tool_choice` 特定のツールを強制するための文字列 `auto`、`required`、または `none`、または <code>{ "type": "function", "function": { "name": "..."} }</code> 。
<code>response_format</code>	JSONB	構造化された出力。 <code>{ "type": "json_schema", "json_schema": { "schema": {...} } }</code> は、レスポンスを JSON スキーマにコンストレインします。`openai_responses` および `anthropic_messages` およびそれらのバリエーションでの `tools` / `tool_choice` と相互排他的。

!!!note `openai_responses` および `anthropic_messages` およびそれらの Azure/Bedrock バリエーションは、これらを各 API のネイティブ `tools` / `tool_choice` 要求フィールドにマッピングし、実際のツールコール応答を解析します。 [OpenAI Responses API](#) および [Anthropic Messages API](#) を参照してください。他のすべての

リモートプロバイダー `openai_completions`、`gemini`、`nim_completions` などは、`tools/tool_choice/response_format` を完全に拒否します。この拒否は、`aidb.generate_text()` のような直接呼び出しにのみ適用されます。これらのプロバイダー `t5_local` を除くは、引き続き [Agents reference](#) をバックギンでできます。エージェントフレームワークは、これらの3つのフィールドを使用するのではなく、プロンプトに記述しモデルの応答を解析することにより、ツール呼び出しのシミュレーションにフォールバックします。

```
-- Tool calling
SELECT aidb.generate_text(
  my_llamacpp_chat,
  What is the weather in Paris?,
  (aidb.inference_config(temperature => 0.2) || jsonb_build_object(
    tools, jsonb_build_array(
      jsonb_build_object(
        type, function,
        function, jsonb_build_object(
          name, get_weather,
          description, Get current weather for a city,
          parameters, jsonb_build_object(
            type, object,
            properties, jsonb_build_object(city, jsonb_build_object(type, string)),
            required, jsonb_build_array(city)
          )
        )
      )
    ),
    tool_choice, auto
  ))::json
);

-- Structured output
SELECT aidb.generate_text(
  my_llamacpp_chat,
  Extract the city from: I live in Paris.,
  jsonb_build_object(
    response_format, jsonb_build_object(
      type, json_schema,
      json_schema, jsonb_build_object(
        schema, jsonb_build_object(
          type, object,
          properties, jsonb_build_object(city, jsonb_build_object(type, string)),
          required, jsonb_build_array(city)
        )
      )
    )
  )
);
```

(次のページに続く)

(前のページからの続き)

```

)::json
);

```

15.1.7 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。

例

```

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 :ref:`aidb.max_tokens_config()` <#aidbmax_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。

例

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

!!!note OpenAI モデルを呼び出す信頼性の高いネイティブツールでエージェントを使用する場合、代わりに [promoteb.openai_responses_config](#) を使用します。2つのプロバイダーの違いについては [ツール呼び出しと構造化出力](#) を参照し、いずれか一方を中心にエージェントを構築するには [Agents reference](#) を参照してください。

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。
<i>format</i>	TEXT	NULL	形式 'default'、'legacy'、または 'both'

例

```
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。

例

```
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。

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。

例

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

promoteb.llama_config

llama_instruct_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。

例

```
SELECT aidb.create_model(my_llama, llama_instruct_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。
<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。

例

```
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。

例

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

!!!note 他のリモートプロバイダーとは異なり、Gemini の `api_key` は、`aidb.create_model()` の別個の `credentials` パラメーターを介して渡されるのではなく、`config` のプレーンフィールドです。これは、より制限された資格情報ストアではなく、モデルの構成の一部として保存されます。表示するキーよりも多くのロールに `aidb.models.options` が可視されている場合に注意してください。

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。

例

```
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。

例

```
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。
<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。

例

```
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 :ref:`aidb.max_tokens_config()` <#aidbmax_tokens_config>` を使用します。

例

```
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。

例

```
SELECT aidb.create_model(my_or_embedder, openrouter_embeddings,
  config => aidb.openrouter_embeddings_config(
    mistral/mistral-embed,
```

(次のページに続く)

(前のページからの続き)

```

    api_key => sk-or...
  )
);

```

promoteb.openai_responses_config

openai_responses および openai_responses_azure プロバイダーの場合。

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	モデル識別子たとえば <i>gpt-5.1</i> 。
<i>api_key</i>	TEXT	NULL	認証用の API キー。
<i>basic_auth</i>	TEXT	NULL	基本的な認証資格情報 <i>user:password</i> 。
<i>url</i>	TEXT	NULL	API エンドポイント URL。 `openai_responses` の OpenAI の Responses エンドポイントのデフォルト。 `openai_responses_azure` に必要な普遍的なデフォルトなし-リソースのエンドポイントを使用します。
<i>max_concurrent_requests</i>	INTEGER	25	エンドポイントへの最大同時要求。
<i>system_prompt</i>	TEXT	NULL	デフォルトのシステムプロンプト、Responses API のネイティブ `instructions` フィールドを介して送信されます。
<i>temperature</i>	DOUBLE PRECISION	NULL	サンプリング温度。
<i>max_output_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>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。

例

```

-- OpenAI
SELECT aidb.create_model(
  my_gpt,
  openai_responses,
  config      => aidb.openai_responses_config(model => gpt-5.1),
  credentials => [{"api_key": "sk-..." }::JSONB

```

(次のページに続く)

(前のページからの続き)

```
);

-- Azure AI Foundry
SELECT aidb.create_model(
  my_gpt_azure,
  openai_responses_azure,
  config      => aidb.openai_responses_config(
    model => gpt-5.1,
    url   => https://<resource>.openai.azure.com/openai/v1/responses
  ),
  credentials => {"api_key": "..."}::JSONB
);
```

このプロバイダーは、[ツール呼び出しと構造化出力](#) で説明されている `tools`、`tool_choice`、および `response_format` フィールドをネイティブにサポートしています。

helpb.anthropic_messages_config

`anthropic_messages`、`anthropic_messages_azure`、および `anthropic_messages_bedrock` プロバイダーの場合。

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	モデル識別子たとえば、 <i>claude-opus-4-6</i> 、または Bedrock バリエーションの Bedrock モデル ID など。
<i>api_key</i>	TEXT	NULL	認証用の API キー。`anthropic_messages_bedrock` の場合、これは Bedrock API キーベアラートークンであり、AWS SigV4 資格情報ではありません。
<i>basic_auth</i>	TEXT	NULL	基本的な認証資格情報 <i>user:password</i> 。
<i>url</i>	TEXT	NULL	API エンドポイント URL。`anthropic_messages` のデフォルトは `api.anthropic.com` です。 `anthropic_messages_azure` リソースのメッセージエンドポイントと `anthropic_messages_bedrock` リージョンの `bedrock-runtime` ベースエンドポイント-モデル ID は自動的に追加されます。

次のページに続く

表 36 – 前のページからの続き

Parameter	Type	Default	Description
<i>max_concurrent_requests</i>	INTEGER	25	エンドポイントへの最大同時要求。
<i>system_prompt</i>	TEXT	NULL	デフォルトのシステムプロンプト。メッセージ API のネイティブ `system` フィールドを介して送信されます。
<i>temperature</i>	DOUBLE PRECISION	NULL	サンプリング温度。
<i>max_tokens</i>	INTEGER	4096	生成する最大トークン。Anthropic の Messages API にはこのフィールドが必要です。AIDB は省略するとデフォルトします。
<i>top_p</i>	DOUBLE PRECISION	NULL	nucleus サンプリングしきい値。
<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。

例

```

-- Anthropic direct
SELECT aidb.create_model(
  my_claude,
  anthropic_messages,
  config      => aidb.anthropic_messages_config(model => claude-opus-4-6),
  credentials => {"api_key": "sk-ant-..."}::JSONB
);

-- Azure AI Foundry
SELECT aidb.create_model(
  my_claude_azure,
  anthropic_messages_azure,
  config      => aidb.anthropic_messages_config(
    model => claude-opus-4-6,
    url   => https://<resource>.services.ai.azure.com/anthropic/v1/messages
  ),
  credentials => {"api_key": "..."}::JSONB
);

-- AWS Bedrock
SELECT aidb.create_model(
  my_claude_bedrock,
  anthropic_messages_bedrock,
  config      => aidb.anthropic_messages_config(

```

(次のページに続く)

(前のページからの続き)

```

model => anthropic.claude-opus-4-6-v1:0,
url   => https://bedrock-runtime.us-east-1.amazonaws.com
),
credentials => {"api_key": "..."}::JSONB
);

```

このプロバイダーは、[ツール呼び出しと構造化出力](#) で説明されている `tools`、`tool_choice`、および `response_format` フィールドをネイティブにサポートしています。Anthropic のメッセージ API には、ネイティブの構造化出力フィールドがないため、`response_format` は合成ツールへの強制呼び出しとして実装されます。機能的には同等ですが、`response_format` および `tools` / `tool_choice` は同じ呼び出しで使用できないことを意味します。

15.2 Knowledge bases reference

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

15.2.1 カタログビュー

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

helpb.Knowledge_base_metrics

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

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

例

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

__OUTPUT__

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

(2 rows)

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

15.2.2 タイプ

helpb.DistanceOperator

取得中に使用されるベクトル距離ファンクション。

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

helpb.PipelineDataFormat

パイプラインソースまたはボリュームのデータのフォーマット。

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

helpb.PipelineAutoProcessing モード

ナレッジベースパイプラインの自動処理モード。

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

15.2.3 ナレッジベース機能

パイプラインの自動処理モードを変更するには、[promoteb.update_pipeline](#) を使用します。

promoteb.retrieve_key

ソースデータを取得せずに、上位一致するエンベディングのソースレコードキーと距離を結果ます。**part_ids** では、マルチステップパイプラインの各ステップに一致するパーツの ID も結果ます。たとえば、単一のソースレコードが **ChunkText** ステップによって 10 の部分に分割されている場合、**part_ids** は、どのチャッ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>schema.vector_table</i> 。
<i>query</i>	TEXT or BYTEA	Required	テキストまたは画像のバイトを照会します。エンベディングモデルのデータ形式と一致する必要があります。

次のページに続く

表 42 – 前のページからの続き

Parameter	Type	Default	Description
<i>topk</i>	INTEGER	<i>1</i>	返される結果の数。正の整数である必要があります。`0`または負の値はエラーを発生します。
<i>deduplicate</i>	BOOLEAN	<i>true</i>	複数の「部分」チャンク/ページなどが一致する場合でも、各ソースの結果を 1 回だけ返します。 重複排除と中間結果 を参照してください。

返品

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

例

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

```
__OUTPUT__
```

```
key | distance | part_ids | pipeline_name
-----+-----+-----+-----
1042 | 1.1931772046185758 | {0,0} | feedback_eu_pipe
318 | 1.1980633963685476 | {2,0} | feedback_na_pipe
1057 | 1.2150919866080878 | {2,0} | feedback_eu_pipe
(3 rows)
```

helpb.retrieve_text

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

パラメーター

Parameter	Type	Default	Description
<i>knowledge_base_name</i>	TEXT	Required	ナレッジベースの名前 <i>schema.vector_table</i> 。
<i>query</i>	TEXT or BYTEA	Required	テキストまたは画像のバイトを照会します。エンベディングモデルのデータ形式と一致する必要があります。
<i>topk</i>	INTEGER	1	返される結果の数。正の整数である必要があります。0 または負の値はエラーを発生します。
<i>deduplicate</i>	BOOLEAN	true	複数の「部分」チャンク/ページなどが一致する場合でも、各ソースの結果を 1 回だけ返します。重複排除と中間結果を参照してください。

返品

Column	Type	Description
<i>key</i>	text	ソースレコードキー
<i>value</i>	text	ソーステキストのコンテンツ。NULL 元のソースコンテンツが非テキスト PDF または画像などの場合。他の列はまだ表面一致するメタデータです。
<i>distance</i>	double precision	クエリーからのベクトル距離
<i>part_ids</i>	bigint[]	ソースレコードのどの部分が一致したのかチャンクまたはページなど
<i>pipeline_name</i>	text	このソースレコードを処理するパイプラインの名前
<i>intermediate_steps</i>	jsonb	マッチングは、 <i>intermediate_destination</i> で構成されたステップからの結果です。つまり、 <i>part_ids</i> に属する値。パイプライン内のステップがない場合、中間ストレージが有効になっている場合、空 []。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を参照してください。

intermediate_steps は、ステップごとに 1 つのエントリを含む JSON 配列であり、中間ストレージが有効になっており、それぞれの形状は次のとおりです。

```
{ "value": "...", "operation": "ChunkText", "step_order": 1, "destination_table":  
  ↳ "public.pipeline_my_pipeline_step_1" }
```

例

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

__OUTPUT__

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

(2 rows)

promoteb.delete_knowledge_base

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

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

パラメーター

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

例

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

15.2.4 ボリューム管理機能

promoteb.create_volume

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

パラメーター

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

例

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

helpb.volumes

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

例

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

promoteb.delete_volume

AIDB ボリュームを削除します。

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

パラメーター

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

例

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

15.3 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。

15.3.1 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
<code>p_pipeline_name</code>	TEXT	Required	パイプラインの名前。
<code>p_source_id</code>	TEXT	NULL	特定のソースレコード ID にフィルターします。
<code>p_pipeline_step</code>	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。
<code>p_error_category</code>	<i>helpb.ErrorBlocking</i>	NULL	カテゴリによるフィルター <i>RecordTemporary</i> 、 <i>RecordPermanent</i> 、 <i>PipelineTemporary</i> 、または <i>PipelinePermanent</i>
<code>p_limit</code>	INTEGER	NULL	返される行の最大数。
<code>p_offset</code>	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
<code>id</code>	bigint	エラーログエントリ ID
<code>source_id</code>	text	失敗したソースレコード
<code>part_ids</code>	bigint[]	部分識別子チャンクレコードの場合
<code>pipeline_step</code>	smallint	障害が発生したステップ番号
<code>step_operation</code>	<i>helpb.PipelineStepOperation</i>	オペレーション名 例 <i>ParsePdf</i> 、 <i>ChunkText</i>
<code>error_message</code>	text	エラーの説明
<code>error_category</code>	<i>helpb.ErrorBlocking</i>	エラーカテゴリ
<code>failed_at</code>	timestampz	障害のタイムスタンプ

次のページに続く

表 50 – 前のページからの続き

Column	Type	Description
<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
<i>last_seen_at</i>	timestampz	このエラーが最後に発生したタイムスタンプ。繰り返しのたびに更新します。`failed_at` は、最初に出現したときに固定されたままです。

例

```

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

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

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

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

```

helpb.get_error_log_summary

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

パラメーター

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

返品

Column	Type	Description
<i>pipeline_step</i>	smallint	ステップ番号
<i>step_operation</i>	<i>helpb.PipelineStepOperation</i>	オペレーション名
<i>error_category</i>	<i>helpb.ErrorBlocking</i>	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_error_log_summary(my_pipeline);

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

promoteb.get_all_error_summaries

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

返品

Column	Type	Description
<i>pipeline_name</i>	text	パイプライン名
<i>pipeline_step</i>	smallint	ステップ番号
<i>step_operation</i>	<i>helpb.PipelineStepOperation</i>	オペレーション名
<i>error_category</i>	<i>helpb.ErrorBlocking</i>	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__
```

(次のページに続く)

(前のページからの続き)

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
p_pipeline_name	TEXT	パイプラインの名前。
p_error_ids	BIGINT[]	削除するエラーログエントリ ID の配列。

返品

BIGINT — 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.requeue_pipeline_errors

7. 5.0 で導入されました。ターゲットのレコードレベルのエラーログエントリを削除し、パイプラインの状態サーフェスでソースレコードをダーティとしてマークします。次の通常の実行で再処理されます。呼び出し自分自身は処理を行いません。ワークフローについては、[失敗したレコードの再キューイング](#) を参照してください。

パラメーター

Parameter	Type	Description
<i>p_pipeline_name</i>	TEXT	パイプラインの名前。
<i>p_error_ids</i>	BIGINT[]	再キューイングするエラーログエントリ ID の配列。

返品

入力 ID ごとに 1 行、昇順 *error_id*。

Column	Type	Description
<i>error_id</i>	bigint	入力配列のエラーログエントリ ID。
<i>action</i>	text	<i>requeued</i> 、 <i>skipped_pipeline_level</i> 、または <i>not_found</i> 以下を参照してください。

- *requeued* レコードレベルの行が削除され、ソースがダーティとしてマークされました。テーブルソースは状態イベントをエンキューします。ボリュームソースは、ワンショットの *retry_requested_at* マーカーを設定します。
- *skipped_pipeline_level* エントリには NULL *source_id* があり、そのまま残されます。根本的な問題を修正して再実行するか、*aidb.clear_error_logs* で却下します。
- *not_found* ID がエラーログに存在しない、またはパイプラインにエラーログテーブルがありません。

重複する ID のコンカレント・リキューは、行ごとの FOR UPDATE ロックでシリアル化されます。重複しない ID セットは並列的に処理されます。バックグラウンドパイプラインは、次のワーカーティックに再キューイングされた行を自動的にドレインします。無効またはライブパイプラインには、フォローアップの *aidb.run_pipeline* 呼び出しが必要です。

例

```
SELECT * FROM aidb.requeue_pipeline_errors(my_pipeline, ARRAY[1, 2, 99]);

__OUTPUT__
error_id | action
-----+-----
      1 | requeued
      2 | requeued
     99 | not_found
(3 rows)
```

15.3.2 エラーカテゴリ

Category	Scope	Blocks pipeline?	Description
<i>RecordTemporary</i>	Single record	No	1 つのレコードの一時的な障害。再試行で成功する場合があります。
<i>RecordPermanent</i>	Single record	No	1 つのレコードの永続的な障害たとえば、破損したデータ。
<i>PipelineTemporary</i>	Entire step	Yes	ステップを停止した一時的な障害たとえば、サービスの停止。再試行で解決される場合があります。
<i>PipelinePermanent</i>	Entire step	Yes	ステップを停止した永続的な障害たとえば、無効なモデル構成。

15.4 Functions reference

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

AI 推論機能 `encode_text`、`generate_text`、`rerank_text`、および関連に関しては、[推論ファンクション](#) を参照してください。

15.4.1 `helpb.chunk_text`

テキスト文字列を、より小さい意味的に一貫したセグメントに分割します。

パラメーター

Parameter	Type	Description
<i>input</i>	TEXT	チャンクへのテキスト。
<i>options</i>	JSONB	<code>`aidb.chunk_text_config()`</code> でビルドされたチャンク構成、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。 <code>`max_length`</code> が省略された場合、厳密な上限として機能します。

次のページに続く

表 59 – 前のページからの続き

Key	Type	Default	Description
<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`、 <i>max_length</i> 、および`overlap_length`の単位を決定します。

返品

Column	Type	Description
<i>part_id</i>	bigint	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}
);
```

15.4.2 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	'StructuredPlaintext'	解析方法 'StructuredPlaintext' 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"}
);
```

15.4.3 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' 仕様のテキストブロック抽出。
<i>allow_partial_parsing</i>	boolean	true	'true' の場合、個々のページでエラーが発生したときに解析を継続し、可能な限り多くのデータを結果ます。

返品

Column	Type	Description
<i>part_id</i>	bigint	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}
);
```

15.4.4 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 対応モデルたとえば、 <code>`nim_paddle_ocr`</code> または <code>`llamacpp_ocr`</code> プロバイダーを使用するモデルの名前。

返品

Column	Type	Description
<i>part_id</i>	bigint	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。

次のページに続く

表 68 – 前のページからの続き

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

例

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

15.4.5 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	`generate_text`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 'Summarize for a 5th grader'。
<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。 `aidb.chunk_text` オプションと同じキーを受け入れます。
<i>strategy</i>	text	'append'	要約戦略 'append' 各チャンクを独立して要約および連結する または 'reduce' 目的の長さに達するまで反復要約する。

次のページに続く

表 70 – 前のページからの続き

Key	Type	Default	Description
<i>reduction_factor</i>	integer	3	`reduce` 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"}
);
```

15.4.6 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。
<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、オプションではありません。

返品

TEXT グループの概要。

例

```
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;
```

15.4.7 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>	'{}'	Must include <i>model</i>
<i>summa- rize_text_aggregate</i>	<i>input</i>	— (required)	—
<i>summa- rize_text_aggregate</i>	<i>options</i>	— (required)	—
<i>summarize_text_config</i>	<i>model</i>	— (required)	—
<i>summarize_text_config</i>	<i>chunk_config</i>	<i>NULL</i>	No chunking
<i>summarize_text_config</i>	<i>prompt</i>	<i>NULL</i>	Standard summarize prompt
<i>summarize_text_config</i>	<i>strategy</i>	<i>NULL</i>	'append'
<i>summarize_text_config</i>	<i>reduction_factor</i>	<i>NULL</i>	3
<i>summarize_text_config</i>	<i>inference_config</i>	<i>NULL</i>	Provider defaults
<i>chunk_text_config</i>	<i>desired_length</i>	— (required)	—
<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>	0
<i>chunk_text_config</i>	<i>strategy</i>	<i>NULL</i>	'chars'

15.5 Pipelines reference

完全なパイプライン API リファレンスは、パイプラインドキュメントとともにあります。 [Pipelines API reference](#) を参照してください。

Category	Items
Types and enums	<i>aidb.PipelineAutoProcessingMode</i> , <i>aidb.PipelineDataFormat</i> , <i>aidb.PipelineSourceType</i> , <i>aidb.PipelineDestinationType</i> , <i>aidb.PipelineStepOperation</i> , <i>aidb.PipelineStatus</i> , <i>aidb.DistanceOperator</i>
Domains	<i>aidb.pipeline_name_50</i> , <i>aidb.background_sync_interval</i>
Views	<i>aidb.pipelines</i> , <i>aidb.pipeline_metrics</i>
Functions	<i>aidb.create_pipeline</i> , <i>aidb.update_pipeline</i> , <i>aidb.delete_pipeline</i> , <i>aidb.run_pipeline</i> , <i>aidb.get_pipeline_metrics</i>
Pipeline step config helpers	<i>aidb.chunk_text_config</i> , <i>aidb.summarize_text_config</i> , <i>aidb.ocr_config</i> , <i>aidb.html_parse_config</i> , <i>aidb.pdf_parse_config</i> , <i>aidb.knowledge_base_config</i> , <i>aidb.knowledge_base_config_from_kb</i> , <i>aidb.inference_config</i>
Vector index config helpers	<i>aidb.vector_index_hnsw_config</i> , <i>aidb.vector_index_ivfflat_config</i> , <i>aidb.vector_index_chord_hnsw_config</i> , <i>aidb.vector_index_chord_vchordq_config</i> , <i>aidb.vector_index_hsphere_optimized_config</i> , <i>aidb.vector_index_disabled_config</i>

15.6 Agents reference

エージェント関連の機能、ビュー、およびタイプのリファレンス。ガイドスタイルのドキュメントについては、[Agents reference](#) を参照してください。

15.6.1 カタログビュー

helpb.agents

エージェント構成。専用の「リストエージェント」機能はありません。このテーブルを直接照会します。

Column	Type	Description
<i>id</i>	uuid	内部識別子。
<i>name</i>	text	一意のエージェント名。
<i>instructions</i>	text	エージェントの指示。
<i>model</i>	text	エージェントをバックグランドで実行するモデル。
<i>role</i>	text	エージェントのツール呼び出しの Postgres ロールは、構成されている場合に実行されます。
<i>delegates</i>	text[]	このエージェントが委任できるエージェントの名前。
<i>tools</i>	text[]	このエージェントが呼び出す可能性のあるツールの名前。
<i>output_type</i>	jsonb	<code>aidb.output_type()</code> で構築された構造化出力スキーマ。
<i>input_token_budget</i>	integer	呼び出しごとの入力トークンバジェット。
<i>output_token_budget</i>	integer	呼び出しごとの出力トークンバジェット。
<i>max_iterations</i>	integer	呼び出しごとの最大推論反復数。
<i>timeout_seconds</i>	integer	通話ごとの最大実行時間。

次のページに続く

表 74 – 前のページからの続き

Column	Type	Description
<i>budget_strategy</i>	<i>aidb.budget_strategy</i>	予算を超過した場合の動作。以下の タイプ を参照してください。
<i>preset</i>	text	保存されていますが、現在動作は解決されていません。
<i>created_at</i>	timestampz	作成時間。
<i>updated_at</i>	timestampz	最終更新時間。

promoteb.agent_tasks

`agent_converse` コール「タスク」ごとに 1 行、そのライフサイクルステータスを含みます。

Column	Type	Description
<i>task_id</i>	uuid	内部タスク識別子。
<i>agent_id</i>	uuid	このタスクを実行したエージェント <i>aidb.agents.id</i> 。
<i>conversation_id</i>	uuid	このタスクが属する会話。
<i>request_message_id</i>	uuid	<code>`aidb.conversation_log`</code> のトリガーメッセージの ID。
<i>response_message_id</i>	uuid	完了すると、 <code>`aidb.conversation_log`</code> の結果の回答の ID。
<i>caller_role</i>	text	<code>`agent_converse`</code> を呼び出した Postgres ロール。
<i>status</i>	<i>aidb.task_status</i>	現在のライフサイクルステータス。以下の タイプ を参照してください。
<i>blocking</i>	boolean	今日は常に <code>`true`</code> — すべてのタスクは同期的に実行されます。
<i>error</i>	text	タスクが失敗した場合のエラーメッセージ。
<i>created_at</i>	timestampz	タスクが開始されたとき。
<i>completed_at</i>	timestampz	タスクが完了したとき。既にある場合。

helpb.conversation_log

すべての会話にわたる、すべてのユーザープロンプトとエージェントの回答 - `aidb.get_conversation()` が返すのと同じ行、`conversation_id` によってフィルタリングされていません。ツール呼び出し、モデル要求、およびその他の内部アクティビティは含まれません。完全なアクティビティ履歴については、`aidb_internal.action_log` を直接照会します。

Column	Type	Description
<i>id</i>	uuid	メッセージ ID
<i>conversation_id</i>	uuid	このメッセージが属している会話。
<i>agent_id</i>	uuid	エージェントが関与しました。

次のページに続く

表 76 – 前のページからの続き

Column	Type	Description
<i>task_id</i>	uuid	このメッセージが含まれていたタスク。
<i>action_type</i>	<i>aidb.action_type</i>	<i>user_prompt</i> または <i>answer</i> 。
<i>payload</i>	jsonb	メッセージの生のペイロード <i>user_message</i> または <i>llm_response</i> 構造体。
<i>created_at</i>	timestampz	メッセージが作成されたとき。
<i>completed_at</i>	timestampz	完成したとき。

helpb.conversations

会話ごとに 1 行、*aidb.conversation_log* から集計されます。

Column	Type	Description
<i>conversation_id</i>	uuid	会話の ID。
<i>agent_id</i>	uuid	エージェントが関与しました。
<i>parent_conversation_id</i>	uuid	常に <i>NULL</i> – 会話フォークは実装されていません。
<i>forked_from_message_id</i>	uuid	常に <i>NULL</i> – 会話フォークは実装されていません。
<i>status</i>	<i>aidb.task_status</i>	この会話の最新のタスクのステータス。
<i>message_count</i>	bigint	会話内のメッセージの数。
<i>last_message_id</i>	uuid	最新のメッセージの ID。
<i>created_at</i>	timestampz	会話が始まったとき。
<i>updated_at</i>	timestampz	会話が最後に更新された日。

15.6.2 エージェント管理機能

promoteb.create_agent

新しいエージェントを登録します。

パラメーター

Parameter	Type	Default	Description
<i>name</i>	TEXT	Required	エージェントの一意の名前。
<i>instructions</i>	TEXT	Required	エージェントの動作を定義する指示。

次のページに続く

表 78 – 前のページからの続き

Parameter	Type	Default	Description
<i>model</i>	TEXT	Required	エージェントをバックアップするモデルの名前。 モデルの選択 を参照してください。
<i>tools</i>	TEXT[]	NULL	エージェントが呼び出すことができるツールの名前`aidb.tools`から。
<i>delegates</i>	TEXT[]	NULL	このエージェントが委任できる他のエージェントの名前。
<i>role</i>	TEXT	NULL	エージェントのツールが呼び出す Postgres ロールは実行対象となります。呼び出し者はこのロールのメンバーである必要があります。
<i>output_type</i>	JSONB	NULL	:ref:``aidb.output_type()`` <#aidb-output_type>` で構築された構造化出力スキーマ。
<i>input_token_budget</i>	INTEGER	NULL (no limit)	`agent_converse` コールごとの入力トークンバジェット。
<i>output_token_budget</i>	INTEGER	NULL (no limit)	呼び出しごとの出力トークンバジェット。
<i>max_iterations</i>	INTEGER	NULL	呼び出しごとの最大推論反復数。25 のハードシーリングは、関係なく常に適用されます。
<i>timeout</i>	INTEGER	NULL (300 applied at runtime)	通話ごとの最大実時計秒数。
<i>budget_strategy</i>	TEXT	'attempt_complete'	'ignore'、'error'、'summarize'、'attempt_complete'のいずれか。 :ref: `予算と制限<予算と制限>` を参照してください。
<i>preset</i>	TEXT	NULL	保存されていますが、現在動作は解決されていません。

返品

TABLE(error TEXT) —成功した場合は行なし。失敗に設定された error を含む単一の行検証/名前衝突失敗の場合に発生することはありません。

例

```
SELECT aidb.create_agent(
  name          => db_helper,
  instructions   => You are a helpful assistant that answers questions about_
  ↳ this database.,
  model         => my_gpt,
```

(次のページに続く)

(前のページからの続き)

```

tools          => ARRAY[run_sql_query], -- optional; tool names from aidb.
→tools. Default: NULL (no tools)
delegates      => NULL,                -- optional; names of other
→agents to delegate to. Default: NULL (no delegation)
role           => NULL,                -- optional; Postgres role tool
→calls run as. Default: NULL (run as the calling role)
output_type    => NULL,                -- optional; structured output
→schema from aidb.output_type(). Default: NULL (plain-text answer)
input_token_budget => NULL,            -- optional; input token cap per
→agent_converse call. Default: NULL (no limit)
output_token_budget => NULL,          -- optional; output token cap
→per agent_converse call. Default: NULL (no limit)
max_iterations => NULL,                -- optional; reasoning-round cap
→per call. Default: NULL (a hard ceiling of 25 always applies)
timeout        => NULL,                -- optional; wall-clock seconds
→cap per call. Default: NULL (300 seconds applied at runtime)
budget_strategy => attempt_complete,   -- optional; behavior when a
→budget is exceeded. Default: attempt_complete
preset         => NULL                 -- optional; reserved for future
→use, has no effect today. Default: NULL
);

```

promoteb.update_agent

既存のエージェントを更新します。name を除くすべてのパラメーターはオプションであり、デフォルトは「変更しない」を意味する NULL になります。budget_strategy を含みます。ここでのデフォルトは 'attempt_complete' ではなく NULL です。

パラメーター

create_agent と同じパラメーター、name を除き、すべてオプションデフォルト NULL。

返品

TABLE(error TEXT) —成功した場合は行なし。失敗した場合に設定された error を含む単一行エージェントが見つからない、無効なフィールド値。

例

```
SELECT aidb.update_agent(db_helper, model => my_new_gpt, max_iterations => 15);
```

promoteb.delete_agent

名前でエージェントを削除します。

パラメーター

Parameter	Type	Default	Description
<i>name</i>	TEXT	Required	削除するエージェントの名前。
<i>force</i>	BOOLEAN	NULL (false)	`true`の場合、エージェントの内部タスク/アクションキューレコードも削除するため、会話履歴がある場合でも削除を続行できます。会話トランスクリプト自分自身`aidb.conversation_log`は、いずれの方法でも保存されます。

返品

TABLE(error TEXT) —成功した場合は行なし。エージェントに会話履歴があり、**force** が渡されなかった場合を含む、失敗に設定された **error** を含む単一の行。

例

```
SELECT aidb.delete_agent(db_helper, force => true);
```

15.6.3 会話機能**promoteb.agent_converse**

プロンプトに対してエージェントの推論ループを実行し、その回答を返します。

パラメーター

Parameter	Type	Default	Description
<i>agent_name</i>	TEXT	Required	会話するエージェントの名前。
<i>prompt</i>	TEXT	Required	送信するプロンプト。
<i>conversation_id</i>	TEXT	NULL	既存の会話を続行します。新しいものを開始する場合は省略します。ID は自動的に生成されます。

[次のページに続く](#)

表 80 – 前のページからの続き

Parameter	Type	Default	Description
<i>read_only</i>	BOOLEAN	NULL	読み取り専用モード を強制的にオンまたはオフにします。省略すると、リードレプリカで自動有効化されます。
<i>output_type</i>	JSONB	NULL	このコールのみにエージェントの設定済み 構造化された出力 スキーマをオーバーライドします。
<i>debug</i>	BOOLEAN	NULL	<i>true</i> の場合、すべてのアクションを <i>NOTICE</i> としても出力します。:ref: デバッグモード<デバッグモード> を参照してください。

返品

TABLE(message TEXT, conversation_id TEXT, error TEXT) – 1 行。エージェントロジックの障害に対してレイズすることはありません。代わりに error が設定され、message / conversation_id は NULL です。ID を添付するものは何も永続化されないため、conversation_id は読み取り専用実行の NULL でもあります。

例

```
SELECT * FROM aidb.agent_converse(
  agent_name      => db_helper,
  prompt          => How many rows are in the orders table?,
  conversation_id => NULL, -- optional; continue an existing conversation by id.
  ↳Default: NULL (starts a new conversation)
  read_only       => NULL, -- optional; force read-only mode on/off. Default: NULL
  ↳(auto: on for a read replica, off otherwise)
  output_type     => NULL, -- optional; override the agents structured output
  ↳schema for this call. Default: NULL (use the agents own configuration)
  debug          => NULL -- optional; also emit every action as a NOTICE.
  ↳Default: NULL (false)
);
```

promoteb.start_agent_session

名前付けエージェントの新しい会話 ID を使用する最初の agent_converse 呼び出しに先立って、名前付けエージェントの新しい会話 ID を生成します。

パラメーター

Parameter	Type	Description
<i>agent_name</i>	TEXT	会話するエージェントの名前。

返品

TABLE(conversation_id TEXT) . agent_converse とは異なり、このファンクションは、agent_name が空または不明な場合に **error** 列を返すのではなく **レイズ** します。

例

```
SELECT * FROM aidb.start_agent_session(db_helper);
```

promoteb.get_conversation

会話のメッセージ-すべてのユーザーのプロンプトと最終的な回答を時系列で取得します。ツール呼び出しおよびその他の内部アクティビティを除外します。完全なアクティビティ履歴については、aidb_internal.action_log を参照してください。

パラメーター

Parameter	Type	Description
<i>conversation_id</i>	TEXT	会話の ID。

返品

TABLE(message_id TEXT, task_id TEXT, action_type TEXT, role TEXT, sender_id TEXT, contents TEXT) — メッセージごとに 1 行。role は"user" または"agent" です。sender_id は、特定の送信者プロンプト自分自身の user フィールド、またはエージェントの ID です。conversation_id が空の場合、**レイズ** error 列を返すのではなく。

例

```
SELECT * FROM aidb.get_conversation(8f14e45f-ceea-467e-9575-a3d1a2c1a123);
```

promoteb.get_message

ID に基づいて 1 つのメッセージのプレーンテキストコンテンツを取得します。

パラメーター

Parameter	Type	Description
<i>message_id</i>	TEXT	メッセージの ID。

返品

TABLE(message TEXT, error TEXT) — 1 行。error message ではないは、そのようなメッセージが存在しない場合に設定されます。上げません。

例

```
SELECT * FROM aidb.get_message(3fa85f64-5717-4562-b3fc-2c963f66afa6);
```

15.6.4 構造化出力ヘルパー

helpb.output_field

エージェントの構造化出力スキーマの 1 つのフィールドを構築します。

パラメーター

Parameter	Type	Default	Description
<i>name</i>	TEXT	Required	フィールドの名前。
<i>field_type</i>	TEXT	Required	フィールドのタイプたとえば 'TEXT'、'FLOAT'、'BOOLEAN' など。
<i>description</i>	TEXT	NULL	モデルに表示される説明。

返品

JSONB — これらのうちの 1 つ以上を aidb.output_type() に渡します。

promoteb.output_type

1 つ以上のフィールドから完全な構造化出力スキーマを構築します。

パラメーター

Parameter	Type	Description
<i>fields</i>	VARIADIC JSONB[]	1 つ以上の `aidb.output_field()` 結果。

返品

JSONB — `create_agent` / `update_agent` の `output_type` パラメーターまたは `agent_converse` に渡します。

例

```
SELECT aidb.create_agent(
  name          => sentiment_tagger,
  instructions => Classify the sentiment of the given text.,
  model         => my_gpt,
  output_type  => aidb.output_type(
    aidb.output_field(sentiment, TEXT, One of: positive, negative, neutral),
    aidb.output_field(confidence, FLOAT)
  )
);
```

15.6.5 タイプ

helpb.budget_strategy

```
CREATE TYPE aidb.budget_strategy AS ENUM (
  ignore,
  error,
  summarize,
  attempt_complete
);
```

各値の機能については、[予算と制限](#) を参照してください。

promoteb.task_status

```
CREATE TYPE aidb.task_status AS ENUM (
  PENDING,
  IN_PROGRESS,
  RECOVERY,
  AWAIT_APPROVAL,
  DENIED,
  ERROR,
  TIMEOUT,
```

(次のページに続く)

(前のページからの続き)

```
    CANCELED,
    SUCCESS,
    EVALUATING,
    COMPLETE
);
```

`aidb.agent_tasks.status` および `aidb.conversations.status` を介して浮上しました。今日のすべてのタスクは 1 つの `agent_converse` 呼び出し内で完了まで同期して実行されるため、タスクのターミナルステータスは通常 `SUCCESS` または `ERROR` になります。残りの値は、まだ実装されていない非同期/承認ワークフロー用に予約されています。

15.7 Tools reference

ツール関連のファンクション、ビュー、およびタイプのリファレンス。ガイドスタイルのドキュメントについては、[Tools reference](#) を参照してください。完全なネイティブツールカタログについては、[Native tools](#) を参照してください。

15.7.1 カタログビュー

helpb.tools

AIDB が認識するすべてのツールの統合された読み取り専用カタログ-ネイティブツール、登録された SQL ツール、およびキャッシュされた MCP ツールのユニオン。

Column	Type	Description
<i>name</i>	text	ツールの名前。
<i>description</i>	text	ツール定義の一部としてモデルに表示されます。
<i>tool_type</i>	text	'native_tool'、'sql_tool'、または 'mcp_tool'
<i>params</i>	jsonb	<i>native_tool/sql_tool</i> の場合 {name, type, description} オブジェクトの配列。 <i>mcp_tool</i> の場合 サーバーの生の JSON スキーマ <i>inputSchema</i> 。
<i>read_only</i>	boolean	ツールが決して書き込めないことが知られているかどうか。 <i>mcp_tool</i> 行の場合は常に <i>false</i> — 読み取り専用モード を参照してください。

promoteb.sql_tool_registry

バック登録された SQL ツール。 `aidb.tools` は、ほとんどのクエリーが代わりに使用するユニオンビューです。これを直接照会して、SQL ツールに保存されているクエリテキストを表示します。

Column	Type	Description
<i>name</i>	text	一意のツール名主キー。

次のページに続く

表 87 – 前のページからの続き

Column	Type	Description
<i>description</i>	text	モデルに見せられました。
<i>query_text</i>	text	`\${name}`プレースホルダーを使用した保存された SQL ステートメント。
<i>read_only</i>	boolean	ツールが読み取り専用として適用されるかどうか。
<i>return_type_hint</i>	<i>aidb.ToolParam[]</i>	文書化された返品の形状。現在モデルには表面化されていません。
<i>params</i>	<i>aidb.ToolParam[]</i>	ツールの宣言されたパラメーター。
<i>created_at</i>	timestamp	作成時間。
<i>updated_at</i>	timestamp	最終更新時間。

promoteb.mcp_registry

登録された外部 MCP サーバー。

Column	Type	Description
<i>name</i>	text	一意のサーバー名主キー。
<i>url</i>	text	サーバーの MCP エンドポイント。
<i>transport</i>	text	'streamable_http'または'sse'。
<i>headers</i>	jsonb	保存された要求ヘッダー。`aidb_users`では読み取れません シークレットを保持する可能性があります。
<i>headers_env</i>	text	代わりにヘッダーを読み取る環境変数名。`headers`と相互排他的。
<i>tool_filter</i>	text[]	フィルタリングされた場合、このサーバーからインポートされたツール名。
<i>created_at</i>	timestamp	登録時間。
<i>updated_at</i>	timestamp	最終更新時間。

15.7.2 ツール管理機能

promoteb.create_sql_tool

パラメーター化された SQL クエリーをツールとして登録します。

パラメーター

Parameter	Type	Default	Description
<i>name</i>	TEXT	Required	ツールの一意の名前。
<i>description</i>	TEXT	Required	ツール定義の一部としてモデルに表示されます。
<i>sql_statement</i>	TEXT	Required	実行するクエリー。宣言されたパラメーターごとに`\${name}`プレースホルダーを使用します。
<i>params</i>	JSONB	Required	`aidb.tool_params()`で構築されたツールのパラメーター。
<i>read_only</i>	BOOLEAN	<i>true</i>	`true`の場合、読み取り専用であることが証明されていない限り、登録時にステートメントを拒否します。
<i>return_type_hint</i>	JSONB	NULL	クエリの結果の形状を文書化します。現在モデルには表面化されていません。

返品

TEXT — ツールの名前、成功時。名前が既に使用されている場合、またはステートメントが単一の呼び出し可能、読み取り専用に準拠した `if read_only => true` ステートメントでない場合に発生します。

例

```
SELECT aidb.create_sql_tool(  
  name           => orders_by_customer,  
  description     => Look up recent orders for a customer by id.,  
  sql_statement   => SELECT id, status, total FROM orders WHERE customer_id =  
→ ${customer_id} ORDER BY created_at DESC LIMIT ${limit},  
  params          => aidb.tool_params(  
    aidb.tool_param(customer_id, INT, The customers id.),  
    aidb.tool_param(limit, INT, Maximum number of orders to return.)  
  ),  
  read_only       => true, -- optional; enforced, not just descriptive. Default:  
→ true  
  return_type_hint => NULL -- optional; documents the returned row shape, same  
→ aidb.tool_params() shape as `params`. Default: NULL  
);
```

promoteb.delete_tool

SQL ツールまたは MCP サーバー登録を名前で削除しますサーバーと一緒にアドバタイズしたすべてのツールを削除します。ビルトインネイティブツールは削除できません。

パラメーター

Parameter	Type	Description
<i>name</i>	TEXT	ツールまたはサーバーの名前。

返品

TEXT —成功した場合の削除された名前。そのようなツールまたはサーバーが存在しない場合、または **name** がネイティブツールの場合に発生します。

promoteb.import_mcp_tools

外部 MCP サーバーを登録し、アドバタイズするツールをインポートします。

パラメーター

Parameter	Type	Default	Description
<i>name</i>	TEXT	Required	サーバー登録の一意の名前。
<i>url</i>	TEXT	Required	サーバーの MCP エンドポイント URL。
<i>transport</i>	TEXT	'streamable_http'	'streamable_http' または 'sse' —現在 'streamable_http' のみが完全にサポートされています。
<i>headers</i>	JSONB	NULL	送信する要求ヘッダー。`headers_env` と相互排他的。
<i>tool_filter</i>	TEXT[]	NULL	これらのツール名のみをインポートします。サーバーがアドバタイズするすべてのインポートを省略します。
<i>headers_env</i>	TEXT	NULL	ヘッダー JSON を保持する環境変数。保存するのではなく、使用するたびに新しく読み取られます。`headers` と相互排他的。名前は <code>:ref: AI pipelines <Configuring AIDB></code> 接頭辞で始まる必要があります。

返品

TEXT —成功した場合のサーバーの名前。名前が使用されるか、URL に到達不能であるか、`tool_filter` と一致する宣伝ツールがない場合に発生します。

例

```
SELECT aidb.import_mcp_tools(
  name      => weather,
  url       => https://weather.example.com/mcp,
  transport => streamable_http, -- optional; streamable_http or sse. Default:
→streamable_http
  headers   => NULL,           -- optional; request headers to send, for
→example, an auth token. Mutually exclusive with headers_env. Default: NULL
  tool_filter => NULL,         -- optional; import only these tool names.
→Default: NULL (import everything the server advertises)
  headers_env => NULL          -- optional; env var holding the headers JSON
→instead of storing it. Mutually exclusive with headers. Default: NULL
);
```

promoteb.refresh_mcp_tools

登録済みの MCP サーバーのアダプタイズツールを再取得し、キャッシュを更新します。

パラメーター

Parameter	Type	Description
<i>name</i>	TEXT	登録されたサーバーの名前。

返品

INTEGER —現在キャッシュされているツールの数 `tool_filter` 適用後

promoteb.get_mcp_tools

現在登録されているすべてのツールネイティブ、SQL、および MCP を同様に MCP tools/list 型の記述子に変換します。

返品

TABLE(name TEXT, description TEXT, input_schema JSONB) —aidb.tools のツールごとに 1 行。

15.7.3 パラメーターヘルパー

promoteb.tool_param

1 つの SQL ツールパラメーターを構築します。

パラメーター

Parameter	Type	Default	Description
<i>name</i>	TEXT	Required	パラメーターの名前。
<i>type</i>	TEXT	Required	パラメーターの SQL タイプ。
<i>description</i>	TEXT	Required	モデルに表示される説明。

返品

JSONB — これらのうちの 1 つ以上を `aidb.tool_params()` に渡します。

promoteb.tool_params

1 つ以上の `aidb.tool_param()` 結果を組み合わせるか、引数なしで呼び出されるは空のパラメーターリストを宣言します。

パラメーター

Parameter	Type	Description
<i>params</i>	VARIADIC JSONB[]	0 個以上の <code>`aidb.tool_param()`</code> 結果。

返品

JSONB — `aidb.create_sql_tool()` の `params` または `return_type_hint` に渡します。

promoteb.param / promoteb.params

汎用のパラメーター仕様コンストラクター。列挙コンストレインの引数を受け入れるネイティブツールによって使用されます。`aidb.param()` はさらに `enum_values TEXT[]` を受け入れます。通常は直接必要ありません。`aidb.tool_param()` / `aidb.tool_params()` は、`create_sql_tool()` のコンストラクターです。