



Analytics Lakehouse

リリース Q3 2025

EnterpriseDB Corporation

2025 年 09 月 25 日

目次

第 1 章 Introduction	3
1.1 分析アクセラレータの操作	3
1.2 概念的基礎	3
1.3 EDB コア分析テクノロジー	3
1.4 実践的なガイダンスと解決策	4
1.5 製品固有の実装	4
1.6 どこから始めるか	4
1.6.1 レベル 100	5
1.6.2 レベル 200	5
1.6.3 レベル 300	6
第 2 章 Quick start	7
2.1 導入	7
2.1.1 知っておくべきこと	7
2.1.2 基本的なアーキテクチャ	7
2.2 はじめる	8
2.2.1 Lakehouse ノードを作成する	8
2.3 Lakehouse ノードの操作	10
2.3.1 ノードに接続する	10
2.3.2 制約を理解する	10
2.4 ベンチマークデータセットを検査する	11
2.5 ベンチマークデータセットをクエリする	11
第 3 章 Tiered Tables	13
3.1 階層型テーブルとは	13
3.2 EDB 分析において階層型テーブルが重要な理由	13
3.3 EDB が階層型テーブルを実装する方法	14
3.4 一般的な使用例	14
3.5 役割ベースのガイダンス	15
3.6 学習パス	15
3.7 関連概念	15
3.8 次のステップ	15
第 4 章 Reference - EDB Postgres® AI Lakehouse	17
4.1 Benchmarking datasets	17
4.2 Queries	18
4.3 Delta Lake Table tools	18
4.3.1 Delta Lake テーブルの作成	18
4.4 DirectScan and fallback modes	19
4.4.1 ダイレクトスキャン	19

4.4.2	EXPLAIN を使用してクエリプランを確認する	19
4.5	User management	20
4.6	Loading data	21
4.6.1	独自のデータを持ち込む	21
4.7	PGAA functions reference	21
4.7.1	概要	21
4.7.2	パラメータ	21
4.7.3	戻り値	21
4.7.4	概要	22
4.7.5	パラメータ	22
4.7.6	戻り値	22
4.8	PGAA functions reference	22
4.8.1	pgaa.create_storage_location	22
4.9	Supported AWS instances	23
第 5 章	Delta Lake	25
5.1	Delta Lake とは	25
5.2	Delta Lake が EDB 分析にとって重要な理由	25
5.3	EDB が Delta Lake を活用する方法	26
5.4	一般的な使用例	26
5.5	役割ベースのガイダンス	26
5.6	学習パス	26
5.7	関連概念	27
5.8	次のステップ	27
第 6 章	Apache Iceberg	29
6.1	Apache Iceberg とは	29
6.1.1	主な目標	29
6.2	主な機能と利点	29
6.3	EDB 分析において Apache Iceberg が重要な理由	30
6.4	EDB ソリューションが Iceberg サポートを実装する方法	30
6.4.1	PGAA 拡張	30
6.4.2	カタログ接続	30
6.4.3	Iceberg テーブルへの書き込み	31
6.5	関連トピック	31
第 7 章	EDB Postgres Lakehouse	33
7.1	EDB Postgres Lakehouse とは	33
7.2	Lakehouse が EDB 分析にとって重要な理由	33
7.3	EDB がレイクハウスアーキテクチャを実装する方法	33
7.4	一般的な使用例	34
7.5	役割ベースのガイダンス	34
7.6	学習パス	35
7.7	関連概念	35
7.8	次のステップ	35
第 8 章	Analytics Accelerator learning resources	37
8.1	ドキュメントのカテゴリ	37

8.1.1	構造化された学習パス	37
8.1.2	概念の説明	37
8.1.3	チュートリアル	37
8.1.4	ハウツーガイド	38
8.1.5	ユースケースとソリューション	38
8.1.6	ペルソナベースのガイダンス	38
8.1.7	ハイブリッド マネージャー (HM) の分析アクセラレータ	38
第 9 章	Analytics Accelerator learning resources	39
9.1	Analytics Accelerator concepts and technologies	39
9.2	Analytics Accelerator concepts and technologies	39
9.2.1	基礎的な分析と最新のデータアーキテクチャ	39
9.3	Analytics Accelerator concepts and technologies	40
9.3.1	Analytics Accelerator のコアコンポーネントの詳細	40
9.4	Analytics Accelerator concepts and technologies	40
9.4.1	EDB ハイブリッドマネージャー (HM) 内の分析を理解する	40
9.4.2	AI/ML ワークロードと相互運用性	41
9.5	Analytics Accelerator learning paths	53
9.5.1	これらの学習パスの使い方	53
9.6	Analytics Accelerator learning paths	53
9.7	Analytics Accelerator learning paths	53
9.7.1	EDB のトレーニングと認定	53
9.8	Analytics Accelerator learning paths	54
9.8.1	次のステップ	54
9.9	a persona-based guide	56
9.9.1	役割を素早く見つける	56
9.9.2	データベース管理者 (DBA)	57
9.9.3	DevOps エンジニア / サイト信頼性エンジニア (SRE)	58
9.9.4	データサイエンティスト/データアナリスト	58
9.9.5	アプリケーション開発者	59
9.10	Analytics Accelerator tutorials	60
9.10.1	チュートリアルのカテゴリ	60

Analytics Accelerator を使用して、EDB Postgres®上に構築された分析機能を探索してください。このアクセラレータは、コアコンセプトの理解、EDB Postgres® Lakehouse などの主要テクノロジーの探索、EDB Hybrid Manager (HM) を使用した分析の実装方法の習得に役立ちます。

最新のデータ アーキテクチャとオープン スタンダードを Postgres の信頼性と柔軟性と統合し、貴重な洞察を引き出すお手伝いをします。

第1章 Introduction

1.1 分析アクセラレータの操作

アクセラレータはコンテンツを次の4つの領域に整理します。

- **概念の基礎** 分析の原則と EDB のアプローチについての理解を深めます。
- **EDB コア分析テクノロジー** 当社の分析サービスを強化する EDB ソリューションとテクノロジーについて学習します。
- **実用的なガイダンスとソリューションユースケース**、ペルソナベースのガイド、ハウツー記事、チュートリアルを見つけます。
- **製品固有の実装** EDB Hybrid Manager などの EDB 製品でこれらの分析機能がどのように表示され、管理されるかについてのドキュメントにアクセスします。

1.2 概念的基礎

最新のデータ分析と EDB のアプローチの背後にある原則と戦略を理解します。

- *Generic analytics concepts*

データ アーキテクチャ (データ ウェアハウス、データ レイク、レイクハウス) と基礎テクノロジー (列指向ストレージ、ベクトル化エンジンなど) について学習します。

- *EDB analytics concepts*

EDB の Postgres[®] 分析に関するビジョンと、EDB がコアテクノロジーをどのように活用しているかをご覧ください。

- Explained: Analytics

EDB の分析機能、設計上の選択肢、高度なトピックに関する詳細な説明を確認します。(近日公開)

1.3 EDB コア分析テクノロジー

EDB の分析テクノロジーとそれが Postgres[®] をどのように拡張するかについて学びます。

- *EDB Postgres Lakehouse*

オブジェクト ストレージの分析を可能にするための EDB Postgres[®] Lakehouse ソリューションとそのコンポーネントを確認します。

- *Apache Iceberg*

EDB ソリューションが Apache Iceberg を使用して大規模な分析データセットを管理する方法について理解します。

- [Delta Lake](#)

EDB Postgres[®] が Delta テーブルと連携して信頼性の高いデータレイクを実現する方法について学習します。

- [EDB Postgres 分散 \(PGD\) と階層化テーブル](#)

EDB Postgres Distributed (PGD) と Lakehouse 機能を使用してストレージ層全体でデータを管理し、コストとパフォーマンスを最適化します。

1.4 実践的なガイダンスと解決策

ニーズに合わせて EDB の分析機能を適用します。

- [Analytics Accelerator for your role: a persona-based guide](#)

DBA、DevOps エンジニア、データサイエンティスト、アプリケーション開発者向けの学習パスに従ってください。

1.5 製品固有の実装

EDB 分析の概念とテクノロジーが EDB 製品にどのように実装されているかを確認します。

- [Introduction](#)

EDB ハイブリッドマネージャーの分析機能に関するドキュメントをご覧ください。これには、HM Lakehouse クラスター、HM での Iceberg、Delta、階層化テーブルの使用、HM 固有のチュートリアルが含まれます。

1.6 どこから始めるか

- コアとなる考え方を理解するには、[Generic analytics concepts](#) と [EDB Postgres Lakehouse](#) から始めてください。
- 可能な場合は実用的なガイダンスを調べます。
- EDB Hybrid Manager を使用する場合は、製品固有のドキュメントを使用します。

Postgres Lakehouse は、さまざまなテクノロジーを使用して構築されています。

- PostgreSQL
- [Seafowl](#)、分析データベース
- [Apache DataFusion](#)、Seafowl で使用されるクエリ エンジン
- [Delta Lake](#) (特に [delta-rs](#)) は、デルタテーブルの保存および取得レイヤーを実装します。

1.6.1 レベル 100

Postgres Lakehouse について理解すべき最も重要な点は、ストレージとコンピューティングが分離されていることです。この設計により、ストレージとコンピューティングを独立してスケーリングできるため、クエリが予測不可能でスパイクが発生する分析ワークロードに最適です。接続されたハードドライブにデータを保持するためだけに、マシンをほぼアイドル状態にしておくのは望ましくありません。その代わりに、データをオブジェクトストレージ（しかも高度に圧縮可能な形式）に保存し、必要な時にクエリを実行するために必要なコンピューティングリソースのみをプロビジョニングできます。

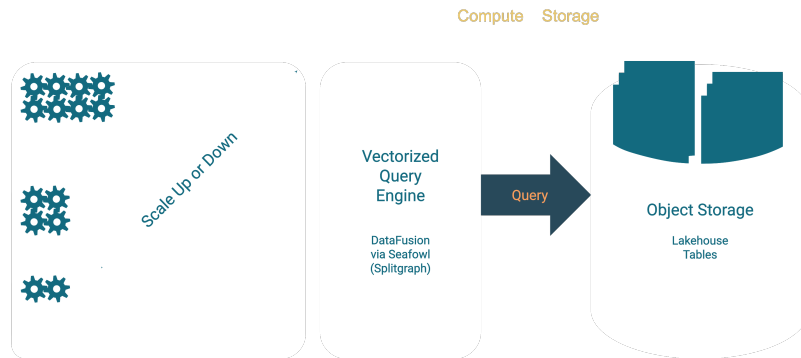


図 1: Level 100 Architecture

コンピューティング側では、ベクトル化されたクエリ エンジンが Lakehouse テーブルをクエリするように最適化されていますが、完全な互換性のために Postgres にフォールバックします。

ストレージ側では、Lakehouse テーブルは分析用に最適化された、高度に圧縮可能な列指向ストレージ形式を使用して保存されます。

1.6.2 レベル 200

これらのサービスがどのように組み合わせられているかを示す、もう少し包括的な図を以下に示します。

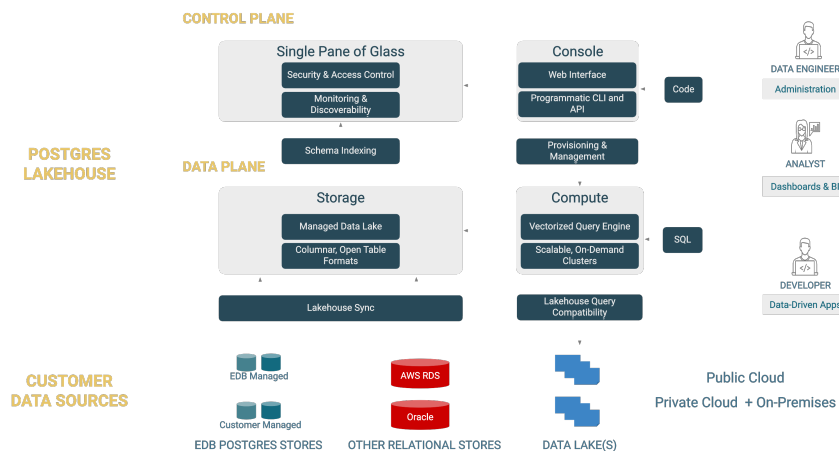


図 2: Level 200 Architecture

1.6.3 レベル 300

以下は、「箱の中身」のより詳細な拡大図です。

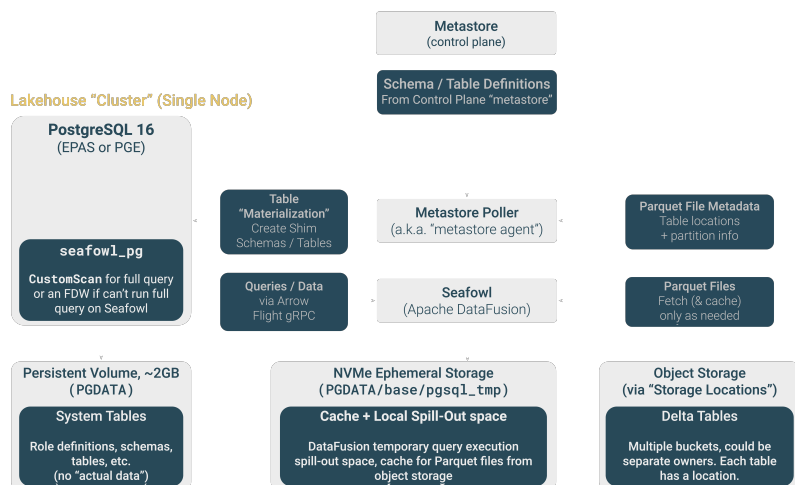


図 3: Level 300 Architecture

第2章 Quick start

このクイック スタートでは、次のことを行います。

1. Lakehouse ノードを作成します。
2. 好みの Postgres クライアントを使用してノードに接続します。
3. オブジェクト ストレージ内のサンプル データ (TPC-H、TPC-DS、Clickbench、または 1BRC) をクエリします。

詳細と高度な使用例については、[PGAA functions reference](#) を参照してください。

2.1 導入

Postgres Lakehouse は、EDB Postgres[®] AI Platform 上でプロビジョニングできる新しいタイプの Postgres クラスタ（実際には 1 ノードのみ）です。Apache DataFusion をベースにしたベクトル化クエリエンジンを搭載しており、オブジェクトストレージ内の Lakehouse 列指向テーブル（[Delta Lake protocol](#) を使用）に対する高速クエリを実現します。

2.1.1 知っておくべきこと

詳細については、[関連概念](#) および [PGAA functions reference](#) を参照してください。ただし、知っておくべき主なポイントは次のとおりです。

- Postgres Lakehouse ノードは一時的です。
- Postgres Lakehouse ノードは、オブジェクト ストレージ内のベンチマークデータセットをクエリするように事前構成されています。
- Postgres Lakehouse は通常は読み取り専用ですが、それ以外はすべての有効な Postgres クエリをサポートします。
- システムテーブルは書き込み可能なので、ユーザーとロールを作成できます。これらのテーブルはアタッチされたブロックストレージに保存されるため、インスタンスの再起動後も保持され、バックアップ/リストア操作の一部として保存されます。Lakehouse テーブルは最終的にオブジェクトストレージに保存されるため、バックアップには含まれません。

2.1.2 基本的なアーキテクチャ

次の図は、Lakehouse ノードのボックスの内容を示しています。

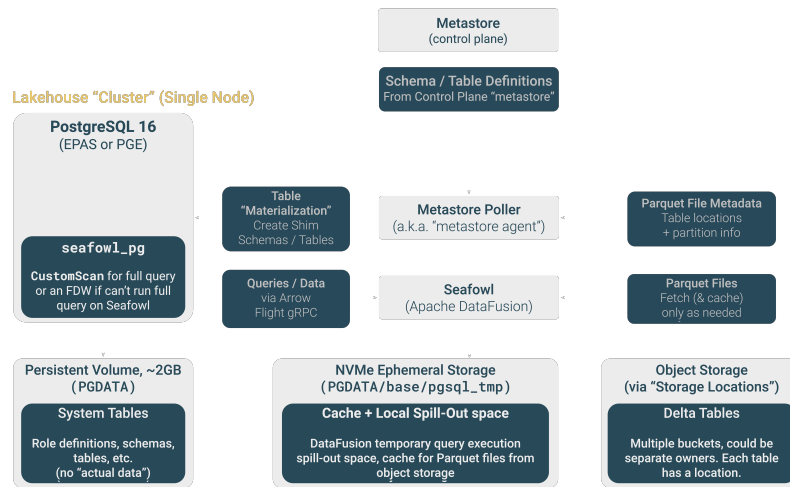


図 1: Level 300 Architecture

2.2 はじめる

HCP ポータルにログインしてプロジェクトを作成したら、クラスターを作成できます。

2.2.1 Lakehouse ノードを作成する

プロジェクト ページで、**[新規作成]** > **[Lakehouse Analytics]** を選択します。

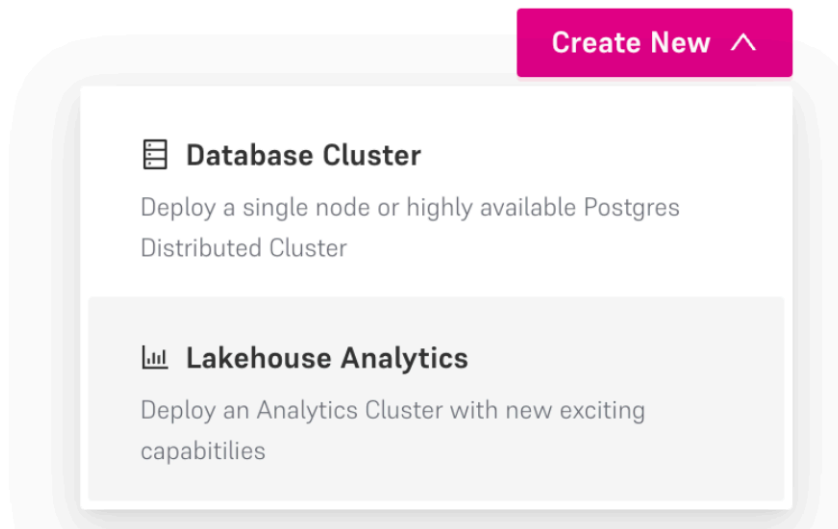


図 2: Create Lakehouse Node Dropdown

分析クラスターの作成 ページが開き、テンプレート ビルドまたはカスタムビルドを選択できます。

カスタムビルド を選択し、**最初から開始** を選択します。**分析クラスターの作成** ページが開き、クラスターのサイズと構成を選択できます。

← Back **Create Analytical Cluster** [Support](#) | [EDB Postgres® AI Docs](#)

[Choose Path](#)

Choose Your Path

Use a template or build from scratch.



No Templates Available, Yet...

No templates are available, but you can create your own! Start building a new template below.

Create New Template



Custom Build

Customize every aspect of your cluster, from primary settings to backups. Best for advanced users needing flexibility.

Start from Scratch

図 3: Create Lakehouse Node Wizard Step 1

← Back **Create Analytical Cluster** [Support](#) | [EDB Postgres® AI Docs](#)

[Cluster Settings](#) [Additional Settings](#)

Cluster Name

Enter a name for your cluster

Password

Passwords must be at least 12 characters.

Enter a password

We do not store your password and cannot retrieve it. If you forget your password, you can simply change it on the "Edit Cluster" page or from Clusters Action menu.

Tags

[Add Tag](#)

Deployment Location

Location

managed-devspatcher

Database Type

Please reference [this page](#) for more information on database types.

Postgres Type

PostgreSQL EDB Postgres Extended Server EDB Postgres Advanced Server

Postgres Version **Software Image**

Select... Select...

16 is the minimum supported version for Lakehouse clusters

Instance Size

CPU (0.1-15 Cores) Memory (0.1-10 Gi)

Cores Gi

Cluster Summary

Cluster Type	Single Node
Cluster Nodes	1 node
Cluster Name	--
Tags	--
Deployment Location	managed-devspatcher
Postgres Type	--
Postgres Version	--
CPU Cores	--
Memory	--
Networking	Private
Backups Retention	30 Days
Scheduled Backup Time	21:49

図 4: Create Lakehouse Node Wizard Step 2

その後、**クラスターページ**に戻り、作成されたクラスターのステータスを確認できます。クラスターのプロビジョニングには 10～15 分かかる場合があります。

いくつかの注意事項:

- パスワードはご自身でご入力ください。安全なパスワードを作成し、パスワードマネージャーに保存してください。HCP はパスワードを保存しません。
- EDB Postgres Advanced Server または EDB Postgres Extended Server を選択できます。
- クラスターのシャットダウン時に、NVMe ドライブ上のすべてのデータが失われます。
- システム テーブル内の永続データ (ユーザー、ロールなど) は、接続されたブロック ストレージデバイスに保存され、再起動やバックアップ/復元サイクルを経ても保持されます。
- Postgres 16 のみがサポートされています。

2.3 Lakehouse ノードの操作

2.3.1 ノードに接続する

EDB Postgres AI Platform から他のクラスターに接続するのと同じ方法で、任意の Postgres クライアントから Lakehouse ノードに接続できます。クラスターの詳細ページに移動し、接続文字列をコピーしてください。

例えば、`.pgpass` の BLOB を `~/.pgpass` にコピーします (`$YOUR_PASSWORD` はクラスターの起動時に指定したパスワードに置き換えてください)。その後、接続文字列をコピーして、`psql` または `pgcli` の引数として使用できます。

一般的に、どの Postgres クライアントでもデータベースに接続できます。すべてのイントロスペクションクエリが機能することを期待しており、機能しないクエリが見つかった場合はバグです。

2.3.2 制約を理解する

- すべてのクラスターは EDB Postgres Advanced Server または EDB Postgres Extended Server を使用しています。そのため、接続時にこれらのフレーバーの定型テーブルがインストール内に表示されることをご承知おきください。
- クエリ可能なデータ (ベンチマークデータセットなど) は、オブジェクトストレージにデルタテーブルとして保存されます。すべてのクラスターは、スケールファクター 1～1000 のベンチマークデータ (TPC-H、TPC-DS、Clickbench) を含むストレージバケットを参照するように事前にロードされています。
- 現在、AWS のみがサポートされています。BYOA (Bring Your Own Account) はサポートされていません。
- EDB Postgres AI アカウントで有効化されている任意のリージョンにクラスターをデプロイできます。各リージョンには、ベンチマークデータのコピーが保存されたバケットがあります。そのため、クラスターを起動すると、最も近いロケーションのベンチマークデータが使用されます。
- クラスターは一時的なものです。システムテーブル (ロール、ユーザー、権限など) のデータを除き、ハードドライブにはデータは保存されません。クラスターを再起動するか、クラスターをバックアップ

ブしてから復元すると、これらのシステムテーブルが復元されます。ただし、オブジェクトストレージ内のデータはそのまま残ります。

- クラスターはオブジェクトストレージ内のデータに対する読み取り専用クエリをサポートしていますが、ユーザー作成などのためのシステムテーブルへの書き込みクエリはサポートしていません。オブジェクトストレージに直接書き込むことはできません。また、テーブルを作成することもできません。
- 独自のデータをオブジェクト ストレージにロードする場合は、[Reference - Bring your own data](#) を参照してください。

2.4 ベンチマークデータセットを検査する

ベンチマークデータセットを調べてください。すべてのクラスターには、すぐに使用できるベンチマークデータが用意されています。pgcli を使用している場合は、`\dn` を実行して利用可能なテーブルを確認できます。

利用可能なベンチマーク データセットは次のとおりです。

- TPC-H、スケール係数 1、10、100、1000
- TPC-DS、スケール係数 1、10、100、1000
- ClickBench、1 億行
- 10 億行チャレンジ

ベンチマーク データセットの詳細については、[Reference - Available benchmarking datasets](#) を参照してください。

2.5 ベンチマークデータセットをクエリする

いくつかの基本的なクエリを実行してみてください。

```
edb_admin> select count(*) from clickbench.hits;
+-----+
| count |
+-----+
| 99997497 |
+-----+
SELECT 1
Time: 0.945s

edb_admin> select count(*) from brc_1b.measurements;
+-----+
| count |
+-----+
| 1000000000 |
+-----+
```

(次のページに続く)

(前のページからの続き)

```
SELECT 1  
Time: 0.651s
```

!!! 注記

詳細な注意事項と構文/クエリの互換性に関する情報については、[PGAA functions reference](#) を参照してください。!!!

第3章 Tiered Tables

階層化テーブルにより、EDB Postgres[®]は、「コールド」データをコスト効率の高いオブジェクトストレージに自動的に移動し、「ホット」データをプライマリランザクションストレージに保持することで、大規模な時間ベースのデータセットを効率的に管理できるようになります。

このパターンは、データセット全体への完全な分析アクセスを維持しながら、パフォーマンスとコストの両方を最適化します。

ハイブリッド マネージャー (HM) 内で階層化テーブルを実装および管理する方法の詳細については、[Tiered Tables](#) を参照してください。

3.1 階層型テーブルとは

階層化テーブルは、Analytics Accelerator アーキテクチャによってサポートされている EDB Postgres Distributed (PGD) のネイティブ機能です。

次のものを使用して、時間パーティション分割されたテーブルの古いパーティションを PGD からオブジェクト ストレージ (Apache Iceberg 形式) に自動的にオフロードします。

- PGD AutoPartition による自動パーティショニングとライフサイクル制御
- オフロードされたデータのクエリとアクセスのための PGAA と PGFS
- ガバナンスと相互運用性のためのオプションのアイスバーグカタログ

その結果、ホット層 (PGD) とコールド層 (オブジェクト ストレージ) にわたるデータへのシームレスで透過的なアクセスが実現します。

関連概念: [データ階層化](#)

3.2 EDB 分析において階層型テーブルが重要な理由

階層型テーブルは組織にとって次のようなメリットがあります。

- PGD 運用ストレージを無駄なく、高パフォーマンスに保つ
- 古いデータをオブジェクトストレージにオフロードすることでストレージコストを削減
- 完全なデータセットへの統一されたクエリアccessを維持する
- Postgres で OLTP と OLAP の両方のユースケースをサポート
- 履歴分析のためのレイクハウスアーキテクチャを実装する

関連概念: [Analytics Accelerator concepts](#)

3.3 EDB が階層型テーブルを実装する方法

コアコンポーネント:

- PGD 自動パーティション:
- 時間ベースの新しいパーティションを自動的に作成します
- オフロードのタイミングを制御するために `analytics_offload_period` を定義します
- PGFS:
- オフロードされたデータ用のオブジェクトストレージへのアクセスを提供します
- PGAA:
- PGD 層と Iceberg 層にわたる統一されたクエリを可能にします
- コールドデータ専用のオフロードビュー (`table_offloaded`) を作成します
- オプションの氷山カタログ:
- ガバナンスとクロスプラットフォームの相互運用性をサポート

クエリの動作:

- 親 PGD テーブルに対するクエリは、ホット データとコールドデータの両方に自動的にアクセスします。
- PGD クエリ プランナーは、WHERE 句をプッシュして、ストレージ層間のアクセスを最適化します。

関連概念:

- [ストレージとコンピューティングの分離](#)
- [ETL/ELT](#)
- [OLAP / OLTP](#)

3.4 一般的な使用例

Use case	Tiered Tables + Analytics Accelerator
IoT and telemetry	Manage large time-series datasets with automated offload
Regulatory and financial data retention	Cost-efficient storage of historical data with full auditability
Analytical reporting on historical data	Use Lakehouse nodes to query offloaded data at scale
Hybrid OLTP / OLAP patterns	Keep current data fast on PGD, analyze large history in Iceberg

3.5 役割ベースのガイダンス

データベース管理者 (DBA) *Analytics Accelerator for your role: DBA*

データサイエンティスト/アナリスト *Analytics Accelerator for your role: Data scientist / analyst*

DevOps / SRE *Analytics Accelerator for your role: DevOps / SRE*

アプリケーション開発者 *Analytics Accelerator for your role: Application developer*

3.6 学習パス

Analytics Accelerator 101: Foundational concepts

Analytics Accelerator 201: Practical application and core solutions

Analytics Accelerator 301: Advanced techniques and optimization

3.7 関連概念

- データ階層化
- ストレージとコンピューティングの分離
- *ETL/ELT*
- *OLAP / OLTP*

3.8 次のステップ

ハイブリッドマネージャーユーザー向け Tiered Tables

ハウツーガイド [Configure PGFS storage for Tiered Tables](#)

[Configure PGD node group for analytics offload](#)

[Configure BDR AutoPartition with analytics offload](#)

[Query Tiered Tables from PGD and Lakehouse](#)

[Analytics Accelerator learning guide](#) でさらに詳しく調べてください。

第4章 Reference - EDB Postgres® AI Lakehouse

HCP Lakehouse は、データに対して分析クエリを実行するための強力なツールです。Postgres 上に構築されているため、既存の Postgres に関する知識とツールを活用できます。

Lakehouse を最大限に活用するには、[Quick start](#) に従ってベンチマークデータをクエリしてください。その後、独自のデータと [query the delta lake tables](#) をロードしてみてください。

このセクションでは、知っておくべき重要な事項のいくつかについて詳しく説明します。

- [Benchmarking datasets](#)
- [Queries in Lakehouse](#)
- [Delta Table tools](#)
- [DirectScan and fallback modes](#)
- [Users](#)
- [Loading data](#)
- [PGAA functions reference](#)

ストレージの場所を管理するための PGFS 関数の詳細については、[AI Factory PGFS documentation](#) を参照してください。

4.1 Benchmarking datasets

Lakehouse ノードをプロビジョニングすると、サンプルのベンチマークデータセットを含む AWS us-east-1 のパブリック S3 バケットを指すように事前構成されます。

注釈

これらのデータセットにアクセスすると、データ転送のコストが発生し、リージョン間の遅延が発生する可能性があります。

これらのデータセット内のテーブルをスキーマ名で参照することでクエリできます。

Schema name	Dataset
<i>tpcds_sf_1</i>	TPC-DS, Scale Factor 1
<i>tpcds_sf_10</i>	TPC-DS, Scale Factor 10
<i>tpcds_sf_100</i>	TPC-DS, Scale Factor 100

次のページに続く

表 1 – 前のページからの続き

Schema name	Dataset
<i>tpcds_sf_1000</i>	TPC-DS, Scale Factor 1000
<i>tpch_sf_1</i>	TPC-H, Scale Factor 1
<i>tpch_sf_10</i>	TPC-H, Scale Factor 10
<i>tpch_sf_100</i>	TPC-H, Scale Factor 100
<i>tpch_sf_1000</i>	TPC-H, Scale Factor 1000
<i>clickbench</i>	ClickBench, 100 million rows
<i>brc_1b</i>	Billion row challenge

!!!note 「ClickBench データに関する注意事項」

データ列 (EventData) は日付ではなく整数です。

ClickBench の列名には大文字が含まれているため、引用符で囲む必要がありますが、Postgres では引用符で囲まれていない識別子は小文字と大文字を区別しません。例：

```
❑ select "Title" from clickbench.hits;
```

```
❑ select Title from clickbench.hits; !!!
```

4.2 Queries

読み取り専用クエリがサポートされています。オブジェクトストレージに直接書き込むことはできません。これには、Postgres のすべての組み込み関数、ステートメント、および型が含まれます。また、導入するディストリビューションに応じて、EDB Postgres Advanced Server または EDB Postgres Extended Server で提供されるものも含まれます。

データの挿入、更新、削除、その他の変更はできません。CREATE TABLE はノード上に通常の Postgres テーブルを作成するためにのみ使用できます。このノードは一時的なものであり、終了すると破棄されます。CREATE TABLE... USING PGAA は、S3 互換オブジェクトストレージ内の Delta Tables を参照する外部テーブルを作成するためにも使用できます。

独自の ETL スクリプトを使用するか、Delta テーブルまたは Iceberg テーブルへの書き込みをサポートするサードパーティの ETL プロバイダーを使用して、帯域外でバケットにデータをロードする必要があります。

詳細については、[Loading data](#) を参照してください。

唯一の例外は、Postgres のシステムテーブル、つまりユーザー、ロール、権限の保存に使用されるテーブルです。これらのテーブルはローカルブロックデバイスに保存され、バックアップとリストアに含まれます。そのため、CREATE USER、CREATE ROLE、または GRANT USAGE を使用すれば、これらのユーザー、ロール、権限は再起動やリストア後も保持されます。

4.3 Delta Lake Table tools

4.3.1 Delta Lake テーブルの作成

deltalake Python ライブラリの使用

deltalake Python ライブラリを使用してデルタテーブルを作成し、バケットに書き込むことができます。

lakehouse-loader ユーティリティの使用

また、EDB がこのタスク用に作成した `lakehouse-loader` ユーティリティを使用して、任意の Postgres インスタンスからストレージバケット内の Lakehouse テーブルにデータをエクスポートすることもできます。

たとえば、`lakehouse-loader` ユーティリティの場合:

```
export PGPASSWORD="..."
export AWS_ACCESS_KEY_ID="..."
export AWS_SECRET_ACCESS_KEY="..."

# export other AWS envvars

./lakehouse-loader postgres-to-delta postgres://test-user@localhost:5432/test-db -q
→ "SELECT * FROM some_table" s3://my-bucket/my_schema/my_table
```

このコードは、`test-db` データベースの `some_table` テーブルから `my-bucket` バケットの `my_schema/my_table` パスにある Delta テーブルにデータをエクスポートします。

`my_schema/my_table` パスの Delta テーブルを参照する外部テーブルを作成することで、Lakehouse ノードでこのテーブルをクエリできます。実行方法の詳細については、[Querying Delta Lake Tables](#) を参照してください。

4.4 DirectScan and fallback modes

4.4.1 ダイレクトスキャン

Postgres Lakehouse は、クエリ全体を DataFusion（可能な場合のクエリ処理に使用されるベクトル化クエリ）に「プッシュダウン」できる場合に最も高速になります。（今後、部分的なプッシュダウンのサポートが追加され、よりきめ細かな制御が可能になります。）

Postgres Lakehouse は、クエリを 2 つのモードで実行できます。まず、DataFusion ベースの専用列指向データベースである Seafoal を使用して、クエリ全体の実行を試みます。Seafoal が JSON などの PostgreSQL 固有の操作を使用しているなどの理由でクエリ全体を実行できない場合、Postgres Lakehouse は PostgreSQL エグゼキュータにフォールバックし、Seafoal はテーブルの全内容を PostgreSQL エグゼキュータにストリーミングします。

クエリが非常に遅い場合は、それが原因である可能性があります。

4.4.2 EXPLAIN を使用してクエリプランを確認する

クエリに対して `EXPLAIN` を実行し、最上位のクエリノードが `DirectScan` であることを確認することで、どのモードが使用されているかを確認できます。例：

```
explain select count from (select count(*) from tpch_sf_1.lineitem);
```

QUERY PLAN

→

(次のページに続く)

(前のページからの続き)

```

--
--
Aggregate (cost=167.52..167.55 rows=1 width=8)
  -> Append (cost=0.00..165.01 rows=1001 width=0)
    -> CompatScan on "16529" lineitem_2 (cost=100.00..150.00 rows=1000 width=0)
        SeafoalPlan: logical_plan
            TableScan: tpch_sf_1.lineitem projection=[l_orderkey, l_partkey, l_
--suppkey, l_linenum, l_quantity, l_extendedprice, l_discount, l_tax, l_returnflag,
l_linestatus, l_shipdate, l_commitdate, l_receiptdate, l_shipinstruct, l_shipmode, l_
--comment]
(6 rows)

```

この場合、クエリは PostgreSQL によって実行され、Seafoal はテーブルをスキャンする際にのみ使用されます（下部の CompatScan を参照）。この場合の修正方法は、内部の COUNT(*) 列に明示的に名前を付けることです。Seafoal は暗黙的に count(*) という名前を付けますが、PostgreSQL は count という名前を使用します。

```

edb_admin=> explain select count from (select count(*) as count from tpch_sf_1.
--lineitem);

```

QUERY PLAN

```

-----
DirectScan: logical_plan
  Projection: COUNT(*) AS count
    Aggregate: groupBy=[[]], aggr=[[COUNT(UInt8(1)) AS COUNT(*)]]
      TableScan: tpch_sf_1.lineitem projection=[]
(4 rows)

```

ここで、上部に DirectScan が表示されています。これは、Seafoal がクエリ全体を実行していることを意味します。

クエリを言い換えて Seafoal で完全に実行できるようにすることが難しい場合は、サポートチケットを開いてください。

4.5 User management

Lakehouse ノードをプロビジョニングする際は、パスワードを入力する必要があります。edb_admin ユーザーとしてログインするには、このパスワードが必要です。edb_admin ユーザーは、ユーザー、ロール、権限付与を作成できるスーパーユーザーアカウントです。

HCP はこのパスワードを保存しません。

edb_admin の資格情報を共有するか、それを使用して他のユーザーを作成し、それらの資格情報を配布することができます。

4.6 Loading data

4.6.1 独自のデータを持ち込む

Lakehouse ノードを、デルタテーブルを含む任意の S3 バケットに向けることも可能です。ただし、これを行うにはいくつか重要な注意点があります。（これらは将来のリリースで解決される予定です。）

注意事項

- テーブルは、その場所内で **Delta Lake Tables** として保存する必要があります。
- **Delta Lake テーブル** (または **Delta テーブル**) は、いくつかの JSON メタデータを含む Parquet ファイルのフォルダーです。

バケットにデータをロードする

lakehouse-loader ユーティリティを使用すると、任意の Postgres インスタンスからストレージバケット内の Delta Tables にデータをエクスポートできます。このユーティリティの入手方法と使用方法の詳細については、[Delta Lake Table tools](#) をご覧ください。

詳細については、[querying delta tables](#) を参照してください。

4.7 PGAA functions reference

4.7.1 概要

インスタンス上のすべての PGAA テーブル、その保存場所、およびパスを一覧表示します。

4.7.2 パラメータ

なし。

4.7.3 戻り値

Column	Type	Description
<i>schema_name</i>	<i>text</i>	テーブルのスキーマ
<i>table_name</i>	<i>text</i>	テーブルの名前
<i>storage_location_name</i>	<i>text</i>	テーブルの保存場所の名前
<i>path</i>	<i>text</i>	保存場所のテーブルへのパス
<i>managed_by</i>	<i>text</i>	テーブルを作成したユーザーの名前
<i>format</i>	<i>text</i>	テーブルの形式（現在は `delta` または `iceberg`）

pgaa.lakehouse_table_stats()

4.7.4 概要

Lakehouse テーブルの最新バージョンのサイズ (バイト単位) とすべてのバージョンの合計サイズを返します。

4.7.5 パラメータ

Parameter	Type	Description
<i>relation</i>	<i>regclass</i>	サイズを取得するテーブルの名前

4.7.6 戻り値

Column	Type	Description
<i>latest_snapshot_size</i>	<i>bigint</i>	テーブルの最新スナップショットのサイズ (バイト単位)
<i>total_size</i>	<i>bigint</i>	テーブルのすべてのバージョンの合計サイズ (バイト単位)

4.8 PGAA functions reference

4.8.1 pgaa.create_storage_location

概要

S3 互換のオブジェクトストレージの場所を参照する新しいストレージの場所を作成します。

パラメータ

Parameter	Type	Description
<i>name</i>	<i>text</i>	保管場所の名前
<i>uri</i>	<i>text</i>	S3 互換オブジェクトストレージの場所の URI (例: <code>s3://bucket-name`または` s3://bucket-name/prefix</code>)
<i>options</i>	<i>json</i>	保存場所のオプション設定

オプション

Option	Type	Description
<i>access_key_id</i>	<i>text</i>	保存場所のアクセスキー ID

[次のページに続く](#)

表 6 – 前のページからの続き

Option	Type	Description
<i>secret_access_key</i>	<i>text</i>	保管場所の秘密アクセスキー
<i>session_token</i>	<i>text</i>	保存場所のセッショントークン
<i>region</i>	<i>text</i>	保管場所の地域
<i>endpoint</i>	<i>text</i>	保管場所のエンドポイント
<i>bucket</i>	<i>text</i>	保管場所のバケット
<i>use_http</i>	<i>boolean</i>	保存場所には HTTPS ではなく HTTP を使用する
<i>skip_signature</i>	<i>boolean</i>	保存場所の署名検証をスキップする

4.9 Supported AWS instances

Lakehouse ノードをデプロイする際は、**m6id** ファミリーのインスタンスからインスタンスタイプを選択する必要があります。重要なのは、これらのインスタンスには NVMe ドライブが接続されていることです。

インスタンスは一時的なものです。これらの NVMe ドライブは、クエリ処理中のスピルアウト領域と、デルタテーブルをディスクにキャッシュするためにのみ使用されます。NVMe ドライブ上のすべてのデータは、クラスターがシャットダウンされると失われます。

システム テーブルは永続化されます。システム テーブル内の永続データ (ユーザー、ロールなど) は、接続されたブロック ストレージデバイスに保存され、一時停止/再開サイクルを経ても保持されます。

サポートされているインスタンス

API Name	Memory	vCPUs	Cores	Storage
<i>m6id.large</i>	8.0 GiB	2 vCPUs	1	118 GB NVMe SSD
<i>m6id.xlarge</i>	16.0 GiB	4 vCPUs	2	237 GB NVMe SSD
<i>m6id.2xlarge</i>	32.0 GiB	8 vCPUs	4	474 GB NVMe SSD
<i>m6id.4xlarge</i>	64.0 GiB	16 vCPUs	8	950 GB NVMe SSD
<i>m6id.8xlarge</i>	128.0 GiB	32 vCPUs	16	1900 GB NVMe SSD
<i>m6id.12xlarge</i>	192.0 GiB	48 vCPUs	24	2850 GB (2 * 1425 GB NVMe SSD)
<i>m6id.16xlarge</i>	256.0 GiB	64 vCPUs	32	3800 GB (2 * 1900 GB NVMe SSD)
<i>m6id.24xlarge</i>	384.0 GiB	96 vCPUs	48	5700 GB (4 * 1425 GB NVMe SSD)
<i>m6id.32xlarge</i>	512.0 GiB	128 vCPUs	64	7600 GB (4 * 1900 GB NVMe SSD)

第5章 Delta Lake

Delta Lake は、ACID 保証、スキーマの適用、タイムトラベル、スケーラブルなパフォーマンスを備えた最新のデータレイクを強化するオープンソースのテーブル形式とトランザクションレイヤーです。

EDB は、Analytics Accelerator の一部として Delta Lake テーブルをクエリするためのファーストクラスのサポートを可能にし、大規模なレイクハウスデータで Postgres SQL を操作できるようにします。

Hybrid Manager (HM) 内で Delta Lake が使用および管理される方法の詳細については、[Delta Lake](#) を参照してください。

5.1 Delta Lake とは

Delta Lake は、S3、GCS、Azure Data Lake Storage などのオブジェクトストレージシステムにデータベースのような信頼性と一貫性を追加します。

主な特徴:

- 強力なデータ一貫性を備えた ACID トランザクション
- スキーマの強制とスキーマの進化のサポート
- データの履歴バージョンを照会するためのタイムトラベル
- バッチおよびストリーミングのサポート
- [Apache Parquet](#) 上にトランザクション ログ (`_delta_log`) を備えたオープンフォーマット

関連概念: [オープンテーブル形式](#)

5.2 Delta Lake が EDB 分析にとって重要な理由

Delta Lake のサポートにより、Analytics Accelerator は次のことが可能になります。

- Postgres SQL を使用して既存の Delta Lake テーブルをクエリする
- Postgres、Spark、Trino、Presto など、相互運用可能なデータアーキテクチャをサポートします。
- システム間の不要なデータの重複や移動を排除
- Lakehouse クラスター内の分析クエリに、管理されたバージョン管理されたデータを提供する

関連概念: [EDB Postgres Lakehouse](#)

5.3 EDB が Delta Lake を活用する方法

EDB Postgres Lakehouse クラスターは、PGAA 拡張機能を通じて Delta Lake データへのアクセスを提供します。

- `CREATE TABLE ... USING PGAA WITH (pgaa.format = 'delta', ...)` を使用して Delta Lake 外部テーブルを定義する
- [Apache DataFusion](#) を使用したベクトル化実行により、Parquet ベースの Delta Lake テーブルを効率的にクエリします。
- スキーマの進化やタイムトラベルなどの Delta Lake 機能のメリットを享受

現在の主なサポート: 既存の Delta Lake テーブルのクエリ。PGD オフロードは現在 Iceberg を対象としています。Delta 読み取りサポートにより、既存の Delta ベースのデータレイクとの統合が可能になります。

5.4 一般的な使用例

Use case	Delta Lake + Analytics Accelerator
Business intelligence reporting	Query Delta Lake tables using Postgres SQL and BI tools
Data science and machine learning	Access Delta tables for model training and feature engineering
Data lake governance	Utilize Delta's ACID guarantees with Lakehouse SQL access
Cross-platform interoperability	Query Delta Lake data alongside Spark, Trino, and Postgres Lakehouse

5.5 役割ベースのガイダンス

- データベース管理者 (DBA)
- *Data scientist / analyst*
- *DevOps / SRE*
- アプリケーション開発者

5.6 学習パス

- *Analytics Accelerator 101: Foundational concepts*
- *Analytics Accelerator 201: Practical application and core solutions*
- *Analytics Accelerator 301: Advanced techniques and optimization*

5.7 関連概念

- データレイクハウス
- 列指向ストレージ形式
- ストレージとコンピューティングの分離
- オープンテーブル形式

5.8 次のステップ

ハイブリッド マネージャー ユーザー向け

- [Delta Lake](#)

ハウツーガイド

- [Query Delta Lake tables](#)
- [Configure PGFS for Delta Lake](#)
- [Performance tuning for Delta Lake queries](#)

Analytics Accelerator learning guide でさらに詳しく調べてください。

第6章 Apache Iceberg

Apache Iceberg は、オブジェクトストレージ上に構築されるデータレイク向けに設計された、高性能なオープンテーブル形式です。スケーラブルなメタデータ管理、スキーマ進化、そして ACID トランザクションを提供します。

EDB は Analytics Accelerator 内で Iceberg サポートを有効にし、Postgres が大規模なデータレイクテーブルを効率的にクエリおよび管理できるようにします。

Hybrid Manager (HM) での実装については、[Working with Apache Iceberg in EDB HM](#) を参照してください。

6.1 Apache Iceberg とは

Iceberg は、データレイク向けに設計されたテーブル形式です。データファイル (Parquet、ORC、Avro) の整理と追跡方法を定義し、オブジェクトストレージにデータベースのような多くの機能を追加します。

6.1.1 主な目標

- **信頼性:** 同時操作によるデータの一貫性を確保
- **パフォーマンス:** パーティションプルーニングとメタデータの最適化により高速クエリが可能になります
- **スケーラビリティ:** ペタバイト規模のテーブルを管理するように設計されています
- **オープン性:** 幅広いエコシステムのサポートを備えた Apache Software Foundation プロジェクト

関連概念: *Generic concepts - open table formats*

6.2 主な機能と利点

- **スキーマの進化:** テーブルデータを書き換えずに列を安全に追加、削除、または変更できます
- **隠しパーティショニング:** データを透過的にパーティション分割し、効率的なパーティションプルーニングを可能にします
- **タイムトラベルとバージョン管理:** 履歴テーブルバージョンのクエリをサポート
- **ACID トランザクション:** 楽観的同時実行性を使用してトランザクションの保証を提供します
- **複数のファイル形式のサポート:** Parquet、ORC、Avro ファイルを管理します
- **効率的なメタデータ管理:** メタデータツリー構造を使用して、高速なクエリ計画とデータスキップを実現します。

- **カタログ統合:** REST ベースのカタログ (Project Nessie、AWS Glue)、Hive Metastoreなどをサポートします

6.3 EDB 分析において Apache Iceberg が重要な理由

Iceberg サポートにより、Analytics Accelerator は次のことが可能になります。

- **運用データと分析データを統合:** 従来の Postgres データと並行して Iceberg テーブルをクエリします
- **費用対効果の高いオブジェクトストレージを活用:** 大規模な履歴データセットを効率的に保存
- **パフォーマンスの向上:** ベクトル化されたエンジンで Iceberg メタデータとパーティションブローニングを使用する
- **階層型ストレージをサポート:** PGD は階層型テーブル戦略の一環として Iceberg にデータをオフロードできます
- **相互運用性を実現:** Spark、Presto、Trino、Flink、Dremioなどのツールとデータを共有します

関連概念: [Analytics Accelerator concepts - EDB Postgres Lakehouse](#)

6.4 EDB ソリューションが Iceberg サポートを実装する方法

6.4.1 PGAA 拡張

Lakehouse ノードと PGD ノード内の Postgres Analytical Appliance (PGAA) コンポーネントは、Iceberg メタデータとデータファイルを読み取り、操作できます。

6.4.2 カタログ接続

PGAA は、以下を使用して Iceberg カタログ (Lakekeeper、AWS Glue、REST カタログなど) に接続します。

```
SELECT pgaa.add_catalog(...);  
SELECT pgaa.attach_catalog(...);
```

これにより、テーブルの検出、スキーマの読み取り、データのクエリが可能になります。

Iceberg テーブルのクエリ ユーザーは、Iceberg テーブルを参照する外部テーブルを Postgres で定義できます。

```
CREATE TABLE my_table USING PGAA WITH (  
  pgaa.format = iceberg,  
  pgaa.managed_by = my_catalog,  
  pgaa.catalog_namespace = my_db,  
  pgaa.catalog_table = my_table  
);
```

定義されると、テーブルは Postgres SQL を使用して完全にクエリ可能になります。

6.4.3 Iceberg テーブルへの書き込み

PGD は、オブジェクト ストレージ内の Iceberg 形式にデータをオフロードできます。

新しい氷山テーブルを作成できます

既存の氷山テーブルは、

オフロード操作は通常、階層型テーブル戦略に従います。

関連概念: 階層型テーブル

6.5 関連トピック

- *[Learn Generic Analytics Concepts](#)*
- *[Learn Analytics Accelerator Concepts](#)*
- *[How-To: Configure an Iceberg REST catalog connection in Hybrid Manager](#)*

第7章 EDB Postgres Lakehouse

EDB Postgres® Lakehouse は、Postgres ベースの分析に最新のレイクハウスアーキテクチャをもたらします。

Apache Iceberg や Delta Lake などのオープンテーブル形式を使用して、オブジェクト ストレージ (S3 互換) に保存されたデータに対して高速な SQL ベースの分析を可能にします。

Hybrid Manager (HM) での実装と管理については、[EDB Postgres Lakehouse](#) を参照してください。

7.1 EDB Postgres Lakehouse とは

EDB Postgres Lakehouse は、次のようなアーキテクチャパターンと機能のセットです。

- Postgres を最新のレイクハウスパターンと統合する
- オブジェクトストレージ内のデータに対して高速かつスケーラブルな分析を提供する
- ベクトル化されたクエリ実行と列指向のストレージ形式を使用する
- 相互運用性とデータガバナンスのためにオープンテーブル形式を活用する

関連概念: [データレイクハウス](#)

7.2 Lakehouse が EDB 分析にとって重要な理由

Lakehouse を使用すると、Analytics Accelerator ユーザーは次のことが可能になります。

- オブジェクト ストレージ上で高速 SQL 分析を実行 - データ移動は不要
- コスト効率を高めるためにストレージとコンピューティングを分離する
- 外部ツール (Spark、Trino、Flink、データサイエンスフレームワーク) との相互運用性をサポート
- PGD 階層化テーブルのクエリを Iceberg にシームレスにオフロード
- Postgres を中心とした統合 OLTP + OLAP アーキテクチャを実現します

関連概念: [Analytics Accelerator concepts](#)

7.3 EDB がレイクハウスアーキテクチャを実装する方法

コアコンポーネント:

- **EDB Postgres Lakehouse ノード** :
- ステートレス分析コンピューティングノード

- ハイブリッドマネージャー（HM）またはセルフマネージドでプロビジョニングおよび管理
- ベクトル化されたクエリエンジン：
- [Apache DataFusion](#) によって提供
- Parquet やその他の列指向形式でデータを効率的に処理します
- **PGAA**：
- Lakehouse の動作を有効にする Postgres 拡張機能:
- 氷山とデルタ湖を見渡す外部テーブル
- PGD ホットデータとオフロードされたコールドデータへの統合アクセス
- **PGFS**：
- オブジェクトストレージへの統合アクセス層
- S3、GCS、MinIO、Azure Data Lake Storage、および互換システムをサポート
- **オープンテーブル形式**：
- Apache Iceberg（カタログを含む完全サポート）
- Delta Lake（読み取り専用サポート）

関連概念:

- [ストレージとコンピューティングの分離](#)
- [ベクトル化されたクエリエンジン](#)
- [列指向ストレージ形式](#)
- [オープンテーブル形式](#)

7.4 一般的な使用例

Use case	EDB Postgres Lakehouse capability
Business intelligence & reporting	Run fast, scalable SQL on data in object storage
Historical analytics	Seamlessly query offloaded PGD Tiered Tables
Data lake analytics	Query existing Iceberg and Delta Lake tables without ETL
Data science pipelines	Provide efficient access to training data and features
Hybrid architectures	Enable Postgres-centered OLTP + OLAP patterns

7.5 役割ベースのガイダンス

データベース管理者（DBA）

- *Analytics Accelerator for your role: DBA*

データサイエンティスト/アナリスト

- *Analytics Accelerator for your role: Data scientist / analyst*

DevOps / SRE

- *Analytics Accelerator for your role: DevOps / SRE*

アプリケーション開発者

- *Analytics Accelerator for your role: Application developer*

7.6 学習パス

- *Analytics Accelerator 101: Foundational concepts*
- *Analytics Accelerator 201: Practical application and core solutions*
- *Analytics Accelerator 301: Advanced techniques and optimization*

7.7 関連概念

- データレイクハウス
- ストレージとコンピューティングの分離
- ベクトル化されたクエリエンジン
- 列指向ストレージ形式
- オープンテーブル形式

7.8 次のステップ

ハイブリッドマネージャーユーザー向け

- [EDB Postgres Lakehouse](#)

ハウツーガイド

- [Create a Lakehouse cluster](#)
- [Query existing Apache Iceberg tables](#)
- [Query Delta Lake tables](#)

[Analytics Accelerator learning guide](#) でさらに詳しく調べてください。

第8章 Analytics Accelerator learning resources

このガイドは、Postgres® 上に構築された Analytics Accelerator のすべてのドキュメントを参照するのに役立ちます。コアコンセプトの理解、ステップバイステップのチュートリアルの実施、特定の問題の解決、あるいはアナリティクスを自分の役割にどのように適用するかを検討など、このガイドは適切なリソースを見つけるのに役立ちます。

ドキュメントは、効果的に学習し、情報を効率的に見つけられるように構成されています。

8.1 ドキュメントのカテゴリ

8.1.1 構造化された学習パス

構造化された学習パスに従って、基礎概念から高度な実装まで進みます。

学習パスを調べる: *Analytics Accelerator learning paths*

- *Analytics Accelerator 101: Foundational concepts*
- *Analytics Accelerator 201: Practical application and core solutions*
- *Analytics Accelerator 301: Advanced techniques and optimization*

8.1.2 概念の説明

分析の原則、最新のデータアーキテクチャ、分析アクセラレータのテクノロジーと戦略に関する基礎知識を構築します。

探索: Explained: Analytics Accelerator concepts and technologies

一般的な分析の概念、Analytics Accelerator のビジョン、EDB Postgres Lakehouse、Apache Iceberg、Delta Lake、Tiered Tables、アーキテクチャパターンなどのコアテクノロジーの詳細について説明します。

8.1.3 チュートリアル

完全なプロジェクトやシナリオを通して実践的に学びます。実践的なスキルを身につけ、ご自身の環境に分析を適用しましょう。

探索: Analytics Accelerator tutorials index

8.1.4 ハウツーガイド

Analytics Accelerator に関連する一般的なタスクの簡潔な手順を見つけます。

探索: [Analytics Accelerator how-to guides index](#)

8.1.5 ユースケースとソリューション

Analytics Accelerator がビジネス上の問題に対処し、業界全体にわたるアーキテクチャパターンをサポートする方法をご覧ください。

探索: [Analytics Accelerator use cases](#)

8.1.6 ペルソナベースのガイダンス

あなたの技術的役割に合わせて調整された、厳選された情報と学習パスにアクセスします。

探索: [Analytics Accelerator for your role](#)

8.1.7 ハイブリッド マネージャー (HM) の分析アクセラレータ

このセクションでは、EDB のマネージド サービスのユーザー向けに、Analytics Accelerator の概念とテクノロジーが Hybrid Manager でどのように実装、プロビジョニング、管理されるかについてのドキュメントを提供します。

探索: [Analytics in Hybrid Manager](#)

第9章 Analytics Accelerator learning resources

この学習ガイドを使用して、コア概念から始める場合でも、高度なアーキテクチャを深く理解する場合でも、Analytics Accelerator ドキュメントの学習パスを計画してください。

9.1 Analytics Accelerator concepts and technologies

このセクションを使用して、Analytics Accelerator を定義する **コア概念**、**テクノロジー**、**アーキテクチャパターン**、および **戦略**を明確に理解します。

このセクションは以下を補足します:

- [How-To Guides](#) — ステップバイステップのタスク用
- [Analytics Terminology](#) — この分野で使用されているキーワード

EDB Hybrid Manager (HM) 環境に固有の説明については、[Analytics in Hybrid Manager](#) を参照してください。

9.2 Analytics Accelerator concepts and technologies

9.2.1 基礎的な分析と最新のデータアーキテクチャ

今日の分析環境を形成する **業界のトレンド** と **アーキテクチャパターン** を理解します。

一般的な概念

- [Analytics Accelerator generic concepts](#)

データ ウェアハウス、データ レイク、レイクハウスアーキテクチャ、列指向ストレージ、ベクトル化エンジン、ストレージとコンピューティングの分離などの基本用語。

追加記事（近日公開予定）

- データレイクハウスへの進化
- オープンテーブル形式の重要性（アイスバーグ、デルタレイク、Hudi）
- 分析における列指向ストレージと行指向ストレージの利点とトレードオフ

9.3 Analytics Accelerator concepts and technologies

9.3.1 Analytics Accelerator のコアコンポーネントの詳細

EDB Postgres で高度な分析を可能にする コア コンポーネントについて学習します。

EDB Postgres レイクハウス

- EDB Postgres Lakehouse: 概要 (近日公開予定)
- EDB Postgres Lakehouse: 詳細な概念と用語 (近日公開予定)

EDB Postgres によるオープンテーブル形式

- EDB ソリューションを使用した Apache Iceberg の理解 (近日公開)
- EDB ソリューションを使用した Delta Lake の理解 (近日公開)

EDB Postgres Distributed (PGD) によるデータ管理と階層化

- EDB Postgres による階層化テーブルの理解 (近日公開予定)

基礎となるエンジンコンポーネント

- EDB のレイクハウスにおける PGAA と PGFS の役割 (近日公開予定)
- EDB Postgres での Apache DataFusion によるベクトル化されたクエリ実行 (近日公開)

9.4 Analytics Accelerator concepts and technologies

9.4.1 EDB ハイブリッドマネージャー (HM) 内の分析を理解する

EDB の分析テクノロジーが **Hybrid Manager** でどのように実装され、管理されるか。

主要な HM 分析ドキュメントのエントリポイント

- [Analytics in Hybrid Manager](#)
 - [Analytics concepts within Hybrid Manager](#)
-

9.4.2 AI/ML ワークロードと相互運用性

Analytics Accelerator は **汎用分析** をサポートし、AI/ML パイプラインのプラットフォーム コンポーネントとしても機能します。

- Lakehouse ノードは、モデルのトレーニングで使用される大規模なデータセットへの効率的なアクセスを提供します。
- 階層化されたデータ パターンと ELT パイプラインにより、AI/ML 処理用にデータをステージングできます。

AI/ML 固有の概念については、[AI Factory concepts](#) を参照してください。

Analytics Accelerator concepts

このページでは、EDB が **Analytics Accelerator** と **Hybrid Manager (HM)** を通じて最新の分析機能を提供する方法について説明します。

確立された業界のパターンとテクノロジーを基盤としています。この分野に初めて触れる場合は、まず以下の内容から始めてください。

- [Analytics Terminology](#) — 使用されている重要な用語を学ぶ
- [Analytics Accelerator generic concepts](#) — 一般的なアーキテクチャとパターンを理解する

EDB の Postgres 分析ビジョン

EDB は Analytics Accelerator を通じて最新の分析パターンを実装し、Postgres が運用ワークロードと分析ワークロードの両方の統合プラットフォームとして機能できるようにします。

主な原則:

- **運用データの近くで** 分析を実行し、データの移動と遅延を削減します
- オブジェクト ストレージの高速分析には、**オープン フォーマット** とベクトル化されたクエリ エンジンを使用します。
- コストとパフォーマンスのバランスをとるために **データ階層化**をサポート
- コンピューティングとストレージを分離し、独立したスケーリングを可能にする
- ハイブリッドマネージャーを介してスタック全体を管理します

関連概念: [データレイクハウス](#)

主要な EDB ソリューションとテクノロジー

EDB Postgres レイクハウス

Lakehouse アーキテクチャにより、EDB Postgres はオブジェクトストレージ内のデータをクエリできるようになります。

- Lakehouse ノードは **オープン テーブル形式** でデータをクエリします (Iceberg、Delta Lake)
- **Apache DataFusion** によるベクトル化されたクエリ実行
- **Parquet** などの列指向ストレージ形式
- コンピューティングとストレージの分離

関連概念:

- **列指向ストレージ形式**
 - **ベクトル化されたクエリエンジン**
 - **ストレージとコンピューティングの分離**
 - **オープンテーブル形式**
-

ベクトル化されたクエリの最適化

Analytics Accelerator は Lakehouse ノードに **Apache DataFusion** を埋め込みます。

- データを **列状のバッチ** として処理します
- **SIMD** 命令を使用してパフォーマンスを高速化します
- **Parquet 形式のデータ** に最適化

関連概念: **ベクトル化されたクエリエンジン**

EDB Postgres 分散 (PGD) と階層化テーブル

EDB Postgres Distributed (PGD) は階層型テーブルを実現します:

- **BDR AutoPartition** は時間ベースのパーティショニングを管理します
- コールドデータは自動的に **オブジェクトストレージ** にオフロードされます (アイスバーグテーブル)
- ホットデータはトランザクション PGD ノードに残ります

これにより、運用パフォーマンスとコスト効率の高い履歴データアクセスのバランスが取れます。

関連概念:

- **データ階層化**
 - **ETL/ELT**
-

高度なクエリ最適化

Analytics Accelerator は、分析のためのコア Postgres 機能を改善します。

- 強化された並列クエリ処理
- 分析ワークロード向けに最適化された結合アルゴリズム
- 高性能 OLAP クエリのためのベクトル化エンジンとの緊密な統合

関連概念: [OLAP / OLTP](#)

分析と AI/ML ワークロード

Analytics Accelerator は、従来の分析と AI/ML データパイプラインをサポートします。

- レイクハウスノードは、モデルのトレーニングで使用される大規模なデータセットへの効率的なアクセスを提供します
- 階層型データパイプラインは、コスト効率の高い AI/ML ステージングパターンをサポートします
- オープンテーブル形式により、ベクター検索および **検索拡張生成 (RAG)** ワークフローとの統合が可能

関連概念:

- [セマンティック検索](#)
 - [ベクトル埋め込み](#)
 - [検索拡張生成 \(RAG\)](#)
-

EDB が製品間で分析を実装する方法

ハイブリッドマネージャー (HM)

Hybrid Manager はコントロールプレーンとして機能します。

- **Lakehouse クラスタ** のプロビジョニングと管理
- 階層型テーブルの構成と監視
- ストレージの場所、カタログ接続、およびコンピューティングノードを管理する

EDB Postgres Advanced Server/ 拡張サーバー

EDB Postgres Advanced Server には次の機能が追加されます。

- 並列クエリ機能の改善
- 分析に役立つ高度な SQL 機能
- PGAA (Postgres Generic Analytics Adapter) 機能のネイティブサポート

EDB Postgres 分散 (PGD)

PGD は次の基盤を提供します。

- 高可用性トランザクションワークロード
 - データの階層化とオフロード（階層化テーブル）
 - Lakehouse クラスターと統合するスケーラブルなアーキテクチャ
-

次に進む

- [Analytics Terminology](#)
- [Analytics Accelerator generic concepts](#)
- [Analytics How-To Guides](#)
- [Analytics Use Cases](#)

Analytics Accelerator generic concepts

このハブでは、最新のデータ分析プラットフォームの基盤を形成する主要な業界の概念、アーキテクチャ、テクノロジーについて説明します。

これらの概念を理解することで、EDB のアナリティクス・アクセラレータが実証済みのパターンに基づいてどのように構築されているかを理解するのに役立ちます。EDB がこれらのアイデアをどのように実装しているかについては、[Analytics Accelerator concepts](#) をご覧ください。

アーキテクチャ

データウェアハウス

概要: 構造化された SQL ベースの分析に最適化された集中型リポジトリ。スキーマオンライトを採用し、高速な分析クエリ向けに設計されています。

理由: ビジネス インテリジェンスチームには、レポート作成や履歴分析を行うための信頼できる一貫性のあるデータが必要です。

方法: データは、ETL プロセスを通じて、強力な型指定と制約を持つリレーショナルテーブルにロードされます。

使用する理由: 既知の一貫性のあるレポートのニーズに対応できる高いパフォーマンスと、データ品質に対する強力なガバナンスが必要な場合。

これが [EDB Postgres Lakehouse](#) にどのような影響を与えるかを学びます。

データレイク

内容: 生データ（構造化データ、半構造化データ、非構造化データ）用の柔軟なリポジトリ。読み取り時にスキーマを実行します。

理由: データレイクを使用すると、すべてのデータをネイティブ形式で低コストで保存できます。

方法: データはファイルとしてオブジェクト ストレージ (S3、GCS など) にロードされます。多くの場合、形式は Parquet または ORC です。

使用する理由: ユースケースの全セットを把握する前であっても、利用可能なすべてのデータをキャプチャし、後で柔軟な処理を実行する必要がある場合。

[More on data lakes](#)

これが *EDB Postgres Lakehouse* にどのような影響を与えるかを学びます。

データレイクハウス

概要: データ レイクの柔軟性とデータウェアハウスのパフォーマンスおよび信頼性を組み合わせた最新のアーキテクチャ。

理由: 組織は、大量のデータを移動することなく、データサイエンスと BI の両方に 1 つのシステムを必要としています。

方法: データは、構造、トランザクション、メタデータを追加するオープンテーブル形式を使用してレイク (オブジェクト ストレージ) に保存されます。

使用する理由: スケーラブルで低コストのストレージが必要であり、ACID 保証と高パフォーマンスのクエリも必要な場合。

[Lakehouse architecture paper \(Armbrust et al.\)](#)

[Apache Iceberg](#)

[Delta Lake](#)

[Apache Hudi](#)

これが *EDB Postgres Lakehouse* にどのような影響を与えるかを学びます。

主要技術

列指向ストレージ形式

概要: 行ではなく列をまとめて保存するデータ形式。

理由: ほとんどの分析クエリは列のサブセットにのみアクセスします。列を直接読み取ることでパフォーマンスが向上し、I/O が削減されます。

方法: 一般的な形式は、圧縮と列ブローニングをサポートする [Apache Parquet](#) と [Apache ORC](#) です。

使用する理由: 大規模で読み取り負荷の高い分析クエリを実行し、スキャン速度を最大限に高めたい場合。

EDB Postgres Lakehouse でこれがどのように使用されるかを確認してください。

ベクトル化されたクエリエンジン

概要: 行ごとではなく列のバッチ (ベクトル) でデータを処理するクエリエンジン。

理由: 列形式のバッチは CPU キャッシュの動作と SIMD 命令に一致するため、分析が大幅に高速化されます。

方法: [Apache DataFusion](#) や [Velox](#) などのエンジンはベクトル化された実行を実装します。

使用する理由: 分析ワークロードが CPU に依存しており、パフォーマンスが重要である場合。

[ベクトル化されたクエリの最適化](#) でこれがどのように使用されるかを確認してください。

ストレージとコンピューティングの分離

概要: コンピューティング (クエリ エンジン) とストレージ (オブジェクト ストレージ) が分離されたアーキテクチャ。

理由: これにより、個別にスケーリングして、コストとパフォーマンスを最適化できます。

方法: Analytics Accelerator などのシステムは、データにはオブジェクトストレージ (S3 など) を使用し、クエリにはステートレス コンピューティングノードを使用します。

使用する理由: コンピューティングのニーズが急増したり予測不能になったりして、過剰なプロビジョニングを避けたい場合。

[EDB Postgres Lakehouse](#) でこれがどのように使用されるかを確認してください。

オープンテーブル形式

概要: データベース機能 (ACID トランザクション、スキーマの進化) をデータ レイクに提供する形式。

理由: オープンテーブル形式がなければ、レイクデータはばらばらのファイルの集合体となってしまいます。そのため、構造化とガバナンスが必要です。

どうやって：

- [Apache Iceberg](#) はメタデータ レイヤーとマニフェストファイルを使用します
- [Delta Lake](#) はトランザクション ログを使用します ([delta-rs](#))
- [Apache Hudi](#) はストリームファーストアップデートに重点を置いています

使用する理由: レイクベースのデータに対する強力なデータ管理が必要な場合。

[EDB Postgres Lakehouse](#) でこれがどのように使用されるかを確認してください。

コアコンセプト

OLAP / OLTP

何：

- OLAP: 大規模データセットに対する分析クエリ (集計、傾向)
- OLTP: 大容量、小規模なトランザクション (更新、挿入)

理由: ワークロードによってパフォーマンス プロファイルが異なります。

方法: OLAP は列ベースのストレージと並列実行によって最適化されます。OLTP は行ベースのストレージとトランザクション保証を使用します。

使用理由: ワークロードの種類ごとに適切なアーキテクチャを選択するため。

[高度なクエリ最適化](#) でこれがどのように使用されるかを確認してください。

ETL/ELT

何：

- ETL: 抽出、変換、ロード
- ELT: 抽出、ロード、変換 (レイクまたはウェアハウス内)

理由: ELT は、大規模な生データを処理する最新のレイクハウスの機能を活用します。

方法: データはオブジェクト ストレージにステージングされ、クエリエンジンまたはパイプラインを使用して変換されます。

使用する理由: データの取り込みを簡素化し、パイプラインの柔軟性を高めます。

Analytics Accelerator concepts でこれがどのように使用されるかを確認してください。

データ階層化

概要: アクセスパターンに基づいてデータをホット、ウォーム、コールドに分類します。

理由: すべてのデータに高速ストレージが必要なわけではありません。コールドデータを安価なストレージに移動すると、コストを節約できます。

方法: データはストレージ タイプ (高速 SSD、オブジェクト ストレージ) 間で物理的に分離されます。

使用理由: ストレージコストとパフォーマンスを最適化するため。

Analytics Accelerator concepts でこれがどのように使用されるかを確認してください。

セマンティック検索

内容: キーワードだけでなく意味を理解する検索。

理由: ユーザーはよりスマートな検索エクスペリエンスを期待しています。

方法: データはベクトル埋め込みとして表現され、類似性検索を使用して比較されます。

使用理由: AI および最新のアプリでの検索アプリケーションをサポートするため。

Analytics Accelerator concepts でこれがどのように使用されるかを確認してください。

ベクトル埋め込み

概要: 高次元空間におけるデータ (テキスト、画像、音声) の数値表現。

理由: 完全一致を超える類似性の比較を有効にします。

方法: FAISS、Pinecone などのライブラリや多くの ML モデルは埋め込みを生成して使用します。

使用理由: AI/ML の検索および取得タスクに。

Analytics Accelerator concepts でこれがどのように使用されるかを確認してください。

検索拡張生成 (RAG)

内容: LLM が回答する前に外部ソースからドキュメントを取得する AI パターン。

理由: LLM 応答がより正確になり、データに基づいたものになります。

方法: ベクトル検索は関連する埋め込みを取得し、LLM はそれをプロンプトと組み合わせます。

使用する理由: AI 駆動型アプリの事実の正確性を向上させるため。

Intro to RAG

Analytics Accelerator concepts でこれがどのように使用されるかを確認してください。

追加の概念

複数エンジンの相互運用性

概要: 複数のクエリエンジンがレイク内の共有データにアクセスできるアーキテクチャ。

理由: 組織は柔軟性を求めています。つまり、さまざまなユースケース (SQL エンジン、データサイエンスノートブック、ストリーミングエンジン) にさまざまなツールが必要です。

方法: オープン テーブル形式 (Iceberg、Delta Lake) により、マルチエンジン アクセスが可能になります。

- Spark がデータを書き込む → Trino、Postgres Lakehouse、その他のエンジンがそれをクエリできます。
- PGD はデータをオフロードします → Lakehouse クラスターと外部エンジンがそれを分析できます。

使用する場合: ベンダー ロックインのない柔軟なデータアーキテクチャを構築します。

データの鮮度とレイテンシ

内容: データがレイクに到着するとすぐに、遅延を最小限に抑えてクエリを実行できます。

理由: ビジネスユーザーは、新しいデータへのほぼリアルタイムのアクセスをますます期待するようになっています。

方法: Lakehouse アーキテクチャはこれをサポートします。

- ELT パイプラインはレイクのデータを継続的に更新できます。
- オープン テーブル形式は、アトミックコミットとトランザクションの可視性をサポートします。
- ベクトル化された Lakehouse エンジンは、更新されたデータを即座にクエリできます。

使用する場合: 完全なウェアハウス ETL サイクルを必要とせずに、運用データまたはストリーミングデータを分析します。

次に進む

- *Analytics Accelerator concepts*
- *Analytics How-To Guides*
- *Analytics Terminology*

Analytics Terminology

このページでは、Analytics Accelerator および Hybrid Manager (HM) の分析機能で使用する主要な用語を定義します。

概念、ハウツー、機能ガイドを調べるときに、クイックリファレンスとして使用します。

アパッチアイスバーク

オブジェクトストレージに保存される大規模な分析データセット向けのオープンテーブル形式。スキーマ進化、タイムトラベル、そして多くの分析エンジン（Spark、Trino、Flink、Postgres）との相互運用性を提供します。

Apache Iceberg

デルタ湖

データレイクに ACID トランザクションと信頼性を追加するオープンテーブル形式とストレージレイヤー。
_delta_log トランザクションログを備えた Parquet ファイル上に構築されています。

Delta Lake

データレイクハウス

データレイクとデータウェアハウスの要素を組み合わせた最新のアーキテクチャ:

- オブジェクトストレージにデータを保存する
- オープンテーブル形式（Iceberg、Delta Lake）をサポート
- ベクトル化されたクエリエンジンを使用して高速な SQL 分析を提供します
- ストレージをコンピューティングから分離

関連概念: [データレイクハウス](#)

EDB Postgres Lakehouse クラスター

オブジェクトストレージに保存されているデータに対して高速 SQL クエリを実行するために、Hybrid Manager によってプロビジョニングされた管理された分析コンピューティングクラスター。

- Iceberg および Delta Lake 形式をサポート
- ベクトル化されたクエリエンジン（Apache DataFusion）を使用します
- スケーラビリティとコスト効率のためにストレージとコンピューティングを分離します

EDB Postgres Lakehouse

EDB Postgres 分散 (PGD)

Postgres の高度な分散バージョン:

- 高可用性とマルチマスターレプリケーションを提供
- 高度なデータ階層化パターン（階層化テーブル）を有効にします
- スケーラブルな分析のために Lakehouse クラスターとシームレスに連携

[Learn more](#)

階層型テーブル

PGD **AutoPartition** を使用して大規模な時間ベースのデータセットを管理する機能:

- **ホットデータ** は PGD トランザクションクラスター内に残ります
- **コールドデータ** は Apache Iceberg テーブルとしてオブジェクトストレージにオフロードされます
- クエリはホットデータとコールドデータの両方に透過的にアクセスできます

EDB Postgres 分散 (PGD) と階層化テーブル

PGAA (Postgres 汎用分析アダプタ)

Postgres で次のことが可能になる拡張機能:

- オープンテーブル形式（Iceberg、Delta Lake）のクエリ
- オブジェクトストレージにマップする **外部テーブル** を定義する
- Power Lakehouse クラスタークエリ
- PGD 階層化テーブルのオフロードとクエリを有効にする

Example in use

PGFS (Postgres ファイルシステム)

Postgres がオブジェクト ストレージにアクセスするために使用できる**ストレージの場所**を定義する拡張機能。

- PGAA と PGD が氷山とデルタ湖へのアクセスに使用
- AWS S3、Google Cloud Storage、および互換性のあるサービスをサポート
- オブジェクトストレージをクエリする必要がある各 Lakehouse または PGD クラスターで構成する必要があります

Example in use

自動パーティション

PGD   BDR (双方向レプリケーション) の機能:

- PGD テーブルの時間ベースのパーティションを自動的に作成します
- `analytics_offload_period` が設定されている場合、古いパーティションをオブジェクトストレージに自動的にオフロードします。

Example in use

BDR 分析テーブル

内部 PGD コンセプト:

- `analytics_table` ビューは、どの PGD テーブルが Tiered Table オフロードの対象としてマークされているかを追跡します。
- 各テーブルの状態とオフロードの進行状況を追跡します

階層型テーブルの動作を監視および検証するのに役立ちます。

Example in use

氷山カタログ

Iceberg テーブルのスキーマ、場所、バージョンを追跡するメタデータサービス:

- 複数のエンジン (Spark、Trino、Postgres) 間での完全な相互運用性のために必要
- HM 管理 (Lakekeeper) または外部 (AWS Glue、Nessie、Polaris など) が可能
- PGAA と PGD はどちらも Iceberg REST カタログへの接続をサポートしています

Example in use

レイクキーパー

HM が管理する Iceberg カタログ サービス:

- アイスバークテーブルの中央カタログを提供します
- カタログ統合による PGD 階層化テーブルオフロードをサポート
- カタログベースのテーブル検出を通じて Lakehouse クラスタークエリをサポート
- Iceberg REST カタログ API を使用

Example in use

オープンテーブル形式

分析に使用される標準化されたファイルレイアウトとメタデータ形式:

- **Apache Iceberg**
- **デルタ湖**

同じデータへのマルチエンジン アクセスを有効にします (Spark、Trino、Postgres Lakehouse など)。

オープンテーブル形式

ベクトル化されたクエリエンジン

データを **列バッチ** で (行ごとではなく) 処理する分析エンジン:

- 分析クエリを高速化
- 最新の CPU 機能 (SIMD) を活用
- Lakehouse クラスターのクエリパフォーマンスを強化

EDB Lakehouse クラスターには、このために **Apache DataFusion** が埋め込まれています。

データ階層化

ホット、ウォーム、コールドデータを異なるストレージ層に保存する方法:

- ホット → 高速トランザクション PGD ノード
- コールド → オブジェクト ストレージ (アイスバーグ テーブル)
- 階層型テーブルは Postgres + HM でこのパターンを自動化します

関連概念: **データ階層化**

ストレージとコンピューティングの分離

Lakehouse アーキテクチャで使用されるパターン:

- **オブジェクト ストレージ** にデータを保存する (独立してスケーリング可能)
- ストレージとは独立して分析コンピューティング (Lakehouse クラスター) を拡張

コスト効率が高く、弾力性のある分析アーキテクチャを実現します。

関連概念: **ストレージとコンピューティングの分離**

次のステップ

重要な用語を理解したら、

- [Learn Analytics Concepts](#)
- [EDB Postgres Lakehouse](#)
- [EDB Postgres 分散 \(PGD\) と階層化テーブル](#)
- [See How-To Guides](#)

9.5 Analytics Accelerator learning paths

これらの学習パスを使用して、基礎概念から高度な実装や最適化手法に至るまで、段階的に知識を構築します。

9.5.1 これらの学習パスの使い方

現在の知識と目標に合ったレベルを選択してください。各レベルには、概念説明、ハウツー記事、実践的なチュートリアルを含む厳選ガイドへのリンクがあります。

高度なトピックと認定については、公式 EDB トレーニングプログラムをご覧ください。

9.6 Analytics Accelerator learning paths

対象者: 基礎知識を持ち、コアテクノロジーを実践的に適用できるユーザー。DBA、データエンジニア、開発者に適しています。

焦点: EDB Postgres Lakehouse、PGD アナリティクス（階層化テーブルを含む）、Iceberg/Delta Lake 統合の実践的な使用方法。基本的なユースケースと構成。

学習内容:

- EDB Postgres Lakehouse クラスターのセットアップとクエリ
- Apache Iceberg および Delta Lake テーブルで作業する
- EDB Postgres Distributed (PGD) を使用して基本的な階層型テーブル構成を実装する
- BI ツールを Analytics Accelerator に接続する
- BI レポート、履歴データ分析、運用分析に分析を適用する

旅を続けましょう: [Analytics Accelerator concepts](#) + [How-To Guides](#)

9.7 Analytics Accelerator learning paths

9.7.1 EDB のトレーニングと認定

専門家による指導と公式認定については、EDB のトレーニングプログラムと認定をご覧ください。

- [Analytics/Lakehouse](#) (近日公開)

- *Analytics/Lakehouse* (近日公開)

9.8 Analytics Accelerator learning paths

9.8.1 次のステップ

- *Analytics Accelerator generic concepts* で基礎を築く
- *Analytics Accelerator concepts* で EDB のアプローチを学ぶ
- *How-To Guides* で知識を応用する
- *Analytics Accelerator use cases* で業界アプリケーションを探索

これらの学習パスを使用して、基礎知識から高度な最適化やアーキテクチャまで、Analytics Accelerator の専門知識を深めましょう。

Foundational concepts

アナリティクスアクセラレータの基盤となるコアとなる考え方を学ぶには、ここから始めましょう。このレベルは、最新のアナリティクスの概念を初めて学ぶ方、または復習したい方向けに設計されています。

学習内容

- データ ウェアハウス、データ レイク、レイクハウスアーキテクチャの基本概念
- 列指向ストレージとベクトル化クエリエンジン
- オープンテーブル形式: Apache Iceberg および Delta Lake
- EDB Analytics Accelerator のビジョンとコアアーキテクチャ
- 分析のためにストレージとコンピューティングを分離することの価値

始める前に

- *Analytics Accelerator generic concepts*
- *Analytics Accelerator concepts*

推奨学習パス

1. *Analytics Accelerator generic concepts*
 2. *Analytics Accelerator concepts*
 3. Understanding Apache Iceberg with EDB solutions (セクションリンクは後日提供)
 4. Understanding Delta Lake with EDB solutions (セクションリンクは後日提供)
- *EDB Postgres Lakehouse*

次のステップ

101 を完了したら、次の作業に進みます。

Practical application and core solutions

基礎知識を基に、主要な Analytics Accelerator テクノロジーを環境に適用し始めます。

学習内容

- EDB Postgres Lakehouse クラスターの設定とクエリ方法
- Apache Iceberg と Delta Lake のテーブルの使い方
- PGD で階層化テーブルを実装する方法
- BI ツールを Analytics Accelerator に接続する方法
- これらのテクノロジーを BI レポート、履歴データ分析、運用分析に適用する方法

始める前に

- *Analytics Accelerator concepts*

推奨学習パス

1. *Create a Lakehouse cluster*
2. *Configure an Iceberg REST catalog connection*
3. *Query existing Apache Iceberg tables*
4. *Query Delta Lake tables*
5. *Configure PGD node group for analytics offload*
6. *Configure BDR AutoPartition with analytics offload*
7. *Query Tiered Tables from PGD and Lakehouse*
8. *Connect BI tools to the Lakehouse (必要に応じて)*

次のステップ

201 を完了したら、次の手順に従ってください。



Advanced techniques and optimization

高度な最適化とアーキテクチャパターンに関する専門知識を深めたい経験豊富なユーザー、アーキテクト、エンジニア向けです。

学習内容

- EDB Postgres Lakehouse、PGAA、PGFS の高度な構成
- ベクトル化されたクエリとオブジェクトストレージアクセスのパフォーマンスチューニング
- コストとパフォーマンスを最適化する複雑な階層型テーブル戦略
- 高度な Iceberg カタログの管理とサードパーティカタログの統合
- Analytics Accelerator と他のデータ フレームワーク (Spark、ストリーミング プラットフォーム) を組み合わせたソリューションの設計
- 複雑な分析クエリとデータパイプラインのトラブルシューティング
- 分析環境におけるセキュリティのベストプラクティス

始める前に

- [Analytics Accelerator concepts](#)

推奨学習パス

1. [Performance tuning for Delta Lake queries](#)
2. [Monitor Tiered Tables status and storage savings](#)
3. [Advanced Iceberg catalog management](#) (セクションリンクは後日提供)
4. [Architectural patterns for real-time vs. batch analytics](#) (セクションリンクは後日提供)
5. [Analytics use cases and industry solutions](#)
6. [Analytics Accelerator for your role](#)
7. [Security best practices for analytics environments](#) (セクションリンクは後日提供)

次のステップ

公式のトレーニングと認定を検討してください。

- [EDB Training portal](#)

9.9 a persona-based guide

このガイドでは、EDB Postgres® AI と分析機能を特定の役割に合わせて活用する方法を解説します。EDB Postgres AI は、幅広い分析ニーズに対応する強力なプラットフォームを提供します。このガイドは、目標達成に必要な機能に集中するのに役立ちます。

9.9.1 役割を素早く見つける

- データベース管理者 (DBA)
- DevOps エンジニア / サイト信頼性エンジニア (SRE)

- データサイエンティスト/データアナリスト
- アプリケーション開発者

これらのソリューションを EDB Hybrid Manager (HM) 内で実装および管理する方法の詳細については、[Analytics Accelerator for your role: a persona-based guide](#) を参照してください。

9.9.2 データベース管理者 (DBA)

データベースインフラストラクチャのパフォーマンス、信頼性、コスト効率、管理性に重点を置きます。

主な目標と Analytics Accelerator ソリューション

パフォーマンスの監視とチューニング

遅いクエリ、ボトルネック、非効率的なリソース使用を特定します。HM ダッシュボードを使用して、EDB Postgres Lakehouse ノードと PGD クラスターのパフォーマンスを監視します。 [Analytics Accelerator for your role: a persona-based guide](#)。

ストレージ管理とコストの最適化

大規模データベースと履歴データをコスト効率よく管理します。PGD と EDB Postgres Lakehouse の階層化テーブルを使用して、コールドデータをオブジェクトストレージ (Iceberg テーブルとして) にオフロードします。 [Analytics Accelerator for your role: a persona-based guide](#)。HM を介して階層化テーブルを管理します。

キャパシティプランニング

ストレージとコンピューティングのニーズを予測します。Lakehouse と HM ダッシュボードで、過去の成長パターンと現在の使用率を分析します。Lakehouse のコンピューティングとストレージの分離を活用して、柔軟なスケーリングを実現します。

分析データのバックアップと復元

オブジェクトストレージ内のデータと関連メタデータを保護します。Iceberg カタログのバックアップ戦略と HM バックアップオプションに従ってください。 [Analytics Accelerator for your role: a persona-based guide](#)。

分析のためのデータアクセスの確保

データアナリストとデータサイエンティストに、パフォーマンスに優れ安全なアクセスを提供します。Lakehouse ノードと PGD オフロードを使用することで、トランザクションシステムへの影響を最小限に抑えます。

推奨リソース

- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide

9.9.3 DevOps エンジニア / サイト信頼性エンジニア (SRE)

自動化、インフラストラクチャ管理、スケーラビリティ、信頼性、および可観測性に重点を置きます。

主な目標と Analytics Accelerator ソリューション

自動プロビジョニングと管理

Hybrid Manager を使用して、コンソール、API、または CLI 経由で Lakehouse クラスターと関連リソースをプロビジョニングします。 Analytics Accelerator for your role: a persona-based guide 。

スケーラビリティと弾力性

需要に応じて Lakehouse のコンピューティング リソースを個別に拡張します。

監視と観測可能性

分析システムのパフォーマンス、健全性、コストを可視化します。HM ダッシュボードとログ機能を活用します。 Analytics Accelerator for your role: a persona-based guide 。

データ パイプラインと分析アプリケーションのための CI/CD

スキーマの変更 (Iceberg の進化) と PGD 経由のデータ オフロードを CI/CD パイプラインに統合します。

クラウドリソースのコスト管理

オブジェクトストレージと分析コンピューティングのコストを追跡・最適化します。HM リソースインサイトを活用します。

推奨リソース

- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide

9.9.4 データサイエンティスト/データアナリスト

洞察の抽出、モデルの構築、ビジネス上の質問への回答に重点を置きます。

主な目標と Analytics Accelerator ソリューション

大規模で多様なデータセットへのアクセス

Lakehouse ノードを介して、Iceberg または Delta Lake テーブルとして保存された大規模データセットをクエリできます。Python、R、SQL クライアント、BI ツールを使用できます。

データの準備と探索

Lakehouse ノードを介して SQL を使用してデータのフィルタリング、集計、変換を実行します。

オープンテーブル形式の操作

Iceberg テーブルと Delta Lake テーブルでスキーマの進化、タイムトラベル、ACID トランザクションを活用します。 Analytics Accelerator for your role: a persona-based guide 。 Analytics Accelerator for your role: a persona-based guide 。

データサイエンスツールおよびフレームワークとの統合

Lakehouse クエリの結果を pandas、R データフレームに読み込むか、Spark、TensorFlow、PyTorch で使用します。

クエリパフォーマンス

Lakehouse ノードのベクトル化されたクエリエンジンを使用して分析 SQL クエリを高速化します。

推奨リソース

- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide

9.9.5 アプリケーション開発者

データ駆動型アプリケーションの構築と洞察の埋め込みに重点を置きます。

主な目標と Analytics Accelerator ソリューション

分析機能を備えたアプリケーションの構築

Lakehouse ノードまたは PGD 読み取りレプリカを使用して、アプリケーションでレポート、ダッシュボード、データ探索を提供します。

大規模なレポートクエリを効率的に処理する

大規模な集計および履歴データクエリを Lakehouse ノードにオフロードします。階層化テーブルを使用して、履歴クエリを分析層に転送します。

統合プラットフォームからデータにアクセスする

運用データと分析データの両方に一貫した SQL アクセスを提供します。

AI/ML 対応アプリケーションの開発

Lakehouse ノードを使用して ML モデル データを保存およびアクセスします。

推奨リソース

- [Analytics Accelerator for your role: a persona-based guide](#)
- [Analytics Accelerator for your role: a persona-based guide](#)
- [Analytics Accelerator for your role: a persona-based guide](#)

より詳細なガイダンスについては、以下のリソースをご覧ください。HM 固有の実装手順については、[Analytics Accelerator for your role: a persona-based guide](#) を参照してください。

9.10 Analytics Accelerator tutorials

このセクションでは、Analytics Accelerator を使用して実用的なソリューションを構築するための詳細なステップバイステップのチュートリアルについて説明します。

チュートリアルは学習指向で、有意義な成果を達成するためのガイドとなります。多くの場合、複数の機能や概念を組み合わせ、より大きな問題を解決する方法を解説します。

特定のタスクに関する簡潔で目標指向的な手順については、[How-To guides](#) を参照してください。Hybrid Manager (HM) 固有のチュートリアルについては、[HM analytics tutorials](#) を参照してください。

9.10.1 チュートリアルのカテゴリ

はじめにと基礎プロジェクト

EDB の高度な分析機能を初めて使用するユーザーや基礎プロジェクトを構築するユーザー向けのチュートリアル。

チュートリアル

**** Analytics/Lakehouse ** レベル:** 初心者 **推定所要時間:** 約 45 分 **使用製品:** Postgres Lakehouse、PGAA、PGFS **最終結果:** Postgres Lakehouse ノードを立ち上げ、Postgres からオブジェクトストレージをクエリします。 **関連概念:** [データレイクハウス](#)、[EDB Postgres Lakehouse](#)

**** Analytics/Lakehouse ** レベル:** 初心者 **推定所要時間:** 約 1 時間 **使用製品:** Postgres Lakehouse、PGFS、Delta Lake **最終結果:** 生データをオブジェクト ストレージにロードし、Lakehouse テーブルとしてクエリを実行します。 **関連概念:** [ストレージとコンピューティングの分離](#)

**** Analytics/Lakehouse ** レベル:** 中級 **所要時間:** 約 1.5 時間 **使用製品:** PGAA、PGFS、PGD、Postgres Lakehouse **最終結果:** 複数のデータソースを統合された分析ビューに結合します。 **関連概念:** [Analytics Accelerator concepts](#)

エンドツーエンドの EDB Postgres Lakehouse 実装

取り込み、保存、クエリなど、Lakehouse のワークフロー全体を実証するプロジェクト。

チュートリアル

**** Analytics/Lakehouse ** レベル:** 中級 **所要時間:** 約 2 時間 **使用製品:** Postgres Lakehouse、PGAA、Iceberg REST カタログ、BI ツール **最終結果:** 分析パイプラインを構築し、BI ツールで Lakehouse データを視覚化します。 **関連概念:** [オープンテーブル形式](#)

**** Analytics/Lakehouse ** レベル:** 中級 **所要時間:** 約 2～3 時間 **使用製品:** Postgres Lakehouse、PGFS、Delta Lake **最終結果:** Delta 形式を使用して履歴データセットを移行および分析します。 **関連概念:** [EDB Postgres Lakehouse](#)

**** Analytics/Lakehouse ** レベル:** 上級 **所要時間:** 約 2～3 時間 **使用製品:** Postgres Lakehouse、PGAA、BI ツール (Tableau、PowerBI) **最終結果:** Lakehouse データに基づいてインタラクティブな分析ダッシュボードを構築します。 **関連概念:** [Analytics Accelerator concepts](#)

分析と階層型ストレージのための高度な PGD

階層化、オフロード、ハイブリッド分析アーキテクチャに PGD を使用します。

チュートリアル

**** Analytics/Lakehouse ** レベル:** 上級 **所要時間:** 約 3 時間 **使用製品:** PGD、AutoPartition、Iceberg カタログ、Postgres Lakehouse **最終結果:** 自動階層化を構成し、パーティションを Iceberg にオフロードします。 **関連概念:** [データ階層化](#)

**** Analytics/Lakehouse ** レベル:** 上級 **所要時間:** 約 2～3 時間 **使用製品:** PGD、PGAA、Postgres Lakehouse **最終結果:** 階層型ストレージ層全体のクエリをチューニングおよびテストします。 **関連概念:** [Analytics Accelerator concepts](#)

**** Analytics/Lakehouse ** レベル:** 上級 **所要時間:** 約 2 時間 **使用製品:** PGD、AutoPartition、Postgres Lakehouse **最終結果:** 階層化テーブルのデータ保持およびライフサイクル管理ポリシーを実装します。 **関連概念:** [Analytics Accelerator concepts](#)

Gen AI Builder と分析機能を使用した AI 搭載アプリケーションの構築

Gen AI Builder + Analytics Accelerator を併用します。

AI/ML の詳細な概念については、[AI Factory concepts](#) を参照してください。

チュートリアル

**** Analytics/Lakehouse ** レベル:** 中級 **推定所要時間:** 約 1 時間**使用製品:** Gen AI Builder、Griptape、Postgres Lakehouse **最終結果:** ユーザーの問い合わせを分析データにルーティングする Griptape 構造を構築します。
関連概念: [Analytics Accelerator concepts](#)

**** Analytics/Lakehouse ** レベル:** 中級 **推定所要時間:** 約 1 時間**使用製品:** Gen AI Builder、Griptape、Postgres Lakehouse **最終結果:** Lakehouse データから口座残高を照会する Griptape ツールを構築します。 **関連概念:** [Analytics Accelerator concepts](#)

**** Analytics/Lakehouse ** レベル:** 上級 **所要時間:** 約 2 時間**使用製品:** Gen AI Builder、Griptape、Postgres Lakehouse、PGAA **最終結果:** オンデマンドで Lakehouse データをクエリできる AI アシスタントを実装します。 **関連概念:** [Analytics Accelerator concepts](#)

業界ソリューションのワークスルー

Analytics Accelerator を使用して実際のビジネス問題を解決するためのエンドツーエンドのチュートリアル。

チュートリアル

**** Analytics/Lakehouse ** レベル:** 上級 **所要時間:** 約 2~3 時間**使用製品:** PGD、PGAA、Lakehouse、ストリーミング エンジン (Kafka、Flink) **最終結果:** Lakehouse + ストリーミングを使用して、リアルタイムの不正検出パイプラインを構築します。 **関連概念:** [Analytics Accelerator concepts](#)

**** Analytics/Lakehouse ** レベル:** 中級 **所要時間:** 約 2 時間**使用製品:** PGD、AutoPartition、Lakehouse、BI ツール **最終結果:** 階層型テーブルを使用してレポート データ マートアーキテクチャを実装します。 **関連概念:** [OLAP / OLTP](#)

このインデックスは、チュートリアルが追加されるにつれて大きくなります。Hybrid Manager (HM) 固有のチュートリアルについては、[HM analytics tutorials](#) を参照してください。