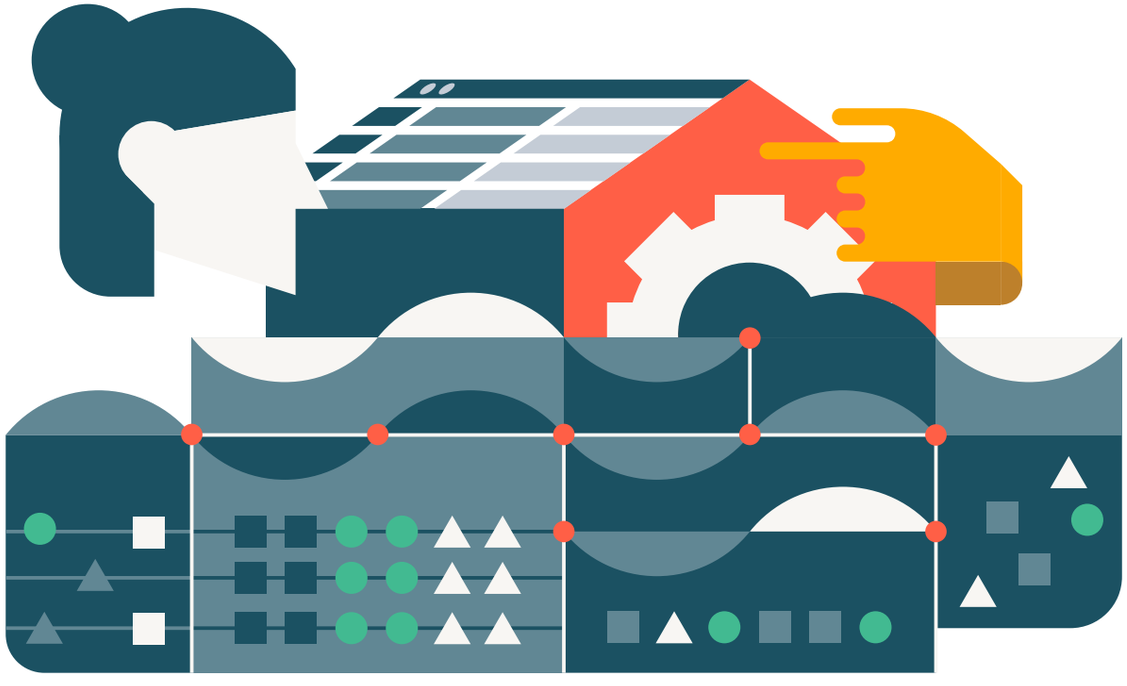
databricks

eBook

データウェアハウスと BI のビッグブック

目次

データは企業の成功に不可欠な要素です。データを活用する組織が増えるにつれて、効率的なデータ管理とガバナンスの維持がますます重要になっています。Databricks のデータインテリジェンスプラットフォームは、効果的なデータ管理とデータ活用、データと AI のアクセスを容易にし、これらの課題を解決します。データレイクとデータウェアハウスの両方の長所を統合したレイクハウスアーキテクチャを基盤として構築されており、コストの削減を可能にし、データ・AI イニシアチブを促進させます。また、ETL、SQL、機械学習、BI のための多目的クエリエンジンと、データと AI の統合ガバナンスを提供します。



データインテリジェンスプラットフォームの概要

Databricks データインテリジェンスプラットフォームは、あらゆるデータと AI の効果的な管理、活用、アクセスを可能にします。データレイクとデータウェアハウスの長所を取り入れたレイクハウスアーキテクチャを基盤に構築されており、データと AI 全体を包括的にサポートする統合ガバナンスレイヤーと、ETL、SQL、AI、BI をサポートする単一の統合クエリエンジンを備えており、コストの削減とデータ・AI イニシアチブの迅速化を支援します。

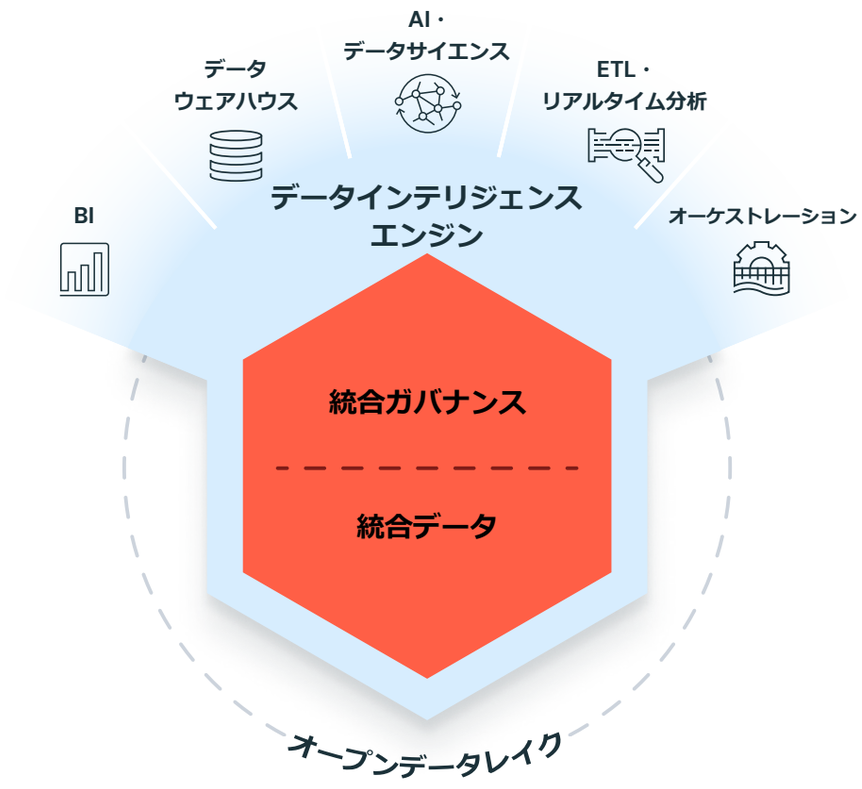
データインテリジェンスプラットフォームは、生成 AI とレイクハウスの統合の利点を組み合わせ、エンタープライズデータ独自のセマンティクスを理解する DatabricksIQ と呼ばれるデータインテリジェンスエンジンを提供します。DatabricksIQ は、データの中身、メタデータ、使用パターン（クエリ、レポート、リネージなど）を含む、データのあらゆる側面を自動的に分析します。この包括的な分析により、データプラットフォームの継続的な学習、機能の改善と新機能の追加が可能になり、データ管理と AI アプリケーションを最適化できます。

Databricks データインテリジェンスプラットフォームは、データの深い理解を通じて、次のような機能を提供します。

- **自然言語へのアクセス：** データインテリジェンスプラットフォームは、AI モデルを活用し、各組織の専門用語や略語を含む自然言語を使用したデータ操作を可能にします。既存のワークロードでデータがどのように使用されているかを観察して組織の用語を学習し、専門家でないユーザーからデータエンジニアまで、カスタマイズされた自然言語インターフェースをあらゆるユーザーに提供します。
- **セマンティックカタログとディスカバリ：** 生成 AI は、各組織のデータモデル、メトリクス、KPI を理解し、比類のないディスカバリ機能を提供します。また、データの使用方法における不整合を自動的に特定します。
- **管理と最適化の自動化：** AI モデルは、データの使用状況に基づいて、データの配置、パーティショニング、インデックスを最適化し、チューニングやノブ設定の手間を軽減します。

ガバナンスとプライバシーの強化：データインテリジェンスプラットフォームは、自然言語を使用して管理を簡素化し、秘密データの悪用を自動的に検知、分類、防止します。

AI ワークロードのためのファーストクラスのサポート：データインテリジェンスプラットフォームは、関連するビジネスデータに接続し、データプラットフォームが学習したセマンティクス（メトリクス、KPI など）を活用して正確な結果を提供し、エンタープライズ AI アプリケーションを強化します。AI アプリケーションの開発者は、プロンプトエンジニアリングによる煩雑な作業によるインテリジェンス構築を回避できます。



Databricksプラットフォームは、エンタープライズAIアプリケーションの開発を簡素化します。エンタープライズは、このプラットフォームにより、データを理解する AI アプリケーションを容易に構築できます。エンタープライズデータを AI システムに直接統合するために Databricks プラットフォームが提供する主な機能は次のとおりです。

- **エンドツーエンドの検索拡張生成（RAG）：**カスタムデータに基づいた高品質の対話エージェントを構築。
- **カスタムモデルのトレーニング：**組織のデータを使用してゼロからカスタムモデルをトレーニングするか、MPT や Llama 2 などの既存モデルを継続的に事前トレーニングし、対象領域に特化したインサイトで AI アプリケーションをさらに強化。
- **統合ガバナンスと品質監視機能：**エンタープライズデータを効率的かつ安全にサーバーレスで推論。
- **エンドツーエンドの MLOps：**実績豊富なオープンソースプロジェクト MLflow をベースとしたエンドツーエンドの MLOps が実用的なデータを生成し、レイクハウスで自動的に追跡・監視。

本リファレンスガイドでは、Databricks データインテリジェンスプラットフォームにおけるデータウェアハウスのベストプラクティスを、エンドツーエンドの実際のユースケースを通じて解説します。あらゆる規模の企業が、データの取り込みから処理、AI、LLM、分析、BI までの過程で、SQL を使用して未加工データを実用的なデータに変換するのをデータプラットフォームがどのように支援するかを解説します。データインテリジェンスプラットフォームにおけるデータライフサイクルのあらゆる側面を探求できるよう、リファレンスアーキテクチャやコードサンプルも掲載しています。

Databricks SQL について詳しくは、eBook「データレイクハウスが次のデータウェアハウスになる理由」をご参照ください。

セクション

02

インテリジェンスと自動化をベースとしたデータウェアハウス

databricks

レイクハウスを基盤とするデータ・AI イニシアチブが成功するか否かは、シンプルなデータアーキテクチャ、データ品質とガバナンス、拡張性、性能などの要素にかかっています。これらの要素は、データ戦略の基盤となり、モダンなデータ管理と分析の目標達成を促進させます。

シンプルなデータアーキテクチャ

データレイクハウスアーキテクチャは、データウェアハウスの機能を提供し、同時にデータのサイロ化を解消します。このアプローチでは、最初の段階で、データをクラウドベースのデータレイクに集約します。**Delta Lake** を基盤とし、単一のデータソースで分析と AI のユースケースを運用できるようになり、ストレージ費用の削減とデータエンジニアリングの効率化を実現します。レイクハウスアーキテクチャは、統合ガバナンスとセキュリティ構造を **Unity Catalog** と統合し、高粒度のデータ制御と迅速なデータアクセスを確保します。

レイクハウスアーキテクチャは、さまざまな分析や AI ワークロードにおけるデータのライフサイクル全体、変換、影響を網羅する包括的なアプローチです。全てのワークロードで同じデータを共有し、一貫したセキュリティおよびガバナンスポリシーを遵守できるようになり、データに信頼性をもたらします。組織内のサイロが減少し、シームレスなコラボレーションが可能になり、データ製品の提供における生産性が向上します。

シンプルなデータアーキテクチャによるその他のメリットを下記に挙げます。

- **データと AI の容易な運用 :** オープンソースと標準に基づいて構築されたレイクハウスは、データのサイロを取り除き、データランドスケープが簡素化される。
- **単一のプラットフォーム :** 一元化されたプラットフォームで、統合、ストレージ、処理、ガバナンス、共有、分析、AI に対応できる。構造化データと非構造化データの両方を扱う統合アプローチ、データのリネージや起源の完全なビュー、Python と SQL、ノートブック、IDE、バッチ、ストリーミング、および主要なクラウドプロバイダ向けに統合されたツールセットを利用できる。
- **最適な TCO の実現 :** パフォーマンスとストレージの自動最適化で最適な TCO を実現し、大規模言語モデル（LLM）などの高度な処理を含むデータウェアハウスと AI のパフォーマンスベンチマークを設定できる。

データ品質とガバナンス

データは組織の根幹をなすものであり、特にその量と多様性が増すにつれて、厳格な品質とガバナンスが不可欠です。レイクハウスの潜在能力を最大限に活かすには、正確性、信頼性、コンプライアンスを優先する必要があります。低いデータ品質はインサイトを歪め、脆弱なガバナンスは規制やセキュリティの問題を引き起こす可能性があります。Databricks データインテリジェンスプラットフォームは Unity Catalog を備えており、これらの問題に対処し、データのライフサイクル全体にわたってデータの品質を管理および改善するための統合フレームワークを提供します。

レイクハウスアーキテクチャのガバナンスの特長は次のとおりです。

- 主要なクラウド、さまざまなプラットフォームで、データと AI 資産を容易に発見、分類、統合。自然言語を使用したデータ探索とインサイトの抽出を単一のアクセスポイントから実行可能。
- 統合インターフェースによるシンプルなアクセス管理。高粒度の制御とスケーラブルなローコードポリシーにより、クラウドおよびプラットフォーム全体で一貫性のあるセキュアなアクセスを実現。
- AI の活用によるデータ・ML モデルの自動監視、プロアクティブなアラートの発行、デバッグの効率化、ビルトインのシステムテーブルを介したレイクハウスのオペレーションの包括的な把握。
- Unity Catalog でオープンソースの Delta Sharing を使用することで、クラウド、リージョン、プラットフォーム間でのデータと AI 資産の効率的な共有が可能。複雑なプロセスや高コストのレプリケーションなしで、セキュアなコラボレーションと価値創造を促進。

拡張性と性能

レイクハウスアーキテクチャは、ストレージから独立した場所でコンピューティング機能を実行し、分散させることで、データ量の増大に対応し、一貫したパフォーマンスを最適なコストで提供することに焦点を当てています。Databricks データインテリジェンスプラットフォームは、耐障害性を重視して設計されており、組織は必要に応じてデータ運用の規模を拡張できます。

Databricks プラットフォームは次のようなさまざまな拡張性を提供します。

サーバーレス

Databricks プラットフォームは、サーバーレスのクラウドベースのコンピューティングリソースを利用し、必要なコンピューティング能力に応じてワークロードの柔軟な調整と拡張を可能にします。このような動的なリソース割り当てにより、需要のピーク時の迅速なデータ処理と分析が保証されます。

同時並行性

サーバーレスコンピューティングの活用と AI による最適化により、データ処理とクエリの容易な同時実行を実現します。複数のユーザーやチームがパフォーマンスの制限を受けることなく、分析タスクを同時に実行できるようになります。

ストレージ

データレイクとシームレスに統合し、データの可用性と信頼性を確保しながら、膨大なデータ量のコスト効率に優れたストレージを実現します。また、データストレージを最適化してパフォーマンスを向上させ、ストレージ費用を削減します。

スケーラビリティは不可欠ですが、それを補完するのがパフォーマンスです。Databricks データインテリジェンスプラットフォームは、AI によるさまざまな最適化を提供し、スケーラビリティとパフォーマンスの両方を実現します。

クエリ処理を最適化

機械学習の最適化技術を活用してクエリの実行を高速化します。自動インデックス作成、キャッシング、述語プッシュダウンを活用してクエリを効率的に処理し、インサイトの取得を迅速化します。

オートスケーリング

Databricks プラットフォームは、ワークロードに合わせてサーバーレスリソースをインテリジェントにスケールします。最適なクエリパフォーマンスの維持と、使用したコンピューティングに対してのみの料金の支払いを保証します。

Photon

Databricks プラットフォームの新しいネイティブな超並列処理（MPP）エンジンは、極めて高速なクエリパフォーマンスを低コストで提供します。データの取り込みから、ETL、ストリーミング、データサイエンス、インタラクティブなクエリに至るまで、データレイク上で直接実行します。Photon は Apache Spark™ API と互換性があり、コードの変更は不要で、ロックインはありません。

Delta Lake

Unity Catalog と Photon を搭載した Delta Lake は、手動での調整は不要で、すぐに最高の価格性能を提供します。Databricks プラットフォームは、AI モデルを使用してデータストレージの一般的な課題を解決します。時間の経過とともに変化するテーブルを手動で管理することなく、高速なパフォーマンスを得ることができます。

- 更新のための予測 I/O は、クエリプランとデータレイアウトを最適化してピークパフォーマンスを実現し、読み取りと書き込みのパフォーマンスをバランスよく管理します。書き込み時のコピーと読み取り時のマージなどの戦略の選択は自動化され、データからより多くの価値を創出できます。
- リキッドクラスタリングにより、高カーディナリティのカラムのパーティション分割の困難さや、パーティションカラムを変更する際の高コストな書き換えなどの問題を回避し、適切に調整、パーティショニングされたテーブルのパフォーマンスを得ることができます。その結果、最小限の構成で高速で適切にクラスタリングされたテーブルが得られます。
- 予測最適化は、顧客データを自動的に最適化し、最高の価格性能を実現します。顧客のデータ使用パターンから学習し、適切な最適化プランを構築し、そのプランをハイパーオプティマイズされたサーバーレスインフラ上で実行します。

レイクハウスアーキテクチャの基盤を確立した後は、Databricks でデータウェアハウスおよび分析機能を提供することが重要です。Databricks SQL（DB SQL）によって促進される適切なデータ構造と管理機能を詳しく説明します。

Databricks SQL サーバーレス：
レイクハウスアーキテクチャに最適なデータウェアハウス

Databricks SQL は、データウェアハウス機能を強化し、Databricksデータインテリジェンスプラットフォーム上で優れた SQL サポートを提供するために導入されました。SQL ベースのデータ変換、探索、分析を効率化し、さまざまな技術的背景を持つユーザーに対応します。BI アナリスト、データアーキテクト、分析エンジニアは、直感的な SQL インターフェースにより、専門的なプログラミングを必要とせずに、クエリや複雑なデータ操作を容易に行うことができます。このように、多様なユーザーがデータにアクセスできるようにすることで、組織全体がデータ駆動の文化を受け入れ、データに基づいた意思決定を行うことが重視されるようになります。

Databricks SQL は、膨大なデータセットを高速かつ効率的に処理できる能力において優れています。Databricksの次世代エンジンPhotonと、AIによる最適化を活用することで、迅速なデータ処理と分析が可能になり、クエリの実行時間を大幅に短縮できます。データに関する課題に直面する組織にとって、パフォーマンスの高さは極めて重要です。高いパフォーマンスにより、さまざまなデータセットからインサイトを取得できます。さらに、Databricks SQL はコラボレーションを促進し、クエリ、結果、理解をデータ専門家が即座に共有できるワークスペースを提供します。このような環境の共有は、知識の交換の促進や問題解決の迅速化につながり、組織はチームの集合知を活用できるようになります。

さらに、Databricks SQL には、データガバナンス、セキュリティ、コンプライアンスに関する高度な規定が組み込まれており、組織はデータ品質の維持、アクセス制限の実施、データ活動の監視、秘密データの保護、規制基準の遵守を行うことができます。まとめると、Databricks SQL は次の機能を提供します。

- **インサイト取得を迅速化：**一般的な英語を使用してデータにアクセスして、自動的に SQL クエリを作成できます。これにより、クエリの改良が迅速になり、エンタープライズ内の誰もが利用できるようになります。
- **優れた価格性能：**AI に最適化された処理と組み合わされたサーバーレスコンピューティングにより、クラウドインフラ管理の必要とせずに、業界トップクラスのパフォーマンスとスケーラビリティを低コストで実現します。
- **統合ガバナンス：**データおよび AI 資産の所在を問わず、統合ガバナンスレイヤーを確立します。
- **複雑さの軽減：**SQL と Python、ノートブックと IDE、バッチとストリーミング、主要なクラウドプロバイダをサポートする 1 つのプラットフォームで、あらゆるデータ、分析、AI を統合します。
- **充実したエコシステム：**SQL、Power BI、Tableau、dbt、Fivetranなどの任意のツールを Databricks で利用し、BI、データ取り込み、データ変換を実行できます。

まとめ

Databricks データインテリジェンスプラットフォームは、データウェアハウジングと分析の領域における革新的な進化を具現化したものです。昨今のデータ主導型ビジネスに求められるデータウェアハウスワークロードの効率化のニーズに応えます。シンプルなデータアーキテクチャによって、充実したエンドツーエンドのデータウェアハウス機能を提供し、データの集約と一元管理を可能にし、コストを削減し、生のデータからの実用的なインサイト抽出を迅速化し、バッチとストリーミングを統合します。さらに、Unity Catalog をによってデータ品質とガバナンスを確保し、複数のクラウド環境を包括的にサポートするデータリネージによる高粒度のガバナンスで、あらゆるデータの容易な発見・保護・管理を可能にします。

拡張性と性能は、Databricks プラットフォームの主要な差別化要素であり、サーバーレスコンピューティング、同時実行のサポート、最適化されたストレージ戦略によって実現しています。AI の活用によるクエリ処理の高度化、性能の向上、データの最適化による価格性能の向上を通じて、迅速で費用対効果の高いデータ分析を可能にします。

Databricks SQL は、優れた SQL サポートを提供し、プラットフォームの機能を強化し、容易なデータ変換と分析を可能にします。コラボレーション、データガバナンス、充実したエコシステムによるツールの利用を促進し、データのサイロ化を解消して、組織におけるデータの最大限の活用を支援します。本 eBook では、Databricks プラットフォームでのデータウェアハウスと BI ワークロードの実行に関するユースケースもご紹介します。

関連リソース

- → データレイクハウスが次のデータウェアハウスとなる理由 : 第 2 版
- → Databricks SQL の新機能のご紹介
- → Unity Catalog にレイクハウスフェデレーション機能を導入
- → AI Functions のご紹介 : 大規模言語モデルを Databricks SQL で統合する

セクション

03

レイクハウスにおけるデータウェアハウジングのベストプラクティス

セクション 3.1

データウェアハウスのモデリング技法とレイクハウスへの実装

レイクハウスでの Data Vaults と Star Schemas の利用について

執筆者 : Soham Bhatt、Deepak Sekar

レイクハウスは、データレイクとデータウェアハウスの長所を組み合わせた、新しいデータプラットフォームパラダイムです。多くのユースケースやデータプロダクトを格納できる、大規模なエンタープライズレベルのデータプラットフォームとして設計されています。データレイクとデータウェアハウスを統合した、単一のエンタープライズデータリポジトリとして使用することができます。

- データドメイン
- リアルタイムストリーミングのユースケース
- データマート
- 異種データウェアハウス
- データサイエンス機能ストア、データサイエンスサンドボックス
- 部門別のセルフサービス型分析サンドボックス

ユースケースの多様性を考えると、レイクハウスのプロジェクトによって異なるデータ整理の原則やモデリングテクニックが適用されるかもしれません。技術的には、レイクハウスアーキテクチャは、多くの異なるデータモデリング形式をサポートすることができます。この記事では、レイクハウスのブロンズ／シルバー／ゴールドデータ編成原則の実装と、異なるデータモデリング技術が各レイヤーにどのようにフィットするかを説明することを目的としています。

Data Vault とは

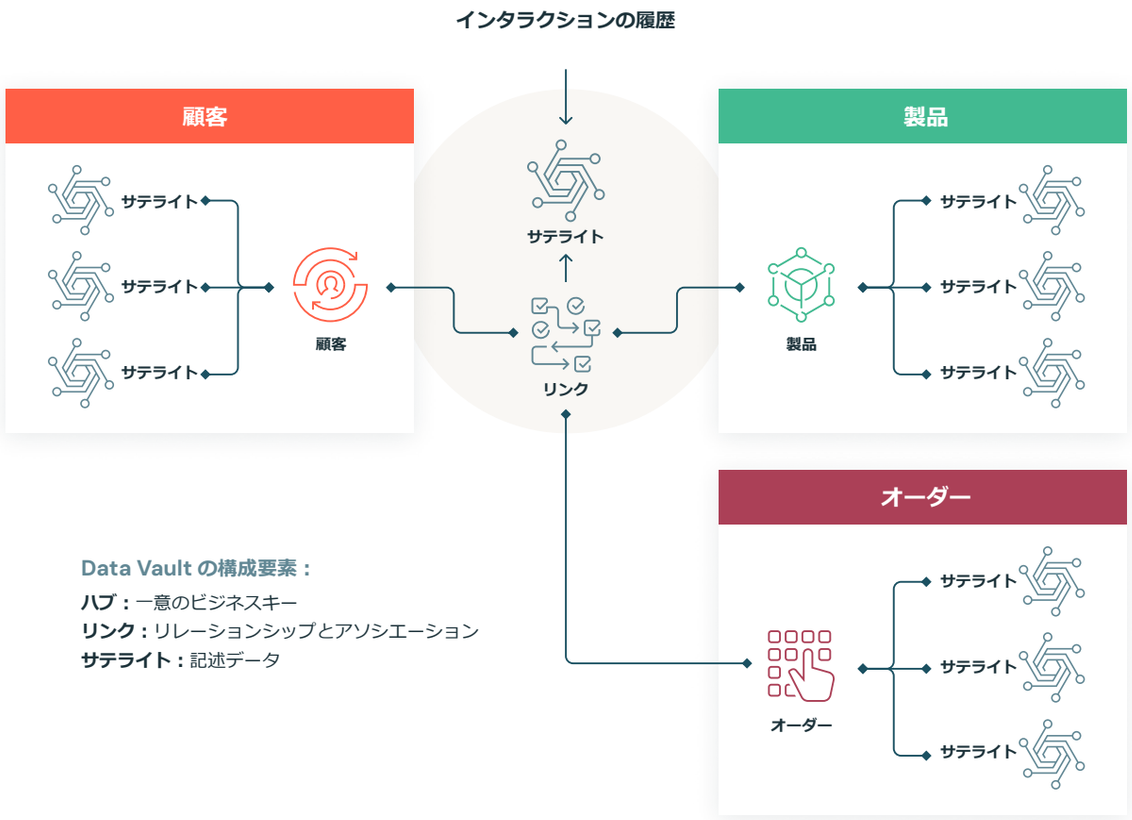
Data Vault は、Kimball や Inmon の手法に比べ、企業規模の分析用データウェアハウスを構築するために用いられる、より新しいデータモデリングのデザインパターンです。

Data Vault は、データをハブ、リンク、サテライトの 3 つのタイプに整理しています。ハブはコアのビジネスエンティティを表し、リンクはハブ間の関係を表し、サテライトはハブやリンクに関する属性を格納します。

Data Vault は、拡張性、データ統合 /ETL、開発スピードが重要視されるアジャイルデータウェアハウス開発に重点を置いています。ほとんどのお客さまは、ランディングゾーン、Vault ゾーン、データマートゾーンを、Databricks の組織パラダイムのブロンズ、シルバー、ゴールドレイヤーに対応させています。Data Vault のハブ、リンク、サテライトテーブルのモデリングスタイルは、通常レイクハウスアーキテクチャのシルバーレイヤーに適合します。

Data Vault のモデリングについては、Data Vault Alliance で詳しく説明しています。

Data Vault のモデリング



Data Vault の構成要素：

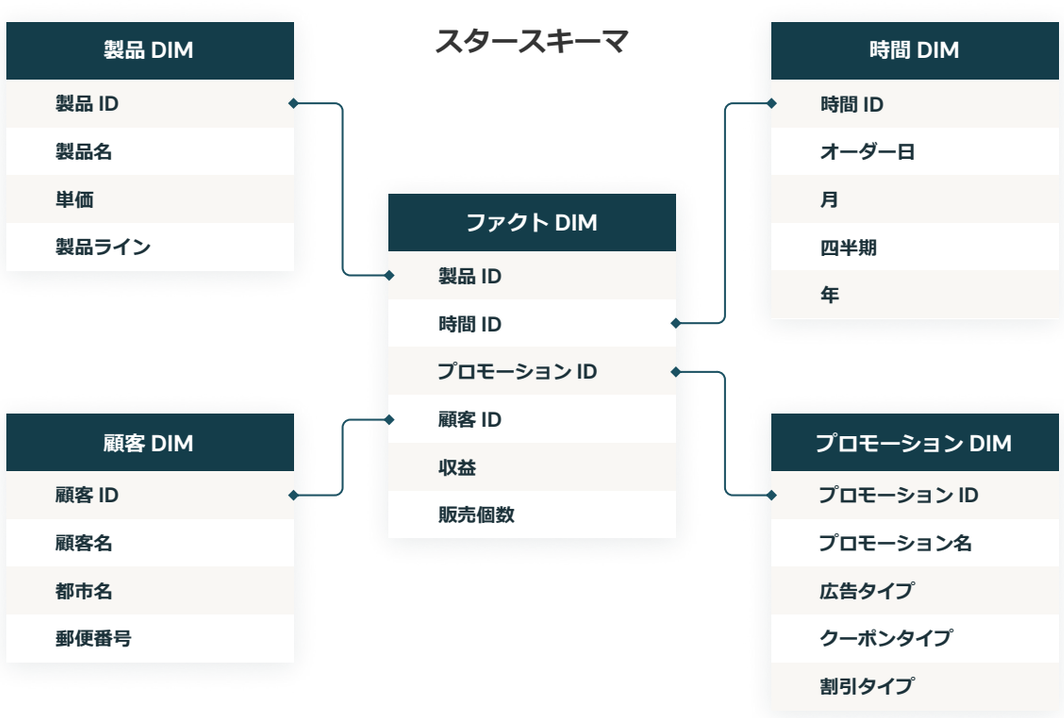
ハブ：一意のビジネスキー
リンク：リレーションシップとアソシエーション
サテライト：記述データ

Data Vault モデリングの仕組み：ハブ、リンク、サテライトが互いに接続されている。

ディメンションモデリングとは

次元モデリングは、データウェアハウスを分析用に最適化するために設計するボトムアップのアプローチです。次元モデルは、ビジネスデータを次元（時間や商品など）とファクト（金額や数量の取引など）に非正規化し、異なる対象領域を適合した次元で接続して、異なるファクトテーブルにナビゲートするために使用されます。

次元モデリングの最も一般的な形式は、スタースキーマです。スタースキーマは多次元データモデルで、データを整理して理解しやすく、分析しやすく、またレポートの実行が非常に簡単で直感的にできるようにするために使用されます。Kimball スタイルのスタースキーマや次元モデルは、データウェアハウスやデータマートのプレゼンテーション層、さらにはセマンティック層やレポート層におけるゴールドスタンダードと言えます。スタースキーマの設計は、大規模なデータセットに対するクエリに最適化されています。

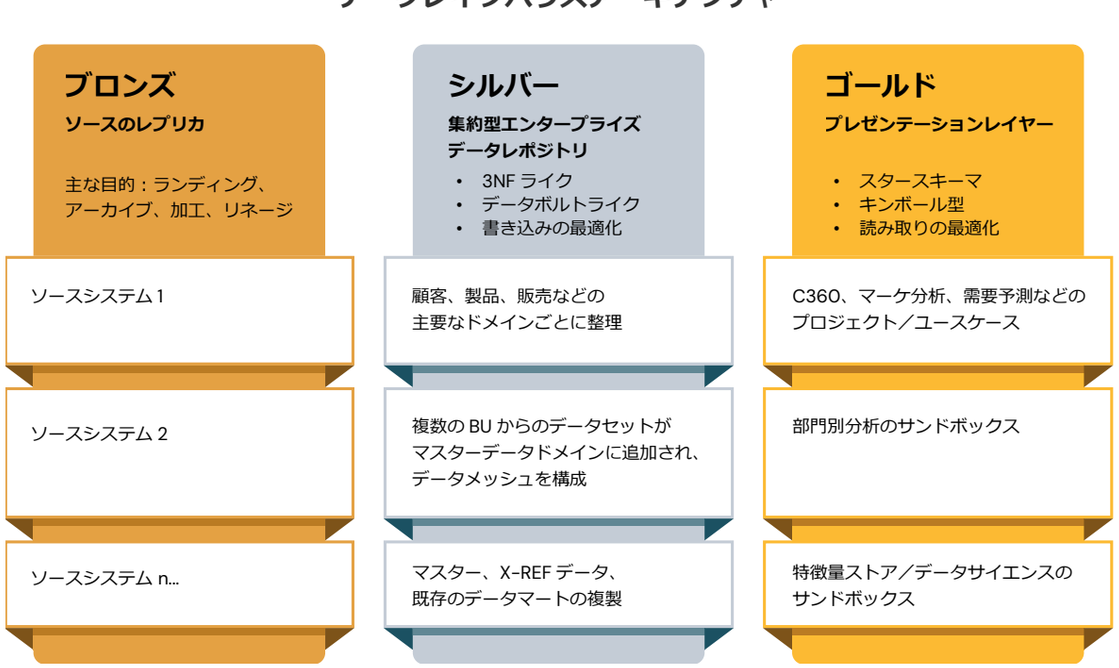


スタースキーマの例

Databricks データインテリジェンスプラットフォームでは、正規化 Data Vault（書き込み最適化）と非正規化ディメンジョンモデル（読み込み最適化）の両方のデータモデリングスタイルが採用されています。シルバーレイヤーの Data Vault のハブおよびサテライトは、スタースキーマのディメンジョンをロードするために使用され、Data Vault のリンクテーブルは、ディメンションモデルのファクトテーブルをロードするためのキードライビングテーブルとなります。Kimball Group のディメンジョンモデリングについて詳しくは、こちらを参照してください。

レイクハウスの各レイヤーにおけるデータ編成の原則

最新のレイクハウスは、全てを網羅したエンタープライズレベルのデータプラットフォームです。ETL、BI、データサイエンス、ストリーミングなど、さまざまなデータモデリングアプローチを必要とするあらゆるユースケースに対応する高い拡張性と性能を備えています。典型的なレイクハウスがどのように構成されているかを見てみましょう。



データレイクハウスアーキテクチャの「ブロンズ」、「シルバー」、「ゴールド」レイヤーの概要

ブロンズレイヤー：ランディングゾーン

ブロンズレイヤーは、ソースシステムから全てのデータを取り込む場所です。このレイヤーのテーブル構造は、ロード日時、プロセス ID などを取得するために追加できるオプションのメタデータカラムを除けば、ソースシステムのテーブル構造に「そのまま」対応します。このレイヤーの焦点は、変更データの取得（CDC）と、ソースデータの履歴アーカイブ（コールドストレージ）、データリネージ、監査可能性、および必要に応じて再処理を提供する機能（ソースシステムからデータを再読込せずに）にあります。

多くの場合、ブロンズレイヤーのデータを Delta フォーマットにしておくと、その後の ETL のためのブロンズレイヤーからの読み込みが効率的になり、ブロンズで CDC の変更を書き込むための更新ができるようになります。JSON や XML 形式のデータが届くと、元のソースデータのフォーマットでランディングし、Delta フォーマットに変更してステージングするお客さまを時々見かけます。そのため、論理的なブロンズレイヤーを物理的なランディング・ステージングゾーンにするお客さまもいらっしゃいます。

ランディングゾーンにオリジナルのソースデータフォーマットで生データを保存することは、ネイティブシンクとして Delta をサポートしていないインジェストツールを介してデータを取り込む場合、またはソースシステムがオブジェクトストアに直接データをダンプする場合の一貫性を保つためにも役立ちます。このパターンは、ソースが raw ファイルのランディングゾーンにデータを取り込み、Databricks オートローダがデータを Delta フォーマットのステージングレイヤーに変換するという、オートローダ取り込みフレームワークともうまく連携しています。

シルバーレイヤー：エンタープライズセントラルリポジトリ

レイクハウスアーキテクチャのシルバーレイヤーでは、ブロンズレイヤーからのデータが照合、マージ、適合、クリーニングされ、シルバーレイヤーが全ての主要なビジネスエンティティ、概念、トランザクションの「エンタープライズビュー」を提供できるようにします。これは、エンタープライズ運用データストア（ODS）、セントラルリポジトリ、データメッシュのデータドメイン（マスター顧客、製品、重複のないトランザクション、相互参照テーブルなど）に似ています。このエンタープライズビューは、異なるソースからのデータをまとめ、アドホックレポート、高度な分析、ML のためのセルフサービス分析を可能にします。また、部門アナリスト、データエンジニア、データサイエンティストがさらにデータプロジェクトを作成し、ゴールドレイヤーの企業および部門データプロジェクトを通じてビジネス上の問題に答えるための分析を行うためのソースとしても機能します。

レイクハウスデータエンジニアリングのパラダイムでは、従来の Extract-Transform-Load（ETL）に対して、Extract-Load-Transform（ELT）メソドロジーが採用されています。ELT アプローチとは、シルバーレイヤーのロード時に最小限の、あるいは「必要十分な」変換とデータクレンジングルールのみが適用されることを意味します。プロジェクト固有の変換ルールがゴールドレイヤーで適用されるのに対して、「エンタープライズレベル」のルールは全てシルバーレイヤーで適用されます。レイクハウスにデータを取り込み、配信するためのスピードと俊敏性が優先されます。

データモデリングの観点からは、シルバーレイヤーはより 3rd-Normal Form に近いデータモデルを持ちます。Data Vault のような書き込み可能なデータアーキテクチャとデータモデルをこのレイヤーで使用することができます。Data Vault の手法を使用する場合、生の Data Vault と Business Vault の両方がレイクの論理的なシルバーレイヤーに収まり、ポイントインタイム（PIT）プレゼンテーションビューまたはマテリアライズドビューはゴールドレイヤーに表示されることになります。

ゴールドレイヤー：プレゼンテーションレイヤー

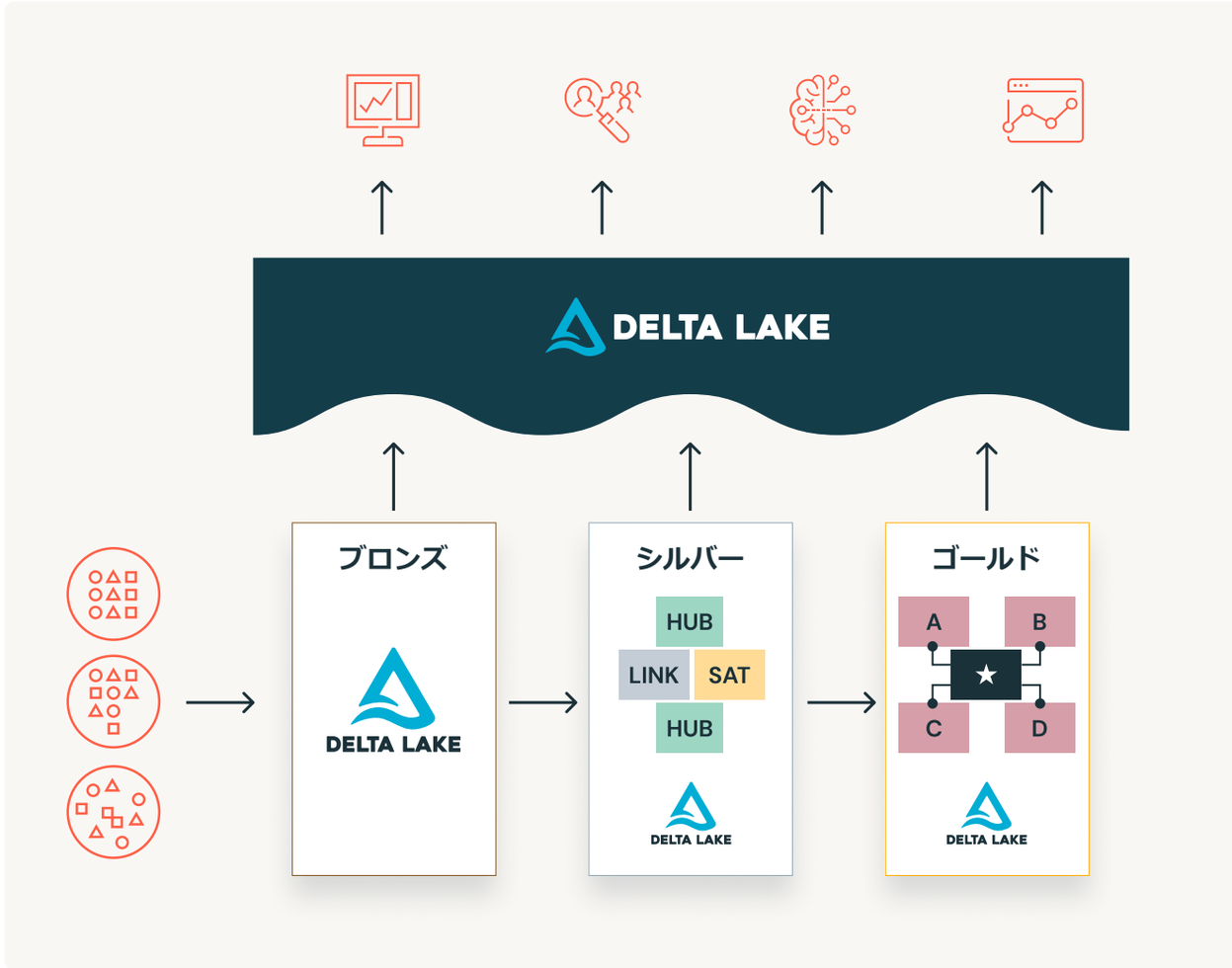
ゴールドレイヤーでは、ディメンションモデリングや Kimball 手法に従って、複数のデータマートやウェアハウスを構築することができます。先に述べたように、ゴールドレイヤーはレポーティング用であり、シルバーレイヤーと比較して結合を減らし、より非正規化、読み取り最適化されたデータモデルを使用します。ゴールドレイヤーのテーブルを完全に非正規化することも可能で、通常はデータサイエンティストが特徴抽出のアルゴリズムに利用するためにそのようにします。

シルバーレイヤーからゴールドレイヤーへのデータ変換の際には、「プロジェクト固有」の ETL とデータ品質のルールが適用されます。データウェアハウス、データマート、あるいは顧客分析、製品 / 品質分析、在庫分析、顧客セグメンテーション、製品推奨、マーケティング / 販売分析などのデータプロダクトなどの最終的なプレゼンテーション層は、このレイヤーで提供されます。キンブル式のスタースキーマ型データモデルやインモン式のデータマートは、このレイクハウスのゴールドレイヤーに適合します。セルフサービス分析のためのデータサイエンスラボラトリーや部門別サンドボックスも、このゴールドレイヤーに属します。

レイクハウスのデータ整理のパラダイム

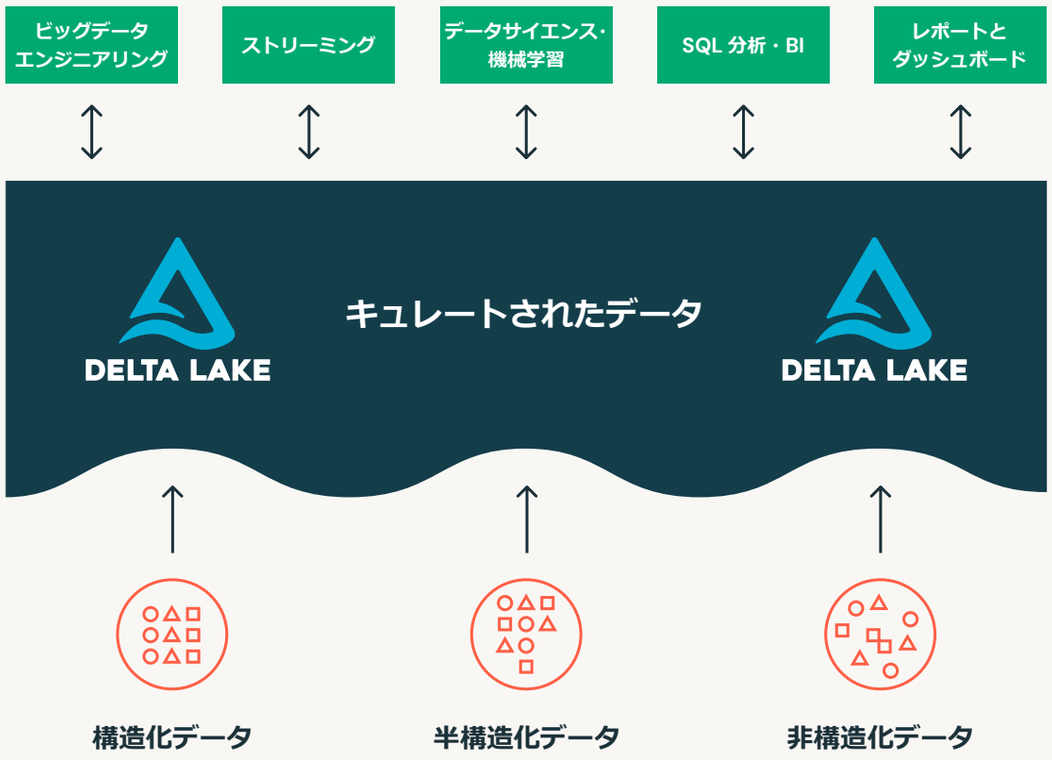
要約すると、データはレイクハウスのさまざまなレイヤーを通過する際にキュレーションされるということです。

- ブロンズレイヤーは、ソースシステムのデータモデルを使用します。もしデータが生のフォーマットで着地した場合、このレイヤー内で DeltaLake フォーマットに変換されます。
- シルバーレイヤーは、異なるソースからのデータをまとめ、エンタープライズビューを作成するために適合させます。通常、より正規化され、書き込みが最適化されたデータモデルを使用します。
- ゴールドレイヤーは、シルバーレイヤーよりも非正規化またはフラット化されたデータモデルを持つプレゼンテーションレイヤーで、一般的にはキンボール式のディメンションモデルやスタースキーマが使用されます。ゴールドレイヤーには、企業全体でセルフサービス分析やデータサイエンスを実現するための部門別サンドボックスやデータサイエンスサンドボックスも配置されます。これらのサンドボックスと独立した計算クラスタを提供することで、ビジネスチームがレイクハウス外でデータのコピーを独自に作成することを防ぎます。



このレイクハウスのデータ組織のアプローチは、データのサイロを壊し、チームをまとめ、適切なガバナンスのもと、1 つのプラットフォームで ETL、ストリーミング、BI や AI を行う権限を与えることを意図しています。中央データチームは、データモデリングプロセスがボトルネックになるのではなく、新しいセルフサービスユーザーのオンボーディングや、多くのデータプロジェクトの開発を並行してスピードアップし、組織内のイノベーションを実現する存在であるべきです。Databricks Unity Catalog は、レイクハウス上で検索と発見、ガバナンス、リネージを提供し、データガバナンスを確実に実行することができます。

Databricks SQL で Data Vaults とスタースキーマデータウェアハウスを今すぐ構築しましょう。→



レイクハウスアーキテクチャの各レイヤーでデータがキュレートされる。

関連リソース

- → Delta Lake で Databricks にスタースキーマを実装するための簡単な 5 ステップ
- → Databricks データインテリジェンスプラットフォームで Data Vault モデルを実装するためのベストプラクティス
- → モデリングのベストプラクティスと現代レイクハウスでの実装
- → サロゲートキーを生成する ID 列がレイクハウスで利用可能に
- → Databricks データインテリジェンスプラットフォームで EDW ディメンションモデルをリアルタイムにロード

セクション 3.2

ディメンションモデリングのベストプラクティスとモダンレイクハウスアーキテクチャでの実装

執筆者 : Leo Mao、Abhishek Dey、Justin Breese、Soham Bhatt

Databricks データインテリジェンスプラットフォームは、データウェアハウスをモダナイズするだけでなく、成熟したストリーミングプラットフォームと高度なアナリティクスプラットフォームへの即座のアクセスを可能にするため、多くのお客さまがレガシーデータウェアハウスを Databricks データインテリジェンスプラットフォームに移行しています。Databricks プラットフォームは、ストリーミング、ETL、BI、AI のあらゆるニーズに対応し、単一のプラットフォームでビジネスチームとデータチームがコラボレーションできるようにします。

お客さまを現場で支援するなかで、Databricks での適切なデータモデリングや物理データモデルの実装に関するのベストプラクティスを多くのお客さまが必要としていることがわかりました。

この記事では、Databricks データインテリジェンスプラットフォームにおけるディメンションモデリングのベストプラクティスを深く掘り下げ、テーブル作成と DDL のベストプラクティスを使用した物理データモデルの実装例を紹介します。

このブログで取り上げるハイレベルなトピックは次のとおりです。

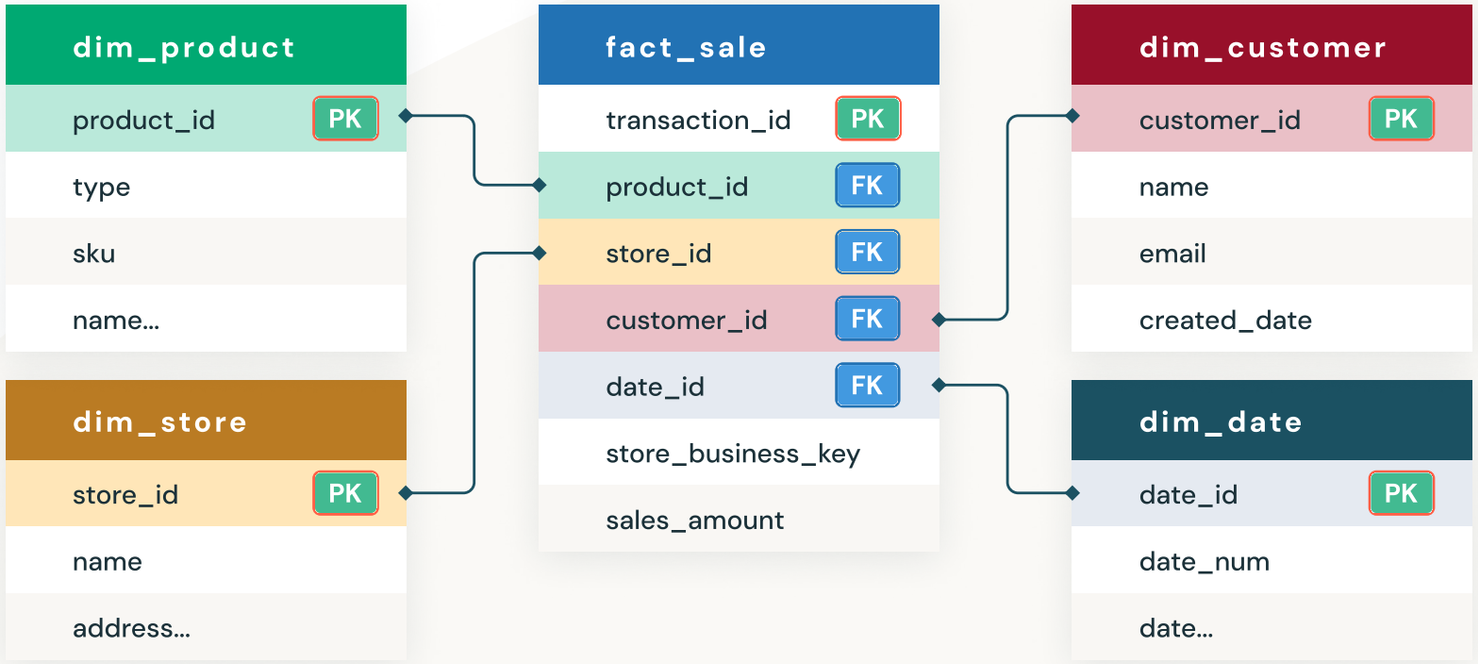
- データモデリングの重要性
- 一般的なデータモデリング技法
- データウェアハウスのモデリング DDL の実装
- Databricks におけるデータモデリングのベストプラクティスと提言

データウェアハウスにおけるデータモデリングの重要性

データウェアハウスの構築では、データモデルが最前線に位置します。通常、このプロセスは、意味論的ビジネス情報モデルを定義することから始まり、次に論理データモデル、そして最後に物理データモデル（PDM）を定義します。全ては適切なシステム分析と設計の段階から始まります。このフェーズでは、まずビジネス情報モデルとプロセスフローが作成され、組織内のビジネスプロセスに基づいて主要なビジネスエンティティ、属性、およびそれらの相互作用がキャプチャされます。次に、エンティティが互いにどのように関連しているかを示す論理データモデルが作成されます。これは技術にとらわれないモデルです。最後に、書き込みと読み取りが効率的に実行できるように、基礎となる技術プラットフォームに基づいて PDM が作成されます。データウェアハウスでは、スタースキーマや Data Vault（データボルト）のような、分析に適したモデリングスタイルが一般的です。

Databricks で物理データモデルを作成するためのベストプラクティス

定義されたビジネス上の問題に基づき、データモデル設計の目的は、再利用性、柔軟性、拡張性のためにデータを簡単な方法で表現することです。以下は典型的なスタースキーマのデータモデルです。各トランザクションを保持する売上ファクトテーブルと、顧客、商品、店舗、日付などのさまざまなディメンションテーブルを示しています。ディメンションをファクトテーブルに結合することで、特定の月の人気商品や、四半期で最も好調な店舗など、特定のビジネス上の質問に答えることができます。Databricks で実装する方法を見てみましょう。



レイクハウスアーキテクチャのディメンションモデル

注 : 各ディメンションテーブルには、SCD タイプ 2 をサポートする __START_AT 列と __END_AT 列が含まれます。ただし、この図では省略しています。

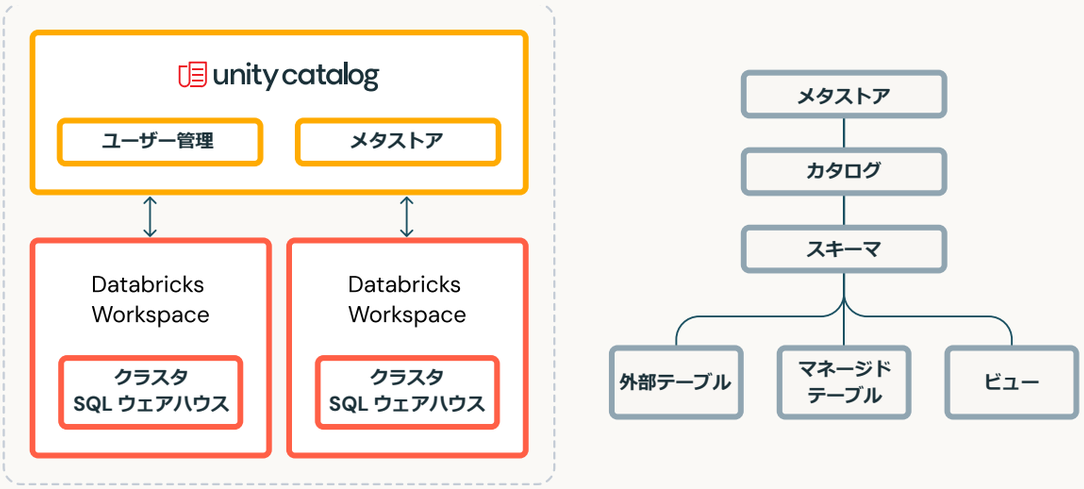
Databricks 上でのデータウェアハウスのモデリング DDL 実装

以下のセクションでは、例を用いて次のことを説明します。

- 3 レベルのカタログ、データベース、テーブルの作成
- 主キー、外部キーの定義
- 代理キーの ID 列
- データ品質のためのカラム制約
- インデックス、最適化、分析
- 高度なテクニック

1. Unity Catalog – 3 レベルの名前空間

Unity Catalog は Databricks のガバナンスレイヤーです。Databricks の管理者やデータスチュワードが 1 つのメタストアを使用して、Databricks アカウント内の全てのワークスペースにわたってユーザとそのデータへのアクセスを一元的に管理できます。異なるワークスペースのユーザーは、Unity Catalog で一元的に付与された権限に応じて、同じデータへのアクセスを共有できます。Unity Catalog には、データを整理する 3 レベルの Namespace (catalog.schema(database).table) があります。Unity Catalog について詳しくは、こちらをご覧ください。



ここでは、データベース内にテーブルを作成する前に、カタログとスキーマを設定する方法を説明します。この例では、以下のようにカタログ **US_Stores** とスキーマ（データベース）**Sales_DW** を作成し、後のセクションで使用します。

```
CREATE CATALOG IF NOT EXISTS US_Stores;
USE CATALOG US_Stores;
CREATE SCHEMA IF NOT EXISTS Sales_DW;
USE SCHEMA Sales_DW;
```

カタログとデータベースのセットアップ

以下は、**fact_sales** テーブルを 3 レベルのネームスペースでクエリする例です。

```
SELECT *
FROM US_Stores.Sales_DW.fact_sales
```

▸ (2) Spark Jobs

Table | Data Profile

	transaction_id	date_id	customer_id	product_id	store_id	store_business_key	sales_amount
1	10001	20211001	1	1	1	PER01	50
2	10004	20211003	2	1	2	BNE02	60
3	10005	20211003	3	2	1	PER01	79
4	10002	20211002	2	1	2	BNE02	79
5	10003	20211002	1	2	2	BNE02	79

Showing all 5 rows.

catalog.database.tablename を使用したテーブルのクエリの例

2. 主キー、外部キーの定義

主キー（PK）と外部キー（FK）の定義は、データモデルを作成する際に極めて重要です。PK／FK 定義をサポートする機能により、Databricks ではデータモデルの定義が非常に簡単になります。また、アナリストが Databricks SQL ウェアハウスの結合関係を素早く把握し、効率的にクエリを作成できるようになります。他の多くの超並列処理（MPP）、EDW、クラウドデータウェアハウスと同様に、PK／FK 制約は情報提供のみです。Databricks は PK／FK 関係の強制をサポートしませんが、セマンティックデータモデルの設計を容易にするために、PK／FK 関係を定義する機能を提供します。

以下は **store_id** を ID 列とし、同時に主キーとして定義した **dim_store** テーブルの作成例です。

```
-- Store dimension
CREATE OR REPLACE TABLE dim_store(
  store_id BIGINT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  business_key STRING,
  name STRING,
  email STRING,
  city STRING,
  address STRING,
  phone_number STRING,
  created_date TIMESTAMP,
  updated_date TIMESTAMP,
  start_at TIMESTAMP,
  end_at TIMESTAMP
);
```

主キー定義によるストアディメンションの作成のための DDL 実装

テーブルが作成された後、以下のテーブル定義で主キー（store_id）が制約として作成されていることがわかります。

Describe Dim Store table information

```
DESC TABLE EXTENDED US_Stores.Sales_DW.dim_store
```

Table | Data Profile

	col_name	data_type
20	Owner	leo.mao@databricks.com
21	Is_managed_location	true
22	Table Properties	[delta.minReaderVersion=1,delta.minWriterVersion=6]
23		
24	# Constraints	
25	dim_store_pk	PRIMARY KEY (`store_id`)

Showing all 25 rows.

主キー storestore_id がテーブル制約として表示されます。

以下は、**transaction_id** を主キーとし、ディメンションテーブルを参照する前キーを持つ **fact_sales** テーブルを作成する例です。

```
-- Fact Sales
CREATE OR REPLACE TABLE fact_sales(
  transaction_id BIGINT PRIMARY KEY,
  date_id BIGINT NOT NULL CONSTRAINT dim_date_fk FOREIGN KEY REFERENCES dim_date,
  customer_id BIGINT NOT NULL CONSTRAINT dim_customer_fk FOREIGN KEY REFERENCES
dim_customer,
  product_id BIGINT NOT NULL CONSTRAINT dim_product_fk FOREIGN KEY REFERENCES
dim_product,
  store_id BIGINT NOT NULL CONSTRAINT dim_store_fk FOREIGN KEY REFERENCES dim_
store,
  store_business_key STRING,
  sales_amount DOUBLE
);
```

外部キー定義で売上ファクトを作成するための DDL 実装

ファクトテーブルが作成されると、以下のテーブル定義で制約として主キー **(transaction_id)** と外部キーが作成されていることがわかります。

Describe Fact Sales Table Info

```
DESC TABLE EXTENDED US_Stores.Sales_DW.fact_sales
```

Table | Data Profile

	col_name	data_type
20	# Constraints	
21	dim_product_fk	FOREIGN KEY (`product_id`) REFERENCES `us_stores`.`sales_dw`.`dim_product` (`product_id`)
22	dim_date_fk	FOREIGN KEY (`date_id`) REFERENCES `us_stores`.`sales_dw`.`dim_date` (`date_id`)
23	dim_customer_fk	FOREIGN KEY (`customer_id`) REFERENCES `us_stores`.`sales_dw`.`dim_customer` (`customer_id`)
24	dim_store_fk	FOREIGN KEY (`store_id`) REFERENCES `us_stores`.`sales_dw`.`dim_store` (`store_id`)
25	fact_sales_pk	PRIMARY KEY (`transaction_id`)

Showing all 25 rows.

ディメンションを参照する主キーと前キーを持つファクトテーブルの定義

3. 代理キー（サロゲートキー）の ID 列

ID 列とは、データベースのカラムで、新しいデータ行ごとに一意の ID 番号を自動的に生成するものです。これらは一般的にデータウェアハウスでサロゲートキーを作成するために使用されます。サロゲートキーとは、システムが生成する無意味なキーのことで、行の一意性を識別するためにさまざまな自然主キーや複数のフィールドの連結に頼る必要がなくなります。通常、これらの代用キーは、データウェアハウスで主キーや外部キーとして使用されます。ID 列の詳細については、このブログで説明します。以下は、customer_id 列を作成する例です。自動的に割り当てられる値は 1 から始まり、1 ずつ増加します。

```sql
-- Customer dimension
CREATE OR REPLACE TABLE dim_customer(
  customer_id BIGINT GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1)
PRIMARY KEY,
  name STRING,
  email STRING,
  address STRING,
  created_date TIMESTAMP,
  updated_date TIMESTAMP,
  start_at TIMESTAMP,
  end_at TIMESTAMP
);
```

ID 列を持つお客さまディメンションを作成するための DDL 実装

4. データ品質のためのカラム制約

主キーおよび外部キーの情報制約に加えて、Databricks はテーブルに追加されるデータの品質と整合性を確保するために適用されるカラムレベルのデータ品質チェック制約もサポートしています。制約は自動的に検証されます。良い例は、NOT NULL 制約とカラム値制約です。他のクラウドデータウェアハウスとは異なり、Databricks はカラム値チェック制約を提供しています。これらは与えられたカラムのデータ品質を保証するのに非常に便利です。以下のように、**valid_ sales_amount** チェック制約は、テーブルに追加する前に、全ての既存の行が制約を満たしているか（**sales_amount** > 0）を検証します。詳細はこちらを参照してください。

以下は、store_id と sales_amount が有効な値であることを確認するために、それぞれ dim_store と fact_sales に制約を追加する例です。

```
-- Add constraint to dim_store to make sure column store_id is between 1 and 9998
ALTER TABLE US_Stores.Sales_DW.dim_store ADD CONSTRAINT valid_store_id CHECK
(store_id > 0 and store_id < 9999);

-- Add constraint to fact_sales to make sure column sales_amount has a valid value
ALTER TABLE US_Stores.Sales_DW.fact_sales ADD CONSTRAINT valid_sales_amount CHECK
(sales_amount > 0);
```

既存のテーブルに列制約を追加してデータ品質を確保

5. インデックス、最適化、分析

伝統的なデータベースには B-tree（B 木）やビットマップインデックスがありますが、Databricks には多次元 Z オーダークラスタ化インデックスという高度なインデックスがあり、ブルームフィルターインデックスもサポートしています。第一に、Delta ファイルフォーマットは、列圧縮ファイルフォーマットである Parquet ファイルフォーマットを使用しているため、列のプルーニングが非常に効率的であり、そのうえ、Z 次インデックスを使用することで、ペタバイトスケールのデータを数秒で選別できます。Z オーダーとブルームフィルターインデックスは、大規模な Delta テーブルに対する選択性の高いクエリに答えるためにスキャンする必要のあるデータ量を大幅に削減します。これは通常、実行時間の大幅な改善とコスト削減につながります。最も頻度の高い結合に使用される主キーと外部キーに Z オーダーを使用します。また、必要に応じて追加のブルームフィルターインデックスを使用します。

```
-- Optimise fact_sales table by customer_id and product_id for better query and
join performance
OPTIMIZE US_Stores.Sales_DW.fact_sales
ZORDER BY (customer_id, product_id);
```

fact_sales を customer_id と product_id で最適化し、パフォーマンスを向上させます。

```
-- Create a bloomfilter index to enable data skipping on store_business_key
CREATE BLOOMFILTER INDEX
ON TABLE US_Stores.Sales_DW.fact_sales
FOR COLUMNS(store_business_key)
```

ブルームフィルターインデックスを作成し、指定したカラムのデータスキップを可能にします。

また、他のデータウェアハウスと同様に、ANALYZE TABLE を使用して統計情報を更新し、クエリオプティマイザが最適なクエリプランを作成するための最適な統計情報を取得できるようにすることができます。

```
-- collect stats for all columns for better performance
ANALYZE TABLE US_Stores.Sales_DW.fact_sales COMPUTE STATISTICS FOR ALL COLUMNS;
```

より良いクエリ実行プランのために、全てのカラムの統計情報を収集します。

6. 高度なテクニック

Databricks はテーブルパーティショニングのような高度なテクニックをサポートしていますが、これらの機能はテラバイト級の圧縮データがある場合にのみ使用するようにしてください。ほとんどの場合、OPTIMIZE インデックスと Z-ORDER インデックスが最適なファイルとデータの削減を行います。したがって、日付や月でテーブルをパーティショニングすることは推奨しません。ただし、テーブルの DDL が自動最適化と自動コンパクションに設定されていることを確認することは良い習慣です。これにより、小さなファイルに頻繁に書き込まれるデータが、より大きなカラム圧縮された Delta フォーマットにコンパクト化されます。

ビジュアルデータモデリングツールの活用をお考えですか？パートナーである Quest の erwin Data Modeler を使用すると、数回クリックするだけで、STAR-Schema、データボルトなど、Databricks のあらゆる業界データモデルのリバースエンジニアリング、作成、実装が可能になります。

Databricks ノートブックの例

Databricks プラットフォームでは、さまざまなデータモデルを簡単に設計・実装できます。上記の全ての例を完全なワークフローで見るには、こちらの例をご覧下さい。

関連ブログ：

Delta Lake で Databricks にスタースキーマを実装する 5 つの簡単なステップ →

セクション 3.3

Databricks データインテリジェンスプラットフォームによる DWH データモデルのリアルタイムの読み込み

Delta Live Tables を使用したモダンレイクハウスでのディメンションモデリングの実装

執筆者 : Leo Mao、Soham Bhatt、Abhishek Dey

ディメンションモデリングは、最新のデータウェアハウスを構築するための最も一般的なデータモデリング手法の 1 つです。エンタープライズのビジネスニーズに基づいて、ファクトとディメンションを迅速に開発できます。現場でお客さまを支援するなかで、多くのお客さまが Databricks のベストプラクティスや実装リファレンスアーキテクチャを求めていることがわかりました。

この記事では、レイクハウスアーキテクチャにおけるディメンションモデリングのベストプラクティスを深く掘り下げます。Delta Live Tables を使用して EDW のディメンションモデルをリアルタイムでロードするライブ例を提供します。

このブログで取り上げる手順の概要は次のとおりです。

- ビジネス上の問題の定義
- ディメンションモデルの設計
- ディメンションモデリングのベストプラクティスと推奨事項
- ディメンションモデルのレイクハウスアーキテクチャへの実装
- 結論

ビジネス上の問題の定義

ディメンションモデリングはビジネス指向であり、常にビジネス上の問題から始まります。ディメンションモデルは、データ資産がどのように提供され、エンドユーザーによって消費されるかを示します。そのため、解決すべきビジネス上の問題を理解し、アクセスが容易で高速なクエリをサポートできるモデルを設計する必要があります。

ビジネスマトリックスは、ディメンションモデリングの基本概念です。以下はビジネスマトリックスの例で、列は共有ディメンション、行はビジネスプロセスを表します。定義されたビジネス上の問題によって、ファクトデータと必要なディメンションの粒度が決まります。ここでの重要な考え方は、ビジネスマトリックスとその共有ディメンションまたは適合ディメンションに基づいて、簡単に追加のデータ資産を段階的に構築できるということです。

共有ディメンション →

業務プロセス ↑	日付	顧客	製品	ベンダー	プロモーション	リセラー	販売地域	従業員	アカウント	組織
インターネット販売	✓	✓	✓	✓	✓		✓			
リセラー販売	✓		✓		✓	✓	✓	✓		
総勘定元帳	✓								✓	✓
販売計画	✓		✓				✓			
インベントリ	✓		✓	✓					✓	
顧客調査	✓	✓								
カスタマーサービスの電話	✓	✓	✓					✓		

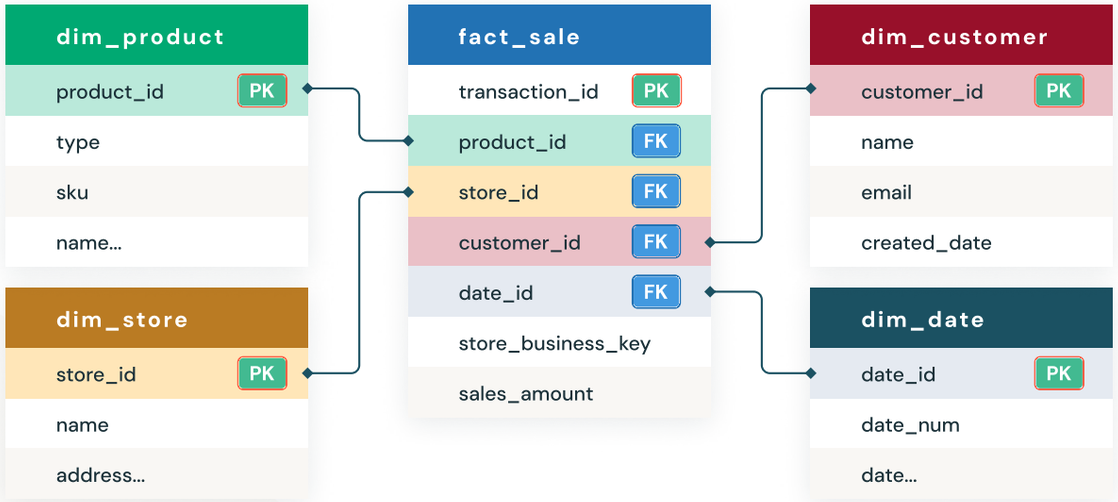
共有ディメンションとビジネスプロセスによるビジネスマトリックス

ここでは、ビジネススポンサーがチームにレポートを作成させ、次のようなインサイトの提供を依頼していると仮定します。

- 商品の人気を把握するための売れ筋商品とは？
- 良い店舗作りを学ぶために、業績の良い店舗とは？

ディメンションモデルの設計

定義されたビジネス上の問題に基づき、データモデルの設計は、再利用性、柔軟性、拡張性のためにデータを効率的に表現することを目的としています。ここでは、上記のビジネス上の問題を解決するためのハイレベルなデータモデルを示します。



レイクハウスアーキテクチャのディメンションモデル

注：各ディメンションテーブルには、SCD タイプ 2 をサポートする __START_AT 列と __END_AT 列が含まれます。ただし、この図では省略しています。

設計は、理解しやすく、データに対するさまざまなクエリパターンで効率的でなければなりません。モデルから、ビジネス上の質問に答えるために売上ファクトテーブルを設計しました。ご覧のように、ディメンションへの外部キー（FK）以外には、ビジネスを測定するために使用される数値メトリクス（sales_amount など）のみが含まれています。

また、ファクトデータのコンテキスト情報を提供する、商品、店舗、顧客、日付などのディメンションテーブルも設計しました。ディメンションテーブルは通常、ファクトテーブルと結合され、特定の月の最も人気のある製品や、四半期の最も業績の良い店舗など、特定のビジネス上の質問に答えます。

ディメンションモデリングのベストプラクティスと推奨事項

Databricks データインテリジェンスプラットフォームでは、ディメンションモデルを簡単に設計、実装できます。また、与えられたサブジェクト領域のファクトとディメンションを構築するだけです。

以下は、ディメンションモデルを実装する際に推奨されるベストプラクティスです。

- ディメンションテーブルを非正規化する必要があります。ディメンションテーブルは、第 3 正規形やスノーフレークタイプのモデルではなく、通常は高度に非正規化され、1 つのディメンションテーブル内の多対 1 のリレーションシップがフラットになります。
- 異なるディメンションテーブルの属性が同じ列名およびドメインコンテンツを持つ場合は、適合ディメンションテーブルを使用します。この利点は、各ファクトテーブルに関連付けられている適合ディメンション属性を使用して、異なるファクトテーブルからのデータを 1 つのレポートに結合できることです。
- ディメンションテーブルの一般的な傾向は、as-is または as-was レポートをサポートするために、時間の経過に伴うディメンションの変更を追跡することです。さまざまな要件に基づいてディメンションを処理するために、以下の基本的なテクニックを簡単に適用できます。
  - タイプ 1 の手法では、ディメンション属性の初期値が上書きされます。
  - SCD の最も一般的なテクニックであるタイプ 2 のテクニックでは、経時的な変化を正確に追跡するために使用します。

  これは、Delta Live Tables の実装で容易に実現できます。
  - APPLY CHANGES INTO を使用すると、Delta Live Tables を使用して SCD タイプ 1 または SCD タイプ 2 を簡単に実行できます。
- 主キーと外部キーの制約により、エンドユーザーはテーブル間の関係を理解できます。
- **ID 列を使用**することで、新しい行の追加で自動的に一意な整数値が生成されます。ID 列はサロゲートキーの一形態として機能します。詳しくはこちらのブログを参照してください。
- 強制 CHECK により、データの品質と正確性が確保されます。

ディメンションモデルのレイクハウスアーキテクチャへの実装

次に、Delta Live Tables ベースのディメンションモデリングの実装例を見てみましょう。

以下のコード例では、SCD タイプ 2 を使用してディメンションテーブル (dim_store) を作成する方法を示します。この変更データはソースシステムから取得されます。

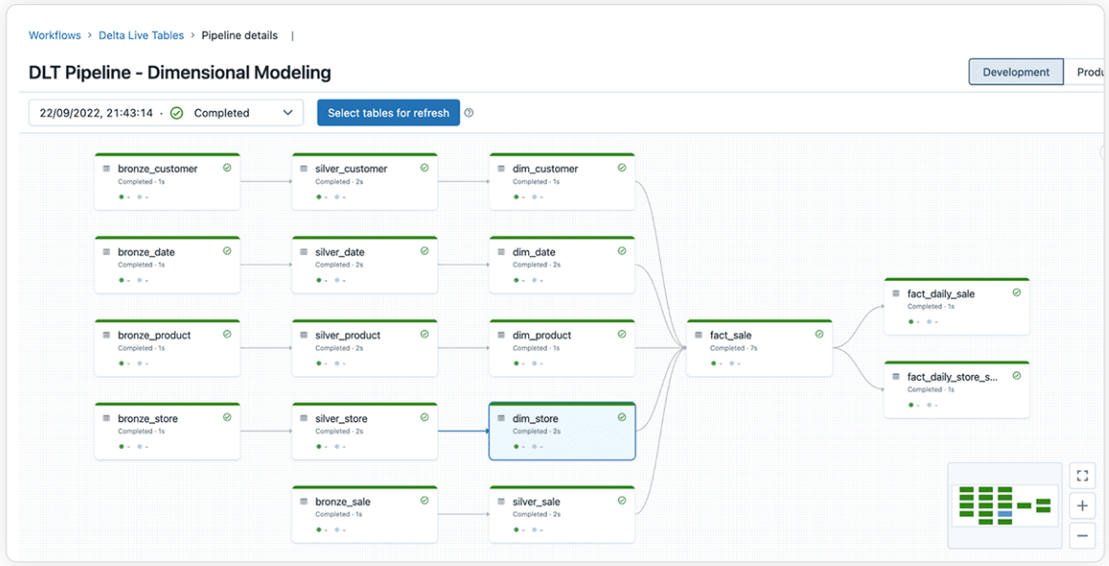
```
-- create the gold table
CREATE INCREMENTAL LIVE TABLE dim_store
TBLPROPERTIES (“quality” = “gold”)
COMMENT “Slowly Changing Dimension Type 2 for store dimension in the gold layer”;

-- store all changes as SCD2
APPLY CHANGES INTO live.dim_store
FROM STREAM(live.silver_store)
  KEYS (store_id)
  SEQUENCE BY updated_date
  COLUMNS * EXCEPT (_rescued_data, input_file_name)
  STORED AS SCD TYPE 2;
```

以下のコード例では、ファクトテーブル (fact_sale) を作成する方法を示しています。valid_product_id の制約を使用することで、ロードされる全てのファクトレコードに有効な製品が関連付けられていることを保証できます。

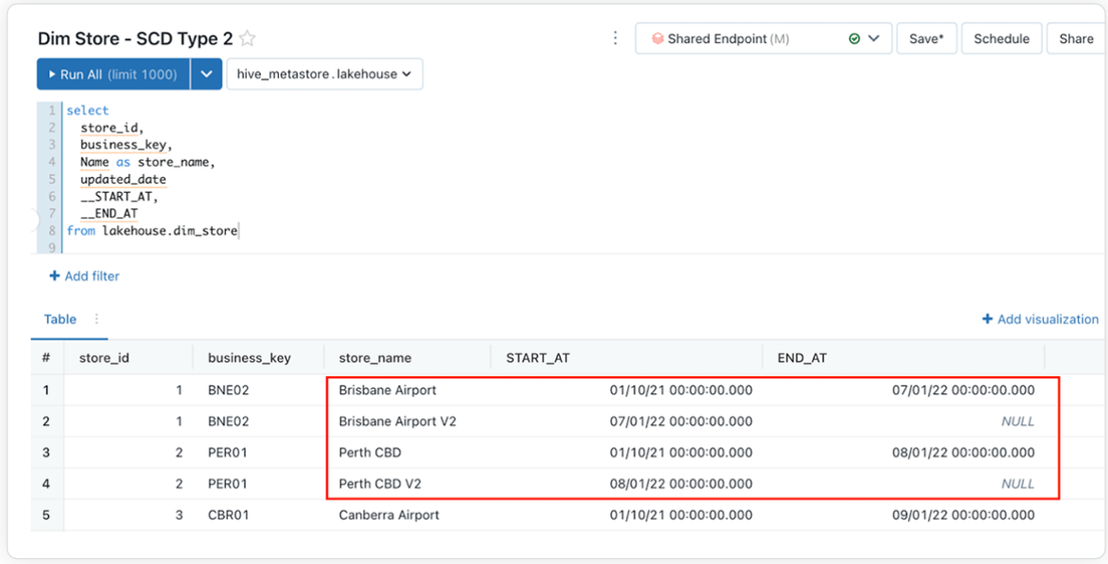
```
-- create the fact table for sales in gold layer
CREATE STREAMING LIVE TABLE fact_sale (
  CONSTRAINT valid_store_business_key EXPECT (store_business_key IS NOT NULL) ON
VIOLATION DROP ROW,
  CONSTRAINT valid_product_id EXPECT (product_id IS NOT NULL) ON VIOLATION DROP
ROW
)
TBLPROPERTIES (“quality” = “gold”, “ignoreChanges” = “true”)
COMMENT “sales fact table in the gold layer” AS
  SELECT
    sale.transaction_id,
    date.date_id,
    customer.customer_id,
    product.product_id AS product_id,
    store.store_id,
    store.business_key AS store_business_key,
    sales_amount
FROM STREAM(live.silver_sale) sale
INNER JOIN live.dim_date date
ON to_date(sale.transaction_date, ‘M/d/yy’) = to_date(date.date, ‘M/d/yyyy’)
-- only join with the active customers
INNER JOIN (SELECT * FROM live.dim_customer WHERE __END_AT IS NULL) customer
ON sale.customer_id = customer.customer_id
-- only join with the active products
INNER JOIN (SELECT * FROM live.dim_product WHERE __END_AT IS NULL) product
ON sale.product = product.SKU
-- only join with the active stores
INNER JOIN (SELECT * FROM live.dim_store WHERE __END_AT IS NULL) store
ON sale.store = store.business_key
```

Delta Live Tablesパイプラインのサンプルはこちらです。Delta Live Tablesパイプラインの作成方法については、Delta Live Tables クイックスタートを参照してください。以下のように、DLT は、レイクハウスのメダリオンアーキテクチャに従って、ブロンズ、シルバー、ゴールドレイヤー間の ETL パイプラインと異なるオブジェクト間の依存関係を完全に可視化します。



エンドツーエンドの DLT パイプライン

以下は、入力された変更に基づいてディメンションテーブル dim_store がどのように更新されるかの例です。以下では、Store Brisbane Airport が Brisbane Airport V2 に更新されています。また、既定の SCD タイプ 2 のサポートにより、元のレコードは 2022 年 1 月 7 日に終了し、同じ日に開始する新しいレコードが作成されました。このレコードは、ブリスベン空港の最新レコードを示すオープンエンド日（NULL）で開始されます。



ストアディメンション用 SCD タイプ 2

実装の詳細については、完全なノートブックのサンプルはこちらを参照してください。

結論

このブログでは、ディメンションモデリングのコンセプトの詳細、ベストプラクティス、Delta Live Tables を使用した実装方法について解説しました。

ディメンションモデリングについて詳しくは、キンボールの技術でご覧ください。

セクション 3.4

Databricks SQL の新機能のご紹介

AI による最適化、レイクハウスフェデレーションなど、エンタープライズグレードの BI を実現

執筆者 : Alex Lichen、Miranda Luna、Can Efeoglu、Cyrielle Simeone

2023 年開催の「Data+AI サミット」では、Databricks SQL がデータウェアハウスの限界に挑み続けていることをご報告しました。製品ライン全体への AI 活用を促進させ、性能と効率性における Databricks のリーダーシップをさらに確実なものにすると同時に、お客さまのエクスペリエンスを簡素化し、機会創出を支援しています。それと並行して、データスタックを Databricks データインテリジェンスプラットフォーム上のレイクハウスアーキテクチャで統合できるように、コアデータウェアハウジング機能の改善も継続しています。

このブログでは、Databricks SQL の新機能と今後の展望についてご紹介します。

- Predictive I/O のような AI 主導のパフォーマンス最適化により、手動チューニングの必要なく、優れたパフォーマンスとコスト削減を実現する。
- AI Functions、ノートブック内の SQL ウェアハウス、新しいダッシュボード、Python ユーザー定義関数による新しいユーザー体験。
- レイクハウスフェデレーションによる豊富な外部データソースのサポート。
- SQL ステートメント実行 API でデータにアクセスする新しい方法。
- ストリーミングテーブル、マテリアライズドビュー、ワークフローの統合により、シンプルで効率的なデータ処理。
- Databricks のナレッジエンジン DatabricksIQ によるインテリジェントなアシスタンス。
- Databricks SQL System Tables と Live Query Profile による管理ツールの強化。
- Fivetran、dbt labs、PowerBI とのパートナー統合のための追加機能。

AI に最適化されたウェアハウス : チューニング不要であらゆるワークロードに対応

私たちは、最高のデータウェアハウスはレイクハウスであると考えています。そのため、ETL ワークロードと AI のパワーの活用においてリーダーシップを発揮し続けています。Databricks SQL は、EDA と BI のワークロードでも業界をリードするパフォーマンスを発揮し、手動チューニングなしでコスト削減を実現します。

手動によるインデックス作成に別れを告げましょう。Predictive I/O for reads (GA) and updates (Public Preview) により、Databricks SQL は過去の読み取りと書き込みのパターンを分析し、インテリジェントにインデックスを作成してワークロードを最適化します。初期のお客さまは、ポイントルックアップ効率が 35 倍に向上し、MERGE オペレーションで 2〜6 倍、DELETE オペレーションで 2〜10 倍という驚異的なパフォーマンス向上の恩恵を受けています。

予測最適化（Public Preview）では、Databricks が OPTIMIZE、VACUUM、ANALYZE、CLUSTERING コマンドを実行することで、ファイルサイズとクラスタリングをシームレスに最適化します。この機能により、Anker Innovations はクエリパフォーマンスを 2.2 倍に高めると同時に、ストレージコストを 50% 削減しました。

> **Databricks の予測最適化により、Unity Catalog ストレージをインテリジェントに最適化し、クエリを 2 倍以上高速化しながら、年間ストレージコストを 50% 節約できました。最大かつ最もアクセス数の多いテーブルの優先順位を学習しました。また、これら全てを自動で行うため、チームの貴重な時間を節約することができました。**
>
> — ANKER INNOVATIONS

ワークロードの大小に応じて異なるウェアハウスを管理したり、スケーリングパラメータを微調整したりするのにうんざりしていませんか？インテリジェントなワークロード管理は、コストを抑えながらクエリを高速に実行するための特徴量です。インテリジェントワークロードマネージメントは、リアルタイムのパターンを分析することで、すでに実行されているクエリを中断することなく、入力された SQL ステートメントを実行するための最適なコンピュート量をワークロードに確保します。

AI を活用した最適化により、Databricks SQL はあらゆるワークロードに対して業界をリードする TCO とパフォーマンスを提供します。利用可能な最適化プレビューの詳細については、Reynold Xin の基調講演および、Data+AI サミットのブレイクアウトセッションの動画「Databricks SQL Serverless Under the Hood: How We Use ML to Get the Best Price/Performance」をご覧ください。

レイクハウスフェデレーションでサイロ化したデータを解放

現在の組織は、断片化されたシステム間でサイロ化されたデータソースを発見、管理、照会するという課題に直面しています。レイクハウスフェデレーションを使用すると、データチームは Databricks SQL を使用して、MySQL、PostgreSQL、Amazon Redshift、Snowflake、Azure SQL Database、Azure Synapse、Google の BigQuery（近日公開予定）などの外部プラットフォームのデータを検出、クエリ、管理できます。

さらに、レイクハウスフェデレーションは、Databricks 内から外部データソースにアクセスする際、Unity Catalog の高度な機能とシームレスに統合します。行および列レベルのセキュリティを適用して、機密情報へのアクセスを制限します。データリネージを活用してデータの出所を追跡し、データの品質とコンプライアンスを確保します。データ資産を整理および管理するために、連携されたカタログ資産に簡単にタグ付けして、シンプルなデータディスカバリーを実現します。

最後に、連携ソースでの複雑な変換やクロスジョインを高速化するために、レイクハウスフェデレーションはクエリのレイテンシを改善するマテリアライズドビューをサポートしています。

詳しくは、Data+AI サミットでの専用セッション「レイクハウスフェデレーション : Unity Catalog からの外部データソースへのアクセスとガバナンス」をご覧ください。

SQL ステートメント実行 API を使用したレイクハウスアーキテクチャでの開発

SQL ステートメント実行 API は、REST API を介して Databricks SQL ウェアハウスにアクセスし、クエリと結果の取得を可能にします。ほぼ全てのプログラミング言語で利用可能な HTTP フレームワークにより、多様なアプリケーションやプラットフォームから Databricks SQL ウェアハウスに簡単に直接接続できます。

Databricks SQL ステートメント実行 API は、Databricks プレミアムおよびエンタープライズ層で利用可能です。詳しくは、私たちのセッションや、チュートリアル（AWS | Azure）、ドキュメント（AWS | Azure）、コードサンプルのリポジトリを参照してください。

ワークフローのストリーミングテーブル、マテリアライズドビュー、DB SQL によるデータ処理の効率化

ストリーミングテーブル、マテリアライズドビュー、ワークフローでの DB SQL により、SQL ユーザーであれば誰でもデータエンジニアリングのベストプラクティスを適用してデータを処理できるようになりました。わずか数行の SQL で、データの取り込み、変換、オーケストレーション、分析を効率的に実行できます。

ストリーミングテーブルはデータを「ブロンズ」テーブルに取り込む理想的な方法です。SQL ステートメント 1 つで、クラウドストレージ（S3、ADLS、GCS）、メッセージバス（EventHub、Kafka、Kinesis）などさまざまなソースからスケーラブルにデータを取り込むことができます。この取り込みはインクリメンタルに行われるため、複雑なインフラを管理することなく、低レイテンシでコスト効率の高いパイプラインを実現します。

```
CREATE STREAMING TABLE web_clicks
AS
SELECT *
FROM STREAM
  read_files('s3://mybucket')
```

マテリアライズドビューは、低速のクエリや頻繁に使用される計算を事前に計算することでコストを削減し、クエリの待ち時間を改善します。データエンジニアリングの文脈では、マテリアライズドビューはデータの変換に使用されます。しかし、(1) エンドユーザーのクエリや BI ダッシュボードを高速化し、(2) データを安全に共有するために使用できるため、データウェアハウスのコンテキストにおけるアナリストチームにとっても価値があります。わずか 4 行のコードで、誰でもマテリアライズドビューを作成し、高性能なデータ処理を行うことができます。

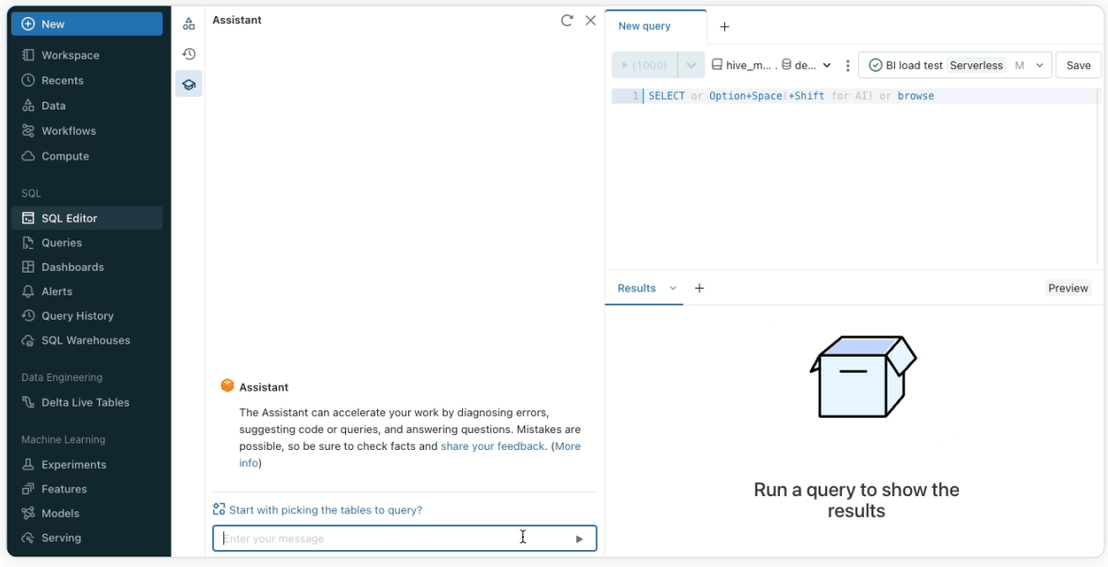
```
CREATE MATERIALIZED VIEW customer_orders
AS
SELECT
  customers.name,
  sum(orders.amount),
  orders.orderdate
FROM orders
  LEFT JOIN customers ON
    orders.custkey = customers.c_custkey
GROUP BY
  name,
  orderdate;
```

DB SQL でオーケストレーションが必要ですか？ Workflows では、SQL クエリ、ダッシュボード、アラートのスケジューリングが可能になりました。直感的な Workflows UI または API を使用して、タスク間の複雑な依存関係を簡単に管理し、過去のジョブ実行を監視できます。

ストリーミングテーブルとマテリアライズドビューがパブリックプレビューになりました。詳細については、専用のブログ記事をお読みください。両方のパブリックプレビューに登録するには、こちらのフォームからご登録いただけます。DB SQL のワークフローは現在一般的利用可能で、ドキュメント（AWS | Azure）で詳細を知ることができます。

Databricks Assistant : 自然言語でより良い SQL を高速に記述

Databricks Assistant は、Databricks Notebooks と SQL Editor に組み込まれた、コンテキストを意識した AI アシスタントです。Databricks Assistant は、自然言語による質問を受け付けると、その質問に答えるための SQL クエリを提案します。複雑なクエリを理解しようとする場合、ユーザーはアシスタントに自然言語を使って説明を求めることができ、誰でもクエリ結果の背後にあるロジックを理解できます。



Databricks Assistant は、より正確で適切な結果を提供するために多くのシグナルを使用します。コードセル、ライブラリ、一般的なテーブル、Unity Catalog スキーマ、タグなどのコンテキストを使用して、自然言語による質問をクエリやコードにマッピングします。

将来的には、DatabricksIQ との統合を追加し、お客さまのリクエストにより多くのコンテキストを提供する予定です。

データウェアハウスの効率的な管理

管理者と IT チームは、データウェアハウスの使用状況を把握するためのツールを必要としています。システムテーブル、ライブクエリプロファイル、ステートメントタイムアウトにより、管理者は問題を監視し、問題が発生したときに修正できます。

システムテーブルを使用して、SQL 環境をより深く可視化し、インサイトを得ることができます。システムテーブルは Databricks が提供するテーブルで、過去のステートメント実行、コスト、リネージなどに関する情報が含まれています。どのステートメントを誰が実行したのか、いつ、どのようにウェアハウスをスケールしたのか、何を請求されたのか、などの質問に答えるために、メタデータと使用メトリクスを探索します。システムテーブルは Databricks に統合されているため、SQL アラートや SQL ダッシュボードなどのネイティブ機能にアクセスし、モニタリングやアラートプロセスを自動化することができます。

2023 年 8 月現在、パブリックプレビュー中のシステムテーブルは 3 つあります。監査ログ、請求可能使用量システムテーブル、リネージシステムテーブル（AWS | Azure）です。ウェアハウスイベントとステートメント履歴のための追加システムテーブルは、まもなく登場する予定です。

例えば、SKU ごとの月間使用 DBU を計算するには、課金対象利用システムテーブルを照会します。

```
SELECT sku_name, usage_date, sum(usage_quantity) as `DBUs`
    FROM system.billing.usage
WHERE
    month(usage_date) = month(NOW())
    AND year(usage_date) = year(NOW())
GROUP BY sku_name, usage_date
```

ライブクエリプロファイルを使用すると、クエリパフォーマンスをリアルタイムで把握できるため、ワークロードをその場で最適化できます。クエリの実行計画を視覚化し、ライブクエリタスクの実行を評価して、爆発的な結合やフルテーブルスキャンなどの一般的な SQL ミスを修正します。ライブクエリプロファイルにより、データウェアハウス上で実行中のクエリが最適化され、効率的に実行されていることを確認できます。詳しくは、ドキュメント（AWS | Azure）をご覧ください。

自動制御をお探しですか？ステートメントタイムアウトでは、ワークスペースまたはクエリレベルのカスタムタイムアウトを設定できます。クエリの実行時間がタイムアウトのしきい値を超えると、クエリは自動的に停止します。詳しくはドキュメント（AWS | Azure）をご覧ください。

DBSQL の新たな機能

この 1 年間、私たちは Databricks SQL に新しい最先端のエクスペリエンスを追加するために懸命に取り組んできました。Databricks プラットフォーム全体で SQL ウェアハウスを実現し、新世代の SQL ダッシュボードを導入し、Python のパワーを Databricks SQL に取り入れるなど、SQL ユーザーの手に AI のパワーをもたらす新機能を発表できることを嬉しく思います。

AI Functions で非構造化データ分析を民主化

AI Functions により、DB SQL は SQL ウェアハウスに AI のパワーをもたらします。センチメント分析、テキスト分類、要約、翻訳などのタスクを実行することで、非構造化データの可能性を容易に活用できます。データアナリストはセルフサービスで AI モデルを適用でき、データエンジニアは AI 対応パイプラインを独自に構築できます。

AI Functions の使い方はとても簡単です。例えば、ユーザーが記事のセンチメントを「不満」、「満足」、「中立」、「満足」に分類したいとします。

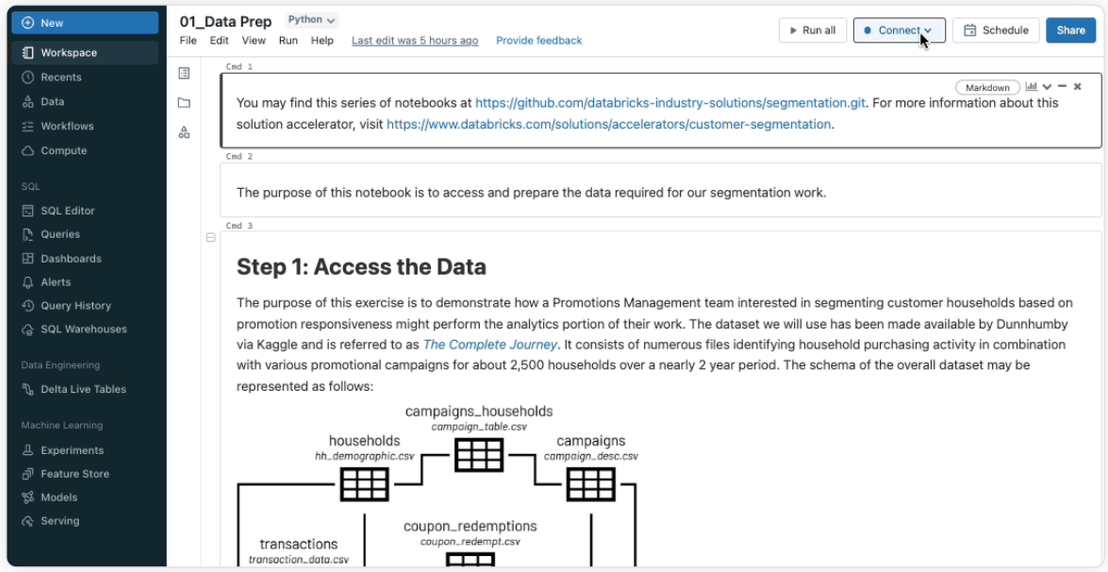
```
-- create a udf for sentiment classification
CREATE FUNCTION classify_sentiment(text STRING)
  RETURNS STRING
  RETURN ai_query(
    'Dolly', -- the name of the model serving endpoint
    named_struct(
      'prompt',
      CONCAT('Classify the following text into one of four categories [Frustrated,
Happy, Neutral, Satisfied]:\n',
        text),
      'temperature', 0.5),
    'returnType', 'STRING');
```

```
-- use the udf
SELECT classify_sentiment(text) AS sentiment
FROM reviews;
```

AI Functions は現在パブリックプレビュー中です。プレビューには、こちらのフォームからお申し込みいただけます。詳しくは、ブログ記事、または、ドキュメント（AWS | Azure）を参照してください。

SQL ウェアハウスのパワーを Notebook に

Databricks SQL ウェアハウスは、ノートブックの柔軟性と Databricks SQL サーバーレスおよび Pro ウェアハウスのパフォーマンスと TCO を組み合わせて、Notebook でパブリックプレビューできるようになりました。ノートブックで SQL ウェアハウスを有効にするには、ノートブックのコンピューティングドロップダウンから使用可能な SQL ウェアハウスを選択するだけです。



Databricks Notebook からのサーバーレス SQL ウェアハウス接続

新世代のダッシュボードによるインサイトの取得と共有

レイクハウスアーキテクチャ上で直接、刷新されたダッシュボードエクスペリエンスをご覧ください。ユーザーは必要なデータセットを選択するだけで、SQL オプションのエクスペリエンスで魅力的なビジュアライゼーションを構築できます。クエリとダッシュボードオブジェクトを別々に管理する必要はありません。オールインワンのコンテンツモデルにより、権限と管理プロセスが簡素化されます。最後に、ダッシュボードを組織全体に公開することで、ID プロバイダの認証済みユーザであれば、Databricks へのアクセス権を持っていなくても、安全な Web リンクを介してダッシュボードにアクセスできます。

新しい Databricks SQL ダッシュボードは現在プライベートプレビュー中です。詳細については、アカウントチームにお問い合わせください。

SQL で Python の柔軟性を活用

Python ユーザー定義関数（UDF）により、Python の柔軟性を Databricks SQL に取り込むことができます。SQL クエリからカスタム Python 関数を直接呼び出すことで、機械学習モデルを統合したり、データ処理や分析にカスタム再編集ロジックを適用したりできます。UDF は再利用可能な関数であり、データパイプラインや分析に一貫した処理を適用できます。

例えば、ファイルから電子メールと電話番号を削除するには、以下の CREATE FUNCTION 文を考えてください。

```
CREATE FUNCTION redact(a STRING)
RETURNS STRING
LANGUAGE PYTHON
AS $$
import json
keys = [“email”, “phone”]
obj = json.loads(a)
for k in obj:
  if k in keys:
    obj[k] = “REDACTED”
return json.dumps(obj)
$$;
```

プライベートプレビューのお申し込みについては、こちらをご覧ください。

データエコシステムとの統合

Databricks SQL はData+AIサミットで、選択したツールとのシームレスなエクスペリエンスのための新しい統合を発表しました。

Databricks + Fivetran

カタログに十分な権限を持つ非管理者を含む全てのユーザーが、Partner Connect で Fivetran にアクセスできるようになりました。このイノベーションにより、全てのユーザーが Fivetran を使用して Databricks にデータを取り込むことが 10 倍容易になりました。Salesforce や PostgreSQL など、Fivetran が提供する何百ものコネクタからレイクハウスアーキテクチャにデータを取り込むことができるようになったため、これは Databricks の全てのお客さまにとって大きな勝利です。Fivetran は現在、サーバーレスウェアハウスも完全にサポートしています！

詳しくはこちらのブログ記事をご覧ください。

Databricks + dbt Labs

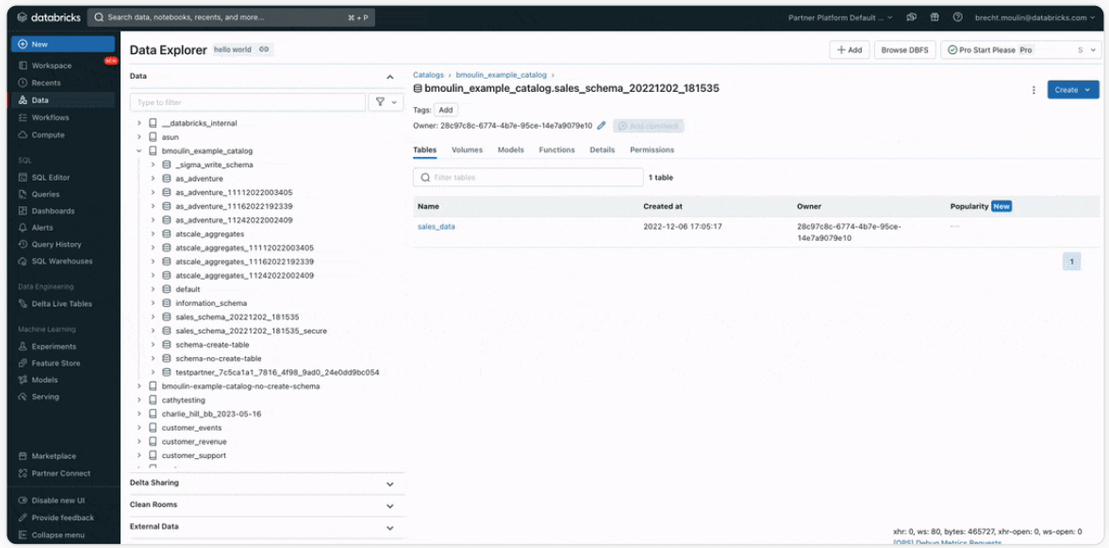
Databricks と dbt Labs でレイクハウスアーキテクチャ上のリアルタイムアナリティクスエンジニアリングを簡素化。高い人気を誇る dbt のアナリティクスエンジニアリングフレームワークと Databricks プラットフォームの組み合わせは、強力な機能を提供します。

- dbt + ストリーミングテーブル : あらゆるソースからのストリーミング取り込みが dbt プロジェクトに組み込まれました。SQL を使用して、アナリティクスエンジニアは dbt パイプライン内で直接クラウド／ストリーミングデータを定義し、取り込むことができます。
- dbt + マテリアライズドビュー : Databricks の強力なインクリメンタルリフレッシュ機能を活用することで、効率的なパイプラインの構築が容易になります。ユーザーは dbt を使用して、MV に支えられたパイプラインを構築および実行し、効率的なインクリメンタル計算によってインフラコストを削減できます。

詳しくはこちらのブログ記事をご覧ください。

Databricks + PowerBI : PowerBI ワークスペースへの公開

Databricks ワークスペースから PowerBI Online ワークスペースへ、数回のクリックでデータセットを公開できます。odbc/jdbc 接続を管理する必要はもうありません。公開するデータセットを選択するだけです。公開したいデータセットまたはスキーマを選択し、PBI ワークスペースを選択するだけです。これにより、BI 管理者やレポート作成者は、Power BI Desktop を使用しなくても PowerBI ワークスペースを簡単にサポートできるようになります。



Databricks SQL を開始するには

SQL ウェアハウスのセットアップ方法に関するガイド（AWS | Azure | GCP）に従って、Databricks SQL を今すぐ開始できます。Databricks SQL サーバーレスは現在 20% 以上のプロモーション割引が適用されています。

また、Data+AI サミットの動画「Databricks SQL : 最高のサーバーレスデータウェアハウスがレイクハウスである理由」や「Databricks SQL の新機能 ― ライブデモ付き」で概要をご覧いただけます。

セクション 3.5

Unity Catalog による分散型データガバナンスと孤立した環境の実現

執筆者 : Max Nienu、Zeashan Pappa、Paul Roome、Sachin Thakur

データ、アナリティクス、AI に業務を依存する組織では、効果的なデータガバナンスが不可欠です。多くの組織で、集中型データガバナンスの価値提案に対する認識が高まってきています。しかし、最高の意図を持っていても、適切な組織プロセスとリソースがなければ、集中型ガバナンスの導入は困難な場合があります。多くの組織では、最高データ責任者 (CDO) の役割がまだ確立されておらず、誰が組織全体のデータガバナンス方針を定義し、実行するのかについて疑問が残ります。

その結果、組織全体のデータガバナンスポリシーを定義し実行する責任が一元化されていないことが多く、組織内のビジネスライン、サブユニット、その他の部門間でポリシーが異なったり、管理団体が異なったりすることになります。簡単のため、このパターンを分散型ガバナンスと呼ぶことにします。このようなガバナンスユニット間の区別に関する一般的な合意はありますが、必ずしも中央データガバナンス機能があるわけではありません。

このブログでは、レイクハウスのデータ、アナリティクス、AI の統合ガバナンスソリューションを提供する Databricks Unity Catalog を使用して、分散型ガバナンスモデルの実装を検討します。

Databricks におけるデータガバナンスの進化形

Unity Catalog の導入以前は、ワークスペースの概念は一枚岩で、各ワークスペースは独自のメタストア、ユーザー管理、テーブル ACL ストアを有していました。そのため、ワークスペース間のデータおよびガバナンスの分離境界が内在し、ワークスペース間の一貫性に対処するための労力が重複していました。
この問題を解決するために、メタストアと ACL を同期させるパイプラインやコードを実行したり、ワークスペース間で使用する自己管理型メタストアを設定したりするお客さまもいらっしゃいました。しかし、これらのソリューションでは、オーバーヘッドとメンテナンスコストが増加し、組織全体でデータをどのように分割するかというアーキテクチャを前もって決定しなければならず、データサイロが発生してしまいます。

Unity Catalog によるデータガバナンス

これらの制約を克服するために、Databricks は、データガバナンスを簡単に実装しながら、データのコラボレーションと共有の能力を最大化することを目的とした「Unity Catalog」を開発しました。これを実現するための最初のステップは、組織内のあらゆるデータへのアクセスを許可する共通ネームスペースの実装でした。

このアプローチは、前述の分散ガバナンスパターンに対する挑戦のように見えるかもしれませんが、Unity Catalog は、組織が従来複数の Hive メタストアを使用して対処してきたネームスペース内の新しい分離メカニズムを提供します。これらの分離メカニズムにより、グループは最小限の相互作用で独立して動作することができ、また、本番環境と開発環境など、他のシナリオでも分離を実現することができるようになります。

Databricks における Hive Metastore と Unity Catalog の比較

Hive では、メタストアはサービスの境界であり、異なるメタストアを持つことは、異なるホストされた Hive の基礎サービスや異なる基礎データベースを意味しました。Unity Catalog は Databricks データインテリジェンスプラットフォーム内のプラットフォームサービスであるため、考慮すべきサービス境界は存在しません。

Unity Catalog は共通の名前空間を提供し、データを 1 か所で管理・監査できるようにします。

Hive を使用する場合、開発環境と本番環境の分離を実現したり、運用単位でデータの分離を可能にするために、それぞれ独自の名前空間を持つ複数のメタストアを使用することが一般的でした。

Unity Catalog では、これらの要件は、データの共有や共同作業の能力を損なわず、一方通行で難しい先行アーキテクチャの決定を必要としない、ネームスペース上の動的な分離メカニズムによって解決されます。

異なるチームや環境での作業

データプラットフォームを利用する場合、開発／生産などの環境間や、組織のビジネスグループ、チーム、事業部間の隔離境界が強く求められることがあります。

まず、Databricks のようなデータプラットフォームにおける隔離境界の定義から始めましょう。

- ユーザーは、合意されたアクセスルールに基づいてのみデータにアクセスできるようにする
- データは指定された人またはチームによって管理することができる
- データはストレージ内で物理的に分離する
- データは指定された環境でのみアクセスできるようにする

ユーザーは、合意されたアクセスルールに基づき、データへのアクセスのみを行うべきである

組織は通常、データの安全性を保つために、組織や規制の要件に基づいてデータアクセスに関する厳格な要件を定めています。典型的な例としては、従業員の給与情報やクレジットカードの支払い情報などが挙げられます。

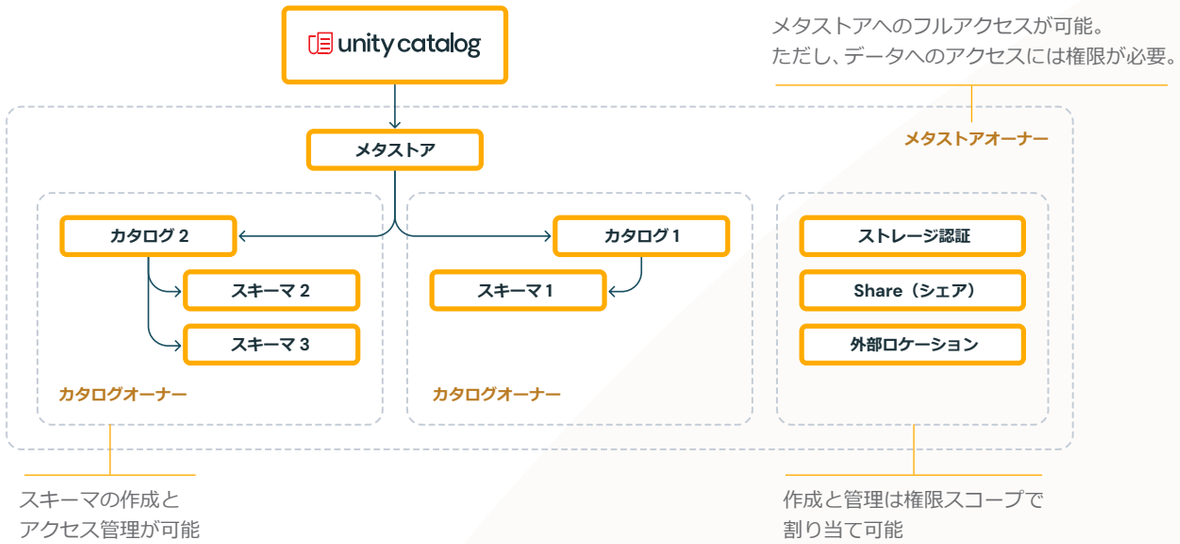
このような情報へのアクセスは、通常、厳しく管理され、定期的に監査されます。Unity Catalog は、これらの業界標準を満たすために、カタログ内のデータ資産に対するきめ細かい制御を組織に提供します。コントロールにより、Unity Catalog は、ユーザーが閲覧およびクエリする権利があるデータのみを閲覧およびクエリすることができます。

データを指定された人またはチームで管理することができる

Unity Catalog では、集中型ガバナンスモデルと分散型ガバナンスモデルから選択することができます。

集中型ガバナンスモデルでは、ガバナンス管理者がメタストアのオーナーとなり、任意のオブジェクトの所有権を取得し、ACL やポリシーを設定することができます。

分散ガバナンスモデルでは、カタログまたはカタログのセットをデータドメインと見なすことになります。そのカタログの所有者は、すべてのアセットを作成し所有し、そのドメイン内のガバナンスを管理することができます。したがって、ドメインの所有者は、他のドメインの他の所有者から独立して操作することができます。管理がツールで行われる場合は、これらのオプションの両方で、グループをオーナーまたはサービスプリンシパルに設定することを強くお勧めします。



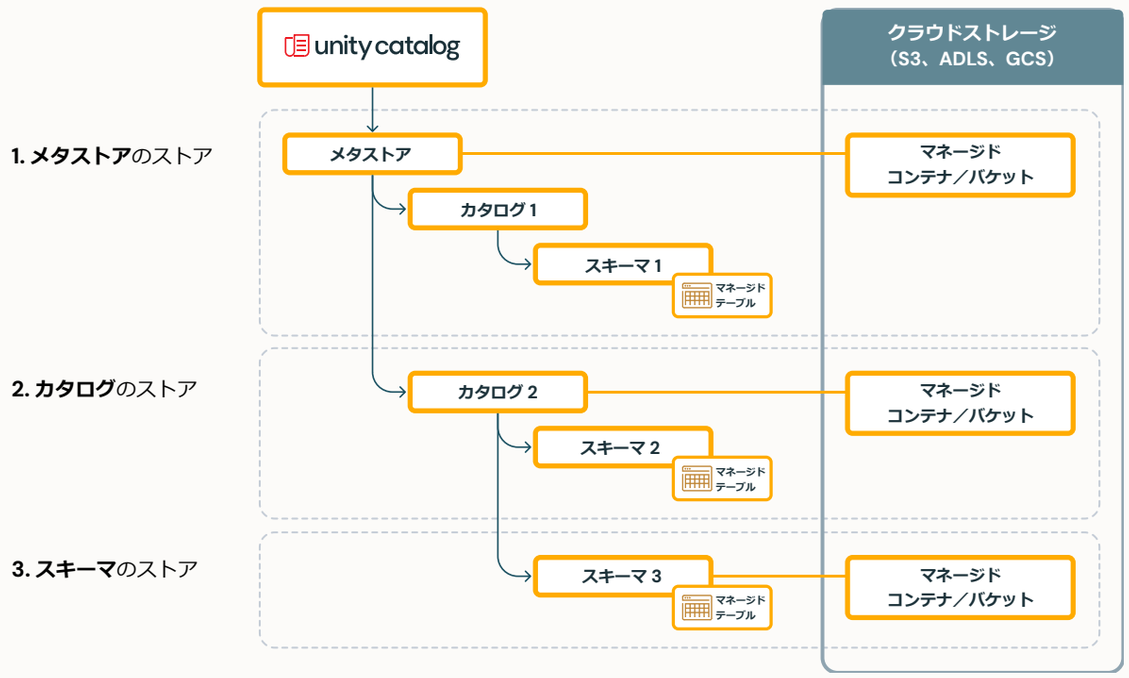
データは物理的に分離して保管する必要がある

デフォルトでは、UC メタストアの作成時に、Databricks アカウント管理者は、管理対象テーブルのデフォルトの場所として、単一のクラウドストレージ場所とクレデンシャルを提供します。

規制上の理由から、あるいは SDLC スコープ間、事業部間、あるいはコスト配分を目的としたデータの物理的な分離を必要とする組織は、カタログおよびスキーマレベルでの管理対象データソース機能を検討する必要があります。

Unity Catalog では、データがストレージ内でどのように分離されるかのデフォルトを選択することができます。デフォルトでは、すべてのデータはメタストアに保存されます。カタログとスキーマでのマネージドデータソースの機能サポートにより、データの保存とアクセスを物理的に分離することができ、組織のガバナンスとデータ管理要件の達成を支援します。

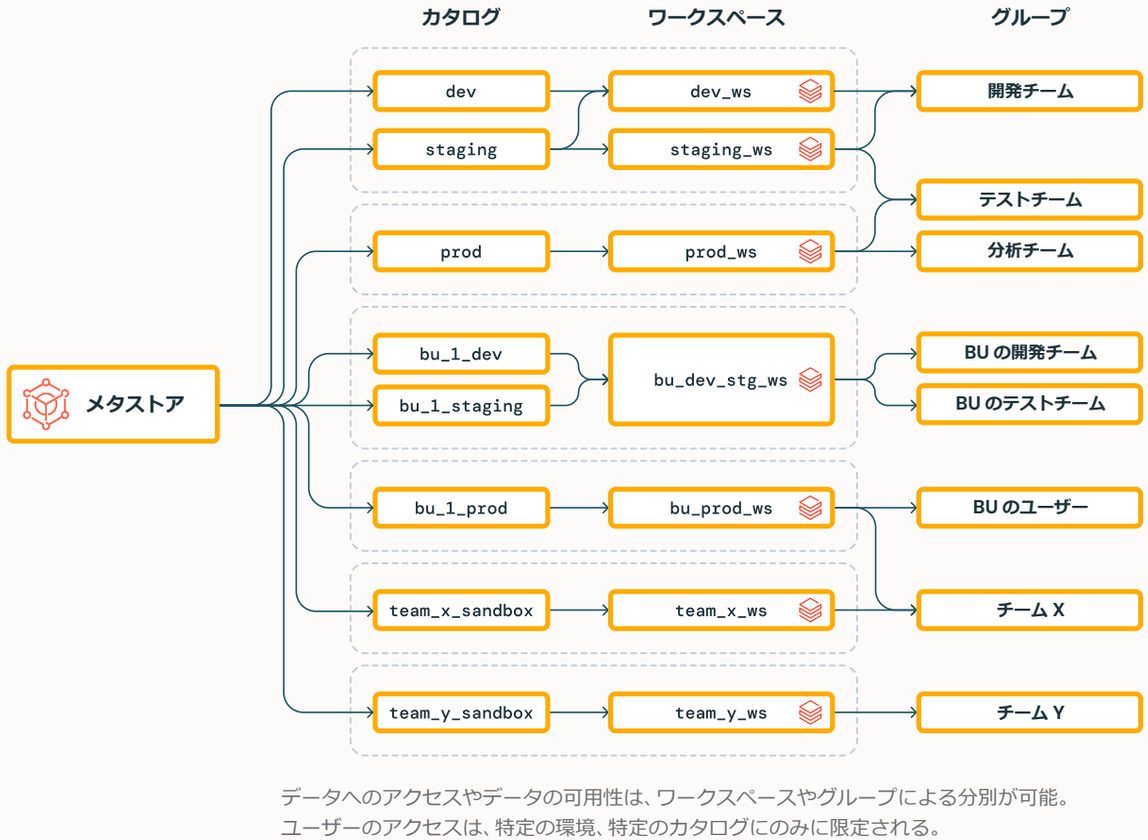
マネージドテーブルを作成する場合、データはスキーマの場所（ある場合）、カタログの場所（ある場合）の順に使用され、前の 2 つの場所が設定されていない場合はメタストアの場所のみが使用されます。



データへのアクセスは目的に基づいて指定された環境でのみ許可される

多くの場合、組織やコンプライアンス要件により、特定のデータに特定の環境でのみアクセスできるようにする必要があることがあります。例えば、開発環境と本番環境、HIPAA や PII 環境では、分析用の PII データを含むため、データにアクセスできる人やそのデータへのアクセスを許可する環境に関する特別なアクセスルールがあります。また、特定のデータセットやドメインが混在したり結合したりできないような要件が定められていることもあります。

Databricks では、ワークスペースを環境の 1 つとみなしています。Unity Catalog には、カタログをワークスペースに「バインド」できる機能があります。この環境を意識した ACL により、ユーザーの個々の ACL に関係なく、ワークスペース内で特定のカタログのみを利用できるようにすることができます。つまり、メタストア管理者、またはカタログ所有者は、データカタログがアクセスできるワークスペースを定義することができます。これは、UI や API/terraform を使用して簡単に統合することができます。また、最近 terraform 経由で Unity Catalog を制御する方法についてのブログを公開しましたので、特定のガバナンスモデルに適合させるのに役立ちます。



データへのアクセスやデータの可用性は、ワークスペースやグループによる分別が可能。
ユーザーのアクセスは、特定の環境、特定のカタログにのみに限定される。

まとめ

Unity Catalog をレイクハウスアーキテクチャの中心に据えることで、データを効率的に管理・共有する能力を犠牲にすることなく、柔軟で拡張性のあるガバナンスの実装を実現できます。Unity Catalog を使用すると、既存の Hive メタストアの制限や制約を克服し、特定のビジネスニーズに応じて、より簡単にデータを分離してコラボレーションできるようになります。Unity Catalog のガイド（AWS | Azure）に従って、まずはお試しください。データレイクハウスの効果的なガバナンス戦略を構築するためのベストプラクティスについては、データ、アナリティクス、AI ガバナンスに関するこの無料の eBook をダウンロードしてください。

セクション 3.6

Unity Catalog におけるデータ権限モデルとアクセス制御のためのヒッチハイカーズガイド

Unity Catalog の特権モデル内の概念をシンプルでわかりやすく抽出し、さまざまなアクセスニーズやパターンをサポートします

執筆者 : Som Natarajan、Vuong Nguyen

データの量、速度、多様性が増すにつれ、組織は、中核となるビジネス成果を適切に満たすために、確固たるデータガバナンスの実践にますます頼るようになっています。Unity Catalog は、Databricks データインテリジェンスプラットフォームを支えるデータと AI のためのきめ細かなガバナンス・ソリューションです。データアクセスを管理・監査するための一元的なメカニズムを提供することで、企業のデータ資産のセキュリティとガバナンスを簡素化することができます。

Unity Catalog がファイル、テーブルの権限モデルを統一し、あらゆる言語をサポートするようになる以前、お客さまはレガシーワークスペースレベルのテーブル ACL（TACL）を使用して Databricks できめ細かいデータアクセス制御を実施していましたが、これは特定のクラスタ構成に限定され、Python と SQL に対してのみ機能しました。Unity Catalog と TACL はどちらも、カタログ、スキーマ（データベース）、テーブル、ビューなどのセキュリティで保護されたオブジェクトへのアクセスを制御できますが、それぞれのアクセスモデルがどのように機能するかには、いくつかのニュアンスがあります。

Unity Catalog を使って大規模にデータガバナンスを実施するには、オブジェクトアクセスモデルをよく理解することが必要です。すでにテーブル ACL モデルを導入しており、Unity Catalog にアップグレードして、多言語対応、集中アクセス制御、データリネージなどの最新機能を活用しようとしている場合は、なおさらです。

Unity Catalog のアクセスモデルの公理

- Unity Catalog の権限は、メタストアで定義されます : Unity Catalog の権限は常にアカウントレベルの ID を参照し、hive_metastore カタログ内で定義された TACL 権限は常にワークスペースのローカル ID を参照します
- 特権の継承 : Unity Catalog のオブジェクトは階層化されており、権限は下へ下へと継承されます。権限が継承される最上位のオブジェクトはカタログです。
- オブジェクトの所有権が重要 : 特権は、メタストア管理者、オブジェクトの所有者、またはオブジェクトを含むカタログやスキーマの所有者のみが付与することができます。オブジェクトの所有者、またはオブジェクトを含むカタログやスキーマの所有者のみが、オブジェクトをドロップできます。
- 境界のための USE 特権 : カタログ / スキーマ内のオブジェクトと対話するには、USE CATALOG/SCHEMA が必要です。しかし、USE 権限では、カタログ / スキーマ内に収容されているオブジェクト・メタデータを参照することはできません。
- 派生オブジェクトのパーミッションが簡略化される - Unity Catalog では、ビューの所有者には SELECT 権限と、ビューの親スキーマの USE SCHEMA、親カタログの USE CATALOG が必要なだけです。TACL とは対照的に、ビューの所有者は参照されるすべてのテーブルとビューの所有者である必要があります。

より複雑な公理をいくつか紹介

- デフォルトでセキュア : Unity-Catalog 固有の access modes（共有またはシングルユーザー）を持つクラスタのみが、Unity Catalog データにアクセスできます。TACL を使用すると、非共有クラスタではすべてのユーザーがすべてのデータにアクセスすることができます。
- シングルユーザークラスタの限界 : シングルユーザークラスターは、ダイナミックビューをサポートしていません。ビューから読み込むには、参照するすべてのテーブルとビューに対して SELECT を持つ必要があります。
- ANY FILE、ANONYMOUS FUNCTION をサポートしていません。- Unity Catalog は、これらの権限をサポートしません。なぜなら、これらの権限は、特権を持たないユーザーが特権コードを実行できるようにすることで、アクセス制御の制限を回避するために使用される可能性があるからです。

興味深いガバナンスパターン

Unity Catalog のアクセスモデルを使って実現できるガバナンスパターンはたくさんあります。

Example 1 - ワークスペース間で一貫した権限設定

公理 1 では、製品チームが自分のワークスペース内でデータ製品の権限を定義し、それを他のすべてのワークスペースに反映させ、消費者がどこから来るかに関係なく強制することができます。

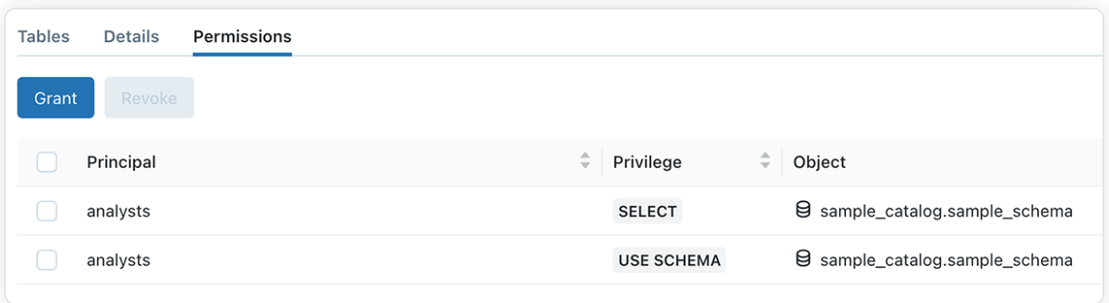
Example 2 - データ共有の境界を設定する

公理 2 では、カタログ / スキーマの所有者がデータに対するデフォルトのアクセスルールを設定することができます。たとえば、次のコマンドは、機械学習チームがスキーマ内でテーブルを作成し、互いのテーブルを読み取ることを可能にします。

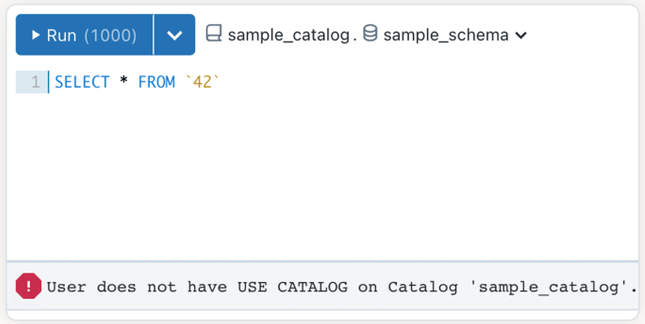
```
CREATE CATALOG ml;
CREATE SCHEMA ml.sandbox;
GRANT USE_CATALOG ON CATALOG ml TO ml_users;
GRANT USE_SCHEMA ON SCHEMA ml.sandbox TO ml_users;
GRANT CREATE TABLE ON SCHEMA ml.sandbox TO ml_users;
GRANT SELECT ON SCHEMA ml.sandbox TO ml_users;
```

さらに興味深いことに、公理 4 ではカタログ / スキーマの所有者が、個々のスキーマおよびテーブルの所有者が生成するデータを共有できる範囲を制限できるようになりました。テーブルの所有者が他のユーザーに SELECT を付与しても、そのユーザーが親カタログの USE CATALOG 権限と親スキーマの USE SCHEMA 権限を付与されていない限り、そのテーブルへの読み取りアクセスは許可されません。

以下の例では、sample_catalog はユーザー A が所有し、ユーザー B は sample_schema スキーマを作成し、テーブル 42 を作成しました。アナリストチームに USE SCHEMA と SELECT 権限が付与されていても、ユーザー A が設定した権限境界のために、アナリストチームはテーブルを照会することができません。



sample_catalog の SELECT および USE SCHEMA 権限を持つアナリストグループを示す権限ページです。



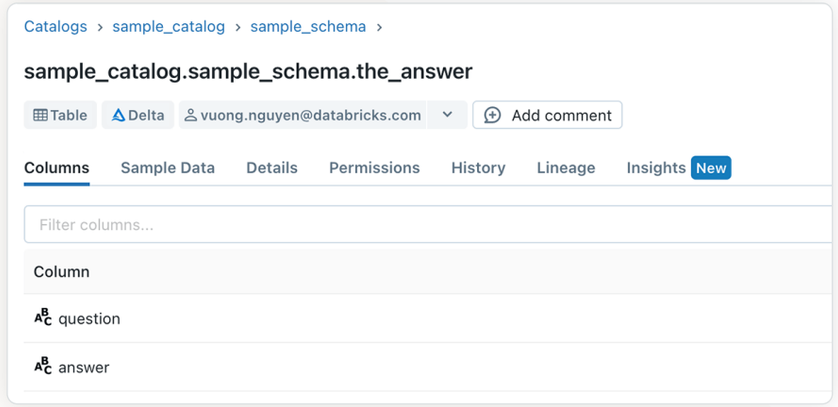
エラーメッセージが表示されるクエリーページ

Example 3 – ビジネスロジックの共有が容易に

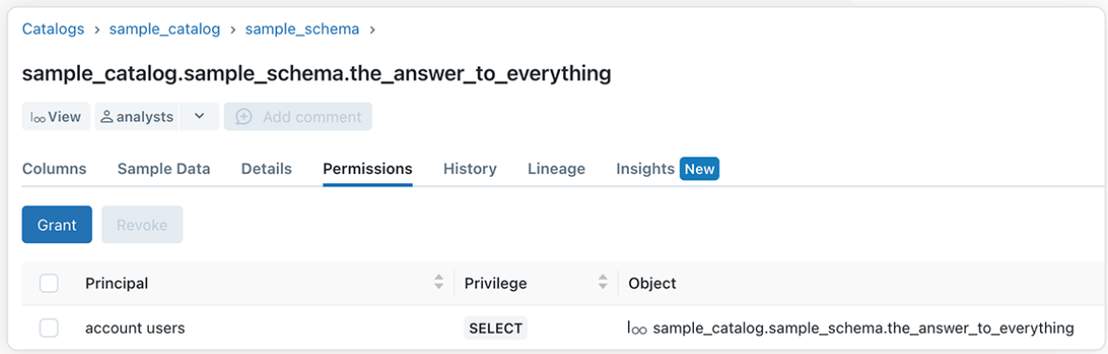
データコンシューマは、その作業や変換ロジックを共有する必要があり、それを行うための再利用可能な方法は、ビューを作成して他のコンシューマに共有することです。

公理５は、データコンシューマがテーブルの所有者と手動でやり取りすることなく、シームレスにこれを実行できるようにする機能を提供します。

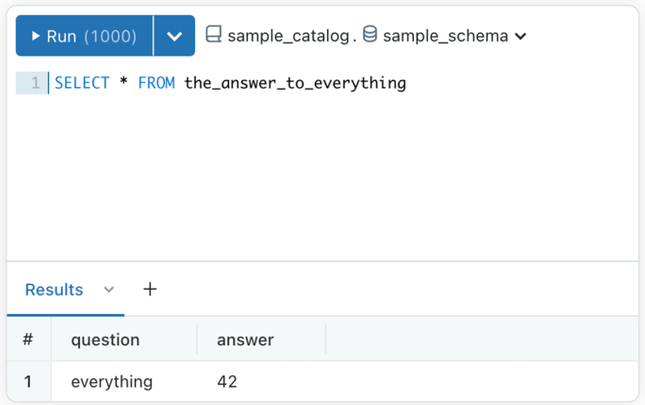
ビューの定義



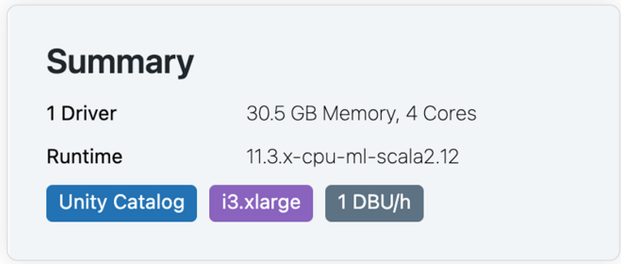
テーブルの所有権



アナリストグループが所有するビューと、アカウントユーザーグループの SELECT 権限が表示される権限ページ



全てに回答した結果を表示するクエリーページ



Unity Catalog のサポートを示すクラスタ概要

Example 4 – 情報漏えいを防ぐ

公理 6 のおかげで、データ所有者は、クラスタの誤設定によるデータへの不正アクセスがないことを確信することができます。正しいアクセスモードが設定されていないクラスタは、Unity Catalog のデータにアクセスすることができません。

ユーザーは、クラスタの作成ページでこの便利なツールチップを使用して、自分のクラスタが Unity Catalog のデータにアクセスできることを確認できます。

データ所有者がデータ権限モデルとアクセス制御を理解できるようになったことで、Unity Catalog を活用し、大規模なアクセスポリシー管理を簡素化することができます。

今後、データ管理者やデータ所有者がより複雑なアクセスポリシーを作成できるようにするための機能が追加される予定です。

- **行フィルタリングと列マスキング :** 標準 SQL 関数を使用して行フィルタと列マスクを定義し、行と列に対するきめ細かなアクセス制御を可能にします。
- **属性に基づくアクセス制御 :** データ資産のタグ（属性）に基づいて、アクセスポリシーを定義します。

セクション

04

Databricks における分析ユースケース

セクション 4.1

Databricks で Fivetran と dbt を使ってマーケティング分析ソリューションを構築する方法

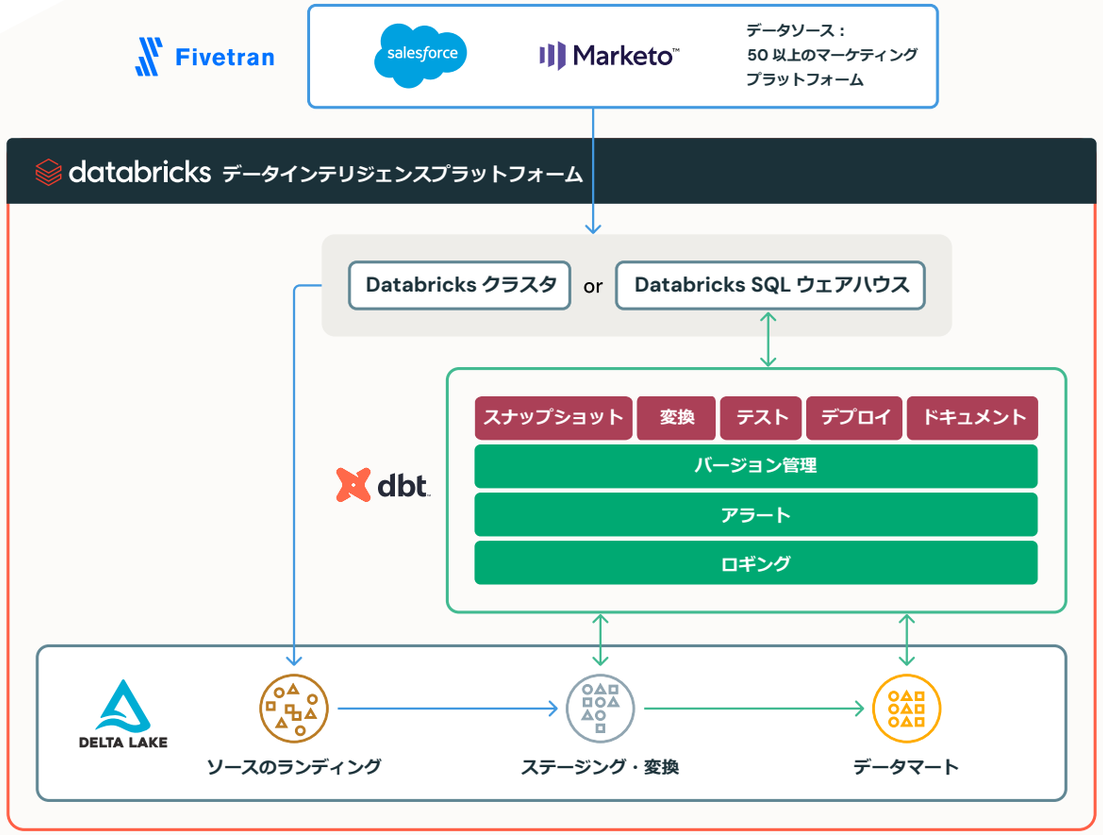
執筆者 : Tahir Fayyaz、Bilal Aslam、Robert Saxby

マーケティング部門は、さまざまなプラットフォームを使用してマーケティングキャンペーンやセールスキャンペーンを行っています。これらのプラットフォームでは、価値はあるものの分断された膨大な量のデータが生成される可能性があります。これら全てのデータを統合することは大きな投資収益率の向上に役立ちます。Publicis Groupe では、データを統合し、キャンペーン収益を 50% 増加させることに成功しました。

データウェアハウスと AI のユースケースを単一のプラットフォーム上で統合する Databricks は、マーケティング分析ソリューションを構築する理想的な場所です。信頼できる唯一の情報源を維持し、AI/ML のユースケースを開拓します。また、Databricks のパートナーソリューションである Fivetran と dbt を活用することで、顧客離脱防止やライフタイムバリューの分析、顧客セグメンテーション、広告効果など、幅広いマーケティングアナリティクスのユースケースを実現できます。

Fivetran は、複雑なパイプラインを構築・維持することなく、50 以上のマーケティングプラットフォームから Delta Lake に容易にデータを取り込むことができます。マーケティングプラットフォームの API が変更されたり、機能しなくなった場合でも、Fivetran が統合の更新と修正を行い、マーケティングデータは継続的に取り込まれます。

dbt は、レイクハウスユーザーがシンプルな SQL を使ってデータパイプラインを構築できる人気のオープンソースフレームワークです。全てがディレクトリ内にプレーンテキストとして整理されているため、バージョン管理、デプロイ、テストが簡単に行えます。データが Delta Lake に取り込まれたら dbt を使ってデータを変換、テストし、文書化します。取り込まれたデータの上に構築されたマーケティング分析データマートは、新しいマーケティングキャンペーンやイニシアチブの推進に役立てることができます。



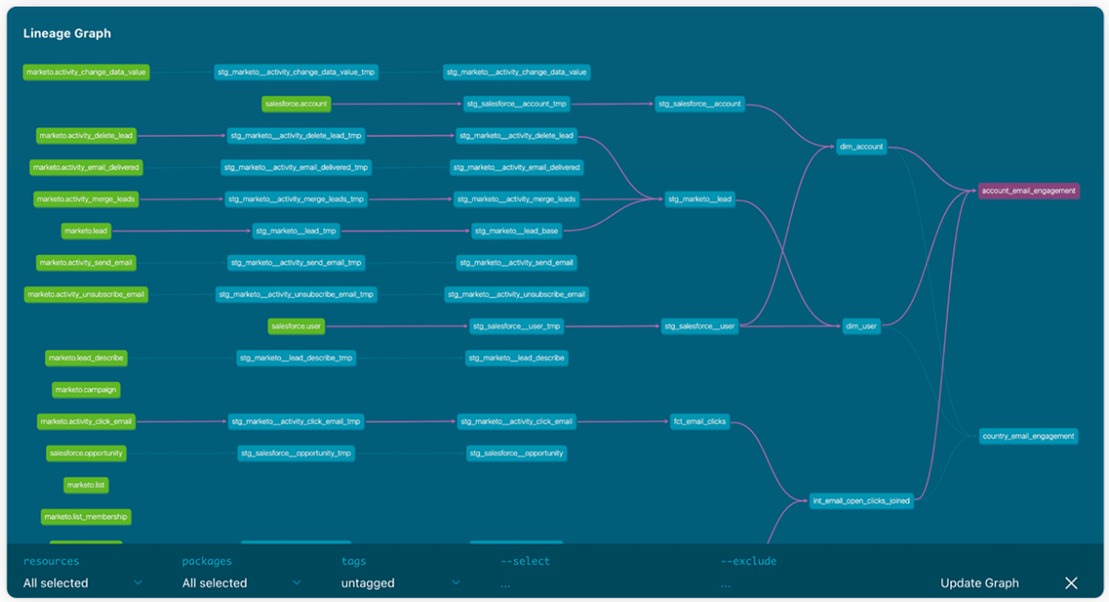
Fivetran と dbt は、Databricks クラスタまたは Databricks SQL ウェアハウスを介して Delta Lake での読み取り・書き込みを行う。

Fivetran と dbt はどちらも Databricks パートナーコネクトの一部であり、Databricks プラットフォーム内でデータ、分析、AI ツールを発見し、安全に直接接続するためのワンストップポータルです。数回クリックするだけで、Databricks ワークスペース内からこれらのツール（およびその他多数）を直接設定し、接続できます。

マーケティング分析ソリューションの構築方法

このハンズオンデモでは、Fivetran を使用して Marketo と Salesforce のデータを Databricks に取り込み、dbt を使用してマーケティング分析データモデルを変換、テスト、文書化する方法を紹介します。

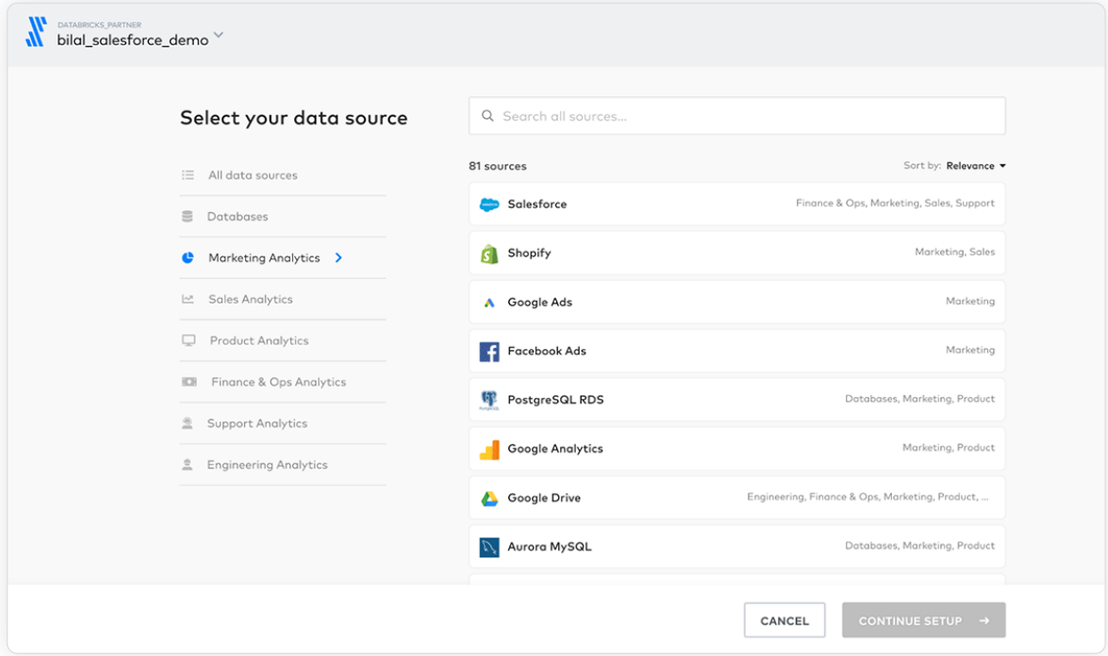
デモのコードは全て Github の workflows-examples リポジトリにあります。



データソースとモデルを示す dbt 系統グラフ

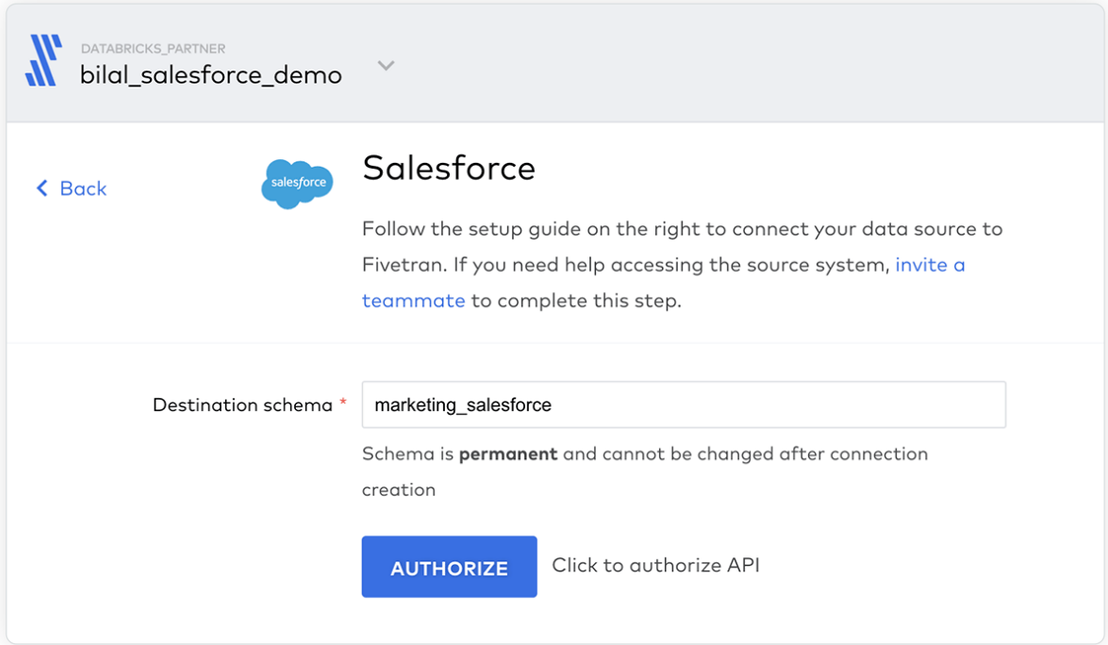
最終的な dbt モデルの系統図はこのようになります。Fivetran のソーステーブルは左側の緑色で、最終的なマーケティング分析モデルは右側にあります。モデルを選択すると、異なるモデルが紫色でハイライトされ、対応する依存関係を見ることができます。

Fivetran を使用したデータ取り込み



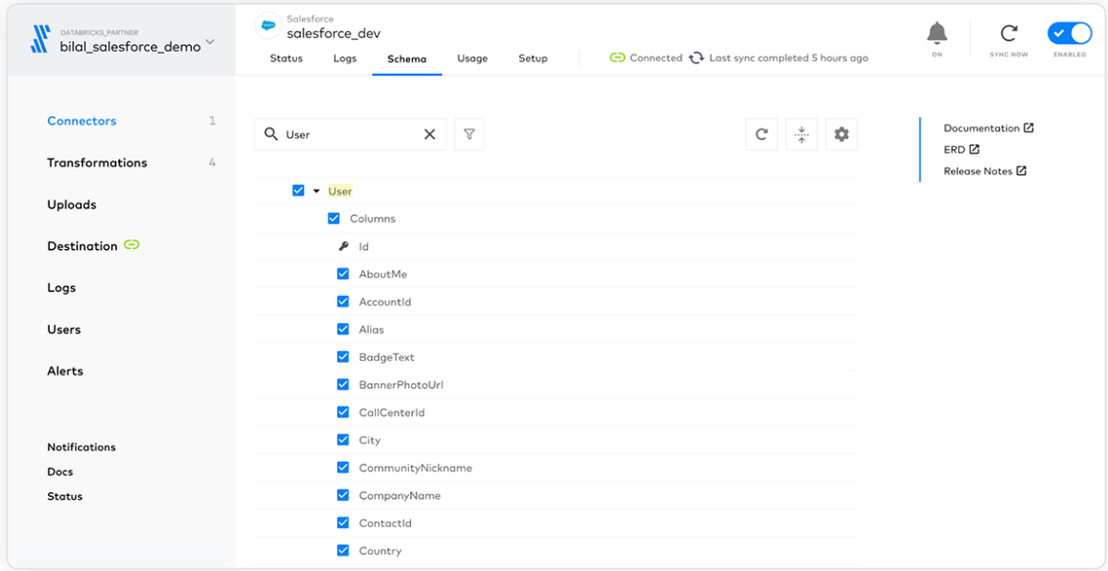
Fivetran は、多くのマーケティング分析データソースのコネクタを持っています。

Fivetran で新しい Salesforce と Marketo の接続を作成し、マーケティングデータの Delta Lake への取り込みを開始します。接続を作成する際、Fivetran は Delta Lake の各データソースのスキーマも自動的に作成し、管理します。後で dbt を使用して、このデータを変換、クリーニング、集計します。



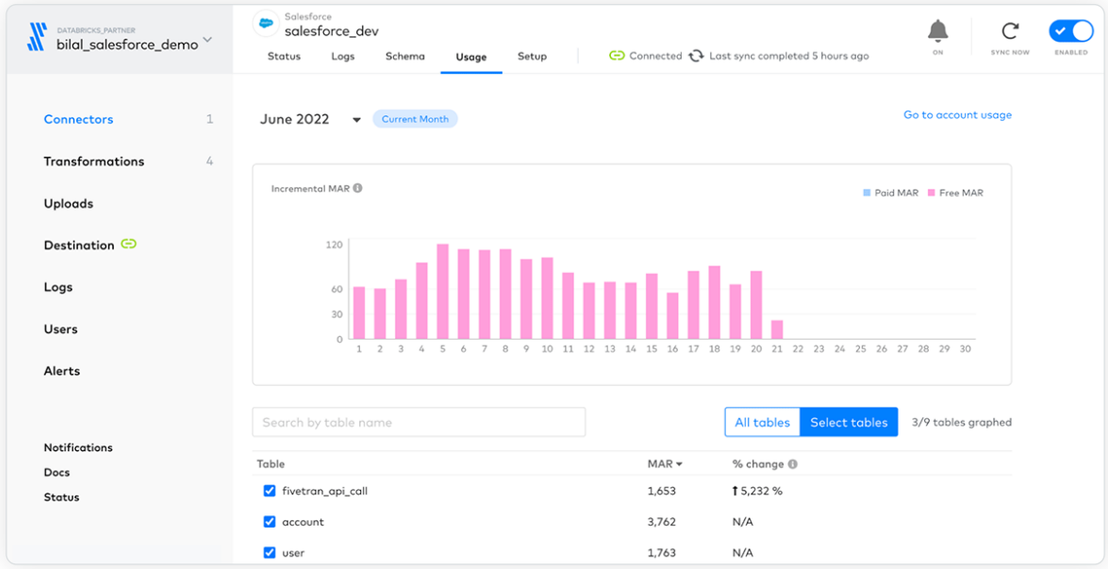
Salesforce データソースのデスティネーションスキーマを Delta Lake で定義します。

デモでは、Delta Lake の marketing_salesforce と marketing_marketo に作成されるスキーマの名前を指定します。スキーマが存在しない場合、Fivetran は最初の取り込みロードの一部としてそれらを作成します。



Delta Lake テーブルとして同期するデータソースオブジェクトを選択します。

次に、各オブジェクトが個別のテーブルとして保存される Delta Lake に同期するオブジェクトを選択することができます。Fivetran はまた、各テーブルで同期されているカラムを簡単に管理・表示できます。



毎月のアクティブな行を同期して監視する Fivetran 監視ダッシュボード

さらに、Fivetran には、各テーブルで毎日および毎月同期されている月次アクティブなデータ行の数、その他の有用な統計やログを分析できる監視ダッシュボードが用意されています。

dbt によるデータモデリング

全てのマーケティングデータが Delta Lake にあるので、以下のステップで dbt を使用してデータモデルを作成できます。

ローカルに dbt プロジェクトをセットアップし、Databricks SQL に接続

dbt Core と dbt-databricks のセットアップ手順に従って、選択した IDE でローカルの dbt 開発環境をセットアップします。

新しい dbt プロジェクトを作成し、dbt init を使用して Databricks SQL ウェアハウスに接続します。これにより、次の情報の入力が求められます。

```
$ dbt init
Enter a name for your project (letters, digits, underscore):
Which database would you like to use?
[1] databricks
[2] spark

Enter a number: 1
host (yourorg.databricks.com):
http_path (HTTP Path):
token (dapiXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX):
schema (default schema that dbt will build objects in):
threads (1 or more) [1]:
```

プロファイルを設定したら、次の方法で接続をテストできます。

```
$ dbt debug
```

ステージング用 Fivetran dbt モデルパッケージのインストール

Marketo と Salesforce のデータを使用する最初のステップは、モデルのソースとしてテーブルを作成することです。幸いなことに、Fivetran はビルド済みの Fivetran dbt モデルパッケージを使用して、これを簡単に実行できるようにしています。このデモでは、`marketo_source` パッケージと `salesforce_source` パッケージを使用してみましょう。

パッケージをインストールするには、`packages.yml` ファイル dbt プロジェクトのルートに追加し、`marketo-source`、`salesforce-source`、および `fivetran-utils` パッケージを追加します。

```
packages:
  - package: dbt-labs/spark_utils
    version: 0.3.0
  - package: fivetran/marketo_source
    version: [“>=0.7.0”, “=0.4.0”, “

To download and use the packages run
```

```
$ dbt deps
```

これで、packages フォルダに Fivetran パッケージがインストールされているはずです。

Fivetran の dbt モデル用に dbt_project.yml を更新

`dbt_project.yml` ファイルには、Fivetran パッケージが Databricks で正しく動作するように変更する必要がある設定がいくつかあります。

`dbt_project.yml` ファイルは dbt プロジェクトのルートフォルダにあります。

dbt_utils マクロをオーバーライドする spark_utils

Fivetran dbt モデルは `dbt_utils` パッケージのマクロを使用していますが、これらのマクロの一部は Databricks で動作するように変更する必要があり、これは `spark_utils` パッケージを使って容易に行うことができます。

これは、`dbt_project.yml` ファイルの dispatch config 使用して設定することができる特定の `dbt_utils` マクロのためのシムを提供することによって動作し、これにより dbt は、`dbt_utils` 名前空間からマクロを解決する際に、最初に `spark_utils` パッケージ内のマクロを検索します。

```
dispatch:
 - macro_namespace: dbt_utils
   search_order: [‘spark_utils’, ‘dbt_utils’]
```

marketo_source および salesforce_source スキーマの変数

Fivetran パッケージでは、Fivetran に取り込まれるデータのカタログ（dbt ではデータベースと呼ばれます）とスキーマを定義する必要があります。

これらの変数を `dbt_project.yml` ファイルに正しいカタログ名とスキーマ名で追加します。デフォルトのカタログは `hive_metastore` で、`_database` が空白の場合はこれが使用されます。スキーマ名は、Fivetran で接続を作成するときに定義したものになります。

```
vars:
 marketo_source:
   marketo_database: # leave blank to use the default hive_metastore catalog
   marketo_schema: marketing_marketo
 salesforce_source:
   salesforce_database: # leave blank to use the default hive_metastore catalog
   salesforce_schema: marketing_salesforce
```

Fivetran ステージングモデルのターゲットスキーマ

Fivetran ソースモデルによって作成される全てのステージングテーブルがデフォルトのターゲットスキーマに作成されるのを避けるには、別のステージングスキーマを定義するのが便利です。

`dbt_project.yml` ファイルにステージングスキーマ名を追加すると、デフォルトのスキーマ名の末尾にこの名前が付けられます。

```
models:
 marketo_source:
   +schema: your_staging_name # leave blank to use the default target_schema
 salesforce_source:
   +schema: your_staging_name # leave blank to use the default target_schema
```

上記のことから、`profiles.yml` で定義したターゲットスキーマが `mkt_analytics` の場合、`marketo_source` テーブルと `salesforce_source` テーブルに使用されるスキーマは `mkt_analytics_your_staging_name` になります。

欠落テーブルの無効化

この段階で、Fivetran モデルパッケージを実行し、正しく動作することをテストできます。

```
dbt run -select marketo_source
```

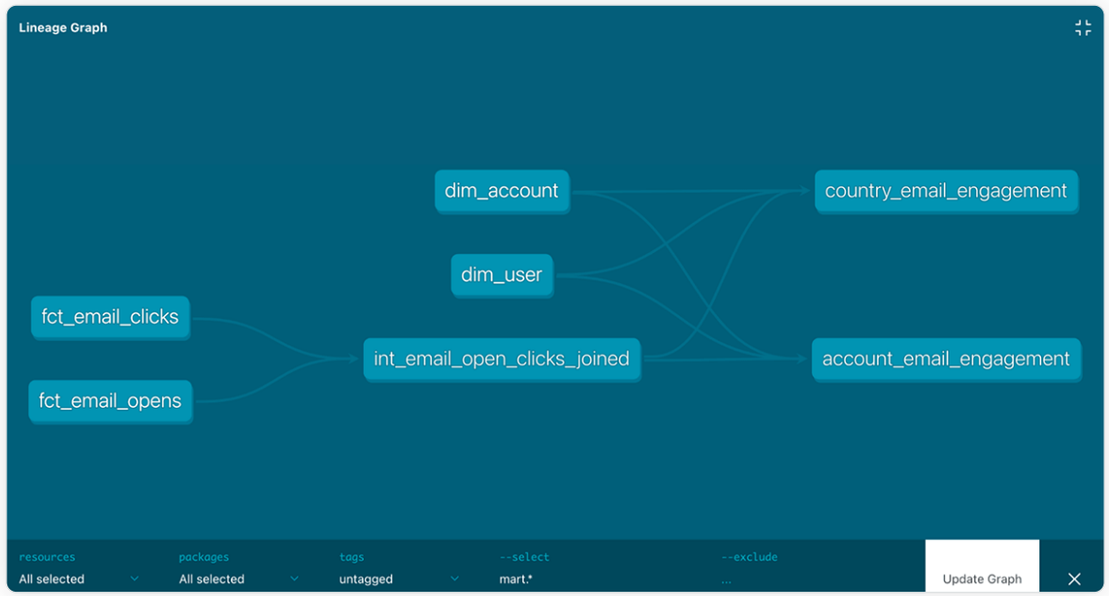
```
dbt run -select marketo_source
```

Fivetran でこれらのテーブルを同期しないことを選択したため、テーブルが欠落してモデルのいずれかが失敗した場合、ソーススキーマで `dbt_project.yml` ファイルを更新することで、これらのモデルを無効にすることができます。

例えば、バウンスされたメールテーブルとメールテンプレートテーブルが Marketo のソーススキーマにない場合、models config の下に以下を追加することで、これらのテーブルのモデルを無効にできます。

```
models:
 marketo_source:
   +schema: your_staging_name
   tmp:
     stg_marketo__activity_email_bounced_tmp:
       +enabled: false
     stg_marketo__email_template_history_tmp:
       +enabled: false
   stg_marketo__activity_email_bounced:
     +enabled: false
   stg_marketo__email_template_history:
     +enabled: false
```

マーケティング分析モデルの開発



スタースキーマと集約テーブルのデータモデルを示す dbt 系統グラフ

Fivetran パッケージによってステージングモデルの作成とテストが行われたので、マーケティング分析ユースケースのデータモデルを開発し始めることができます。これは、マテリアライズド集計テーブルとともにスタースキーマデータモデルになります。

例えば、最初のマーケティングアナリティクスダッシュボードでは、特定の企業や営業地域が、メールキャンペーンを開封したりクリックしたりした数によって、どの程度関与しているかを確認できます。

そのためには、Salesforce のユーザーメールアドレス、Salesforce の `account_id`、Marketo の `lead_id` を使用して、Salesforce と Marketo のテーブルを結合します。

モデルは mart フォルダの下に以下のように構成されます。

```
marketing_analytics_demo
|-- dbt_project.yml
|-- packages.yml
|-- models
      |-- mart
            |-- core
            |-- intermediate
            |-- marketing_analytics
```

Github の全てのモデルのコードは、/models/mart ディレクトリにあります。以下では、各フォルダーの内容と例について説明します。

コアモデル

コアモデルは、全ての下流モデルで使用されるファクトとディメンションテーブルです。下流のあらゆるモデルが構築する際に使用するものです。

`dim_user` モデルの dbt SQL コード

```sql
with salesforce_users as (
   select
       account_id,
       email
   from {{ ref('stg_salesforce__user') }}
   where email is not null and account_id is not null
),
marketo_users as (
   select
       lead_id,
       email
   from {{ ref('stg_marketo__lead') }}
),
joined as (
   select
     lead_id,
     account_id
   from salesforce_users
     left join marketo_users
     on salesforce_users.email = marketo_users.email
)

select * from joined
```

フォルダ内の yaml ファイルを使って、モデルのドキュメントとテストを追加することもできます。

`core.yml` ファイルには 2 つの簡単なテストが追加されています。

```
version: 2

models:
 - name: dim_account
   description: “The Account Dimension Table”
   columns:
     - name: account_id
       description: “Primary key”
       tests:
         - not_null
 - name: dim_user
   description: “The User Dimension Table”
   columns:
     - name: lead_id
       description: “Primary key”
       tests:
         - not_null
```

中級モデル

最終的な下流モデルのいくつかは、同じ計算メトリクスに依存している可能性があります。したがって、SQL の繰り返しを避けるために、再利用可能な中間モデルを作成することができます。

`int_email_open_clicks_joined` モデルの dbt SQL コード

```
with opens as (
        select *
        from {{ ref('fct_email_opens') }}
), clicks as (
        select *
        from {{ ref('fct_email_clicks') }}
), opens_clicks_joined as (

    select
      o.lead_id as lead_id,
      o.campaign_id as campaign_id,
      o.email_send_id as email_send_id,
      o.activity_timestamp as open_ts,
      c.activity_timestamp as click_ts
    from opens as o
      left join clicks as c
      on o.email_send_id = c.email_send_id
      and o.lead_id = c.lead_id

)

select * from opens_clicks_joined
```

マーケティング分析モデル

これらは、マーケティングチームや営業チームが使用するダッシュボードやレポートに使用される最終的なマーケティング分析モデルです。

`country_email_engagement` モデルの dbt SQL コード

```
with accounts as (
        select
        account_id,
        billing_country
        from {{ ref('dim_account') }}
), users as (
        select
        lead_id,
        account_id
        from {{ ref('dim_user') }}
), opens_clicks_joined as (

    select * from {{ ref('int_email_open_clicks_joined') }}

), joined as (

        select *
        from users as u
        left join accounts as a
        on u.account_id = a.account_id
        left join opens_clicks_joined as oc
        on u.lead_id = oc.lead_id

)

select
        billing_country as country,
        count(open_ts) as opens,
        count(click_ts) as clicks,
        count(click_ts) / count(open_ts) as click_ratio
from joined
group by country
```

dbt モデルの実行とテスト

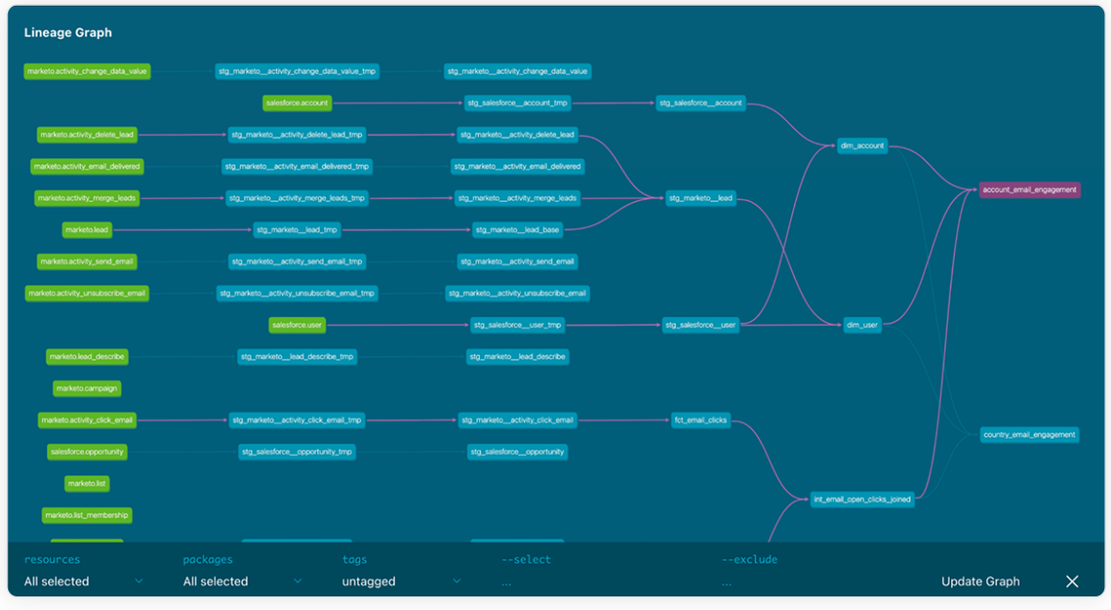
モデルの準備が整ったので、以下を使用して全てのモデルを実行できます。

```
dbt run
```

そして、以下を使用してテストを実行します。

```
dbt test
```

dbt のドキュメントとリネージグラフのビュー



マーケティング分析モデルの dbt リネージグラフ

モデルが正常に実行されたら、以下の方法でドキュメントと系統図を作成できます。

```
$ dbt docs generate
```

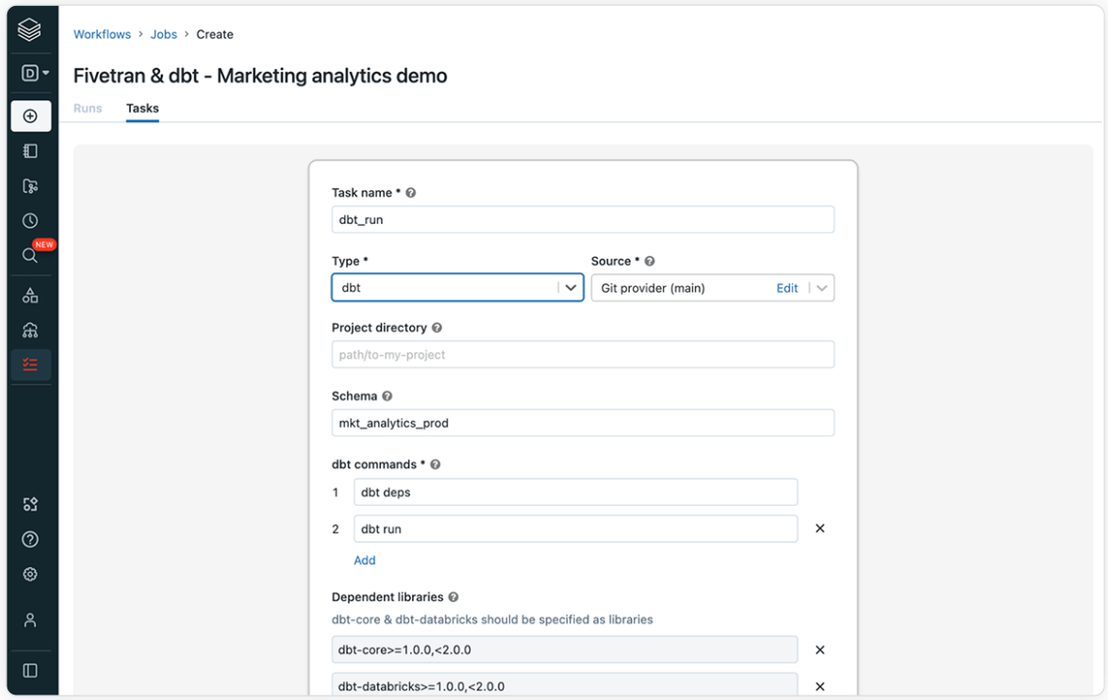
ローカルで表示するには次のようにします。

```
$ dbt docs serve
```

dbt モデルの本番環境へのデプロイ

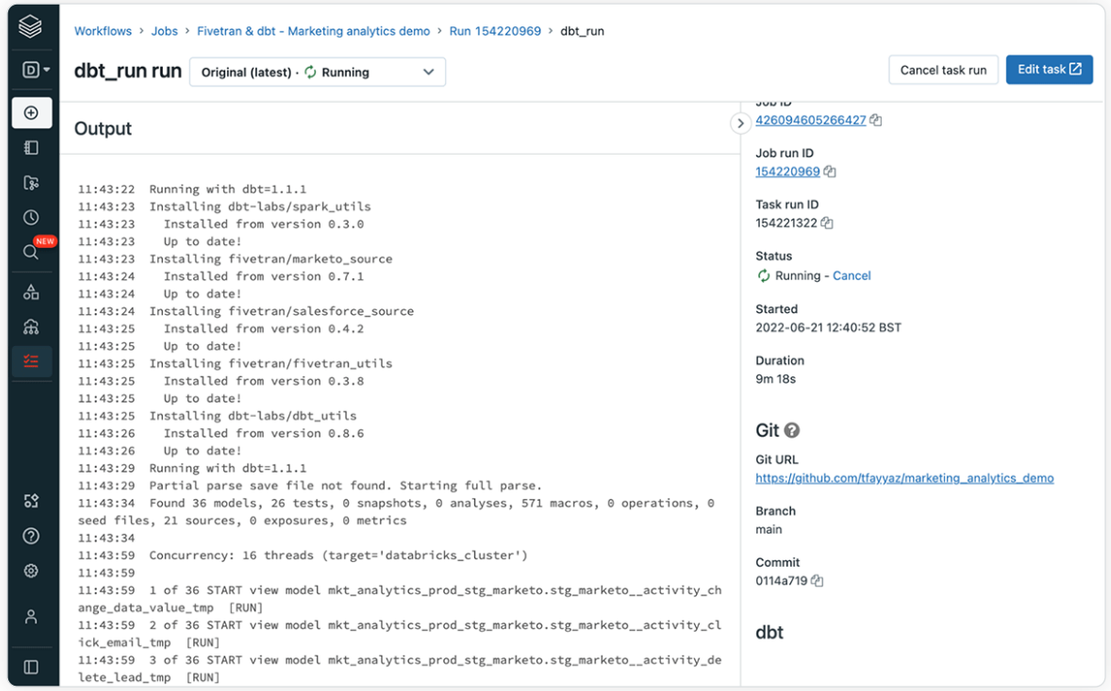
ローカルで dbt モデルを開発し、テストした後、本番環境にデプロイするための複数のオプションがあります。そのうちの1つが Databricks Workflows の新しい dbt タスクタイプです（プライベートプレビューでの提供）。

dbt プロジェクトは Git リポジトリで管理され、バージョン管理されている必要があります。Git リポジトリを指す dbt タスクを Databricks Workflows ジョブに作成できます。



Databricks ワークフローでの dbt タスクタイプの使用による dbt のオーケストレーション

dbt プロジェクトでパッケージを使用している場合、最初のコマンドは dbt deps、その後に最初のタスクでは dbt run、次のタスクでは dbt test と記述する必要があります。



各 dbt 実行の dbt ログの表示

dbt コマンドを実行するたびにログを見ることができ、デバッグや問題の修正に役立ちます。

Fivetran と dbt によるマーケティング分析の強化

このように、Fivetran と dbt を Databricks データインテリジェンスプラットフォームと共に使用することで、セットアップや管理が簡単で、どのようなデータモデリング要件にも柔軟に対応できる強力なマーケティング分析ソリューションを容易に構築できます。

Fivetran と dbt を Databricks と統合するためのドキュメントを参照し、marketing_analytics_demo プロジェクトのサンプルを再利用して、独自のソリューションの構築をすぐに始めることができます。

Databricks Workflows の dbt タスクタイプはプライベートプレビューで提供しています。dbt タスクタイプを試すには、Databricks アカウントエグゼクティブにお問い合わせください。

セクション 4.2

Databricks による保険金請求処理の自動化

スマート請求は、請求処理の取り込みから分析、意思決定までを自動化し、効率を向上させます。

執筆者 : Anindita Mahapatra、Marzi Rasooli

グローバルなコンサルタント会社である EY 社の最新レポートによると、保険業界の将来はますます**データ駆動型**になり、**分析が可能**なものになると予測されます。最近のクラウドへの注目により、先進的な技術インフラへのアクセスが増えています。しかし、ほとんどの組織では、これらの機能の実装と活用にサポートを必要としています。今こそ、価値を実現するためのサービスの運用に焦点を当てるべきです。

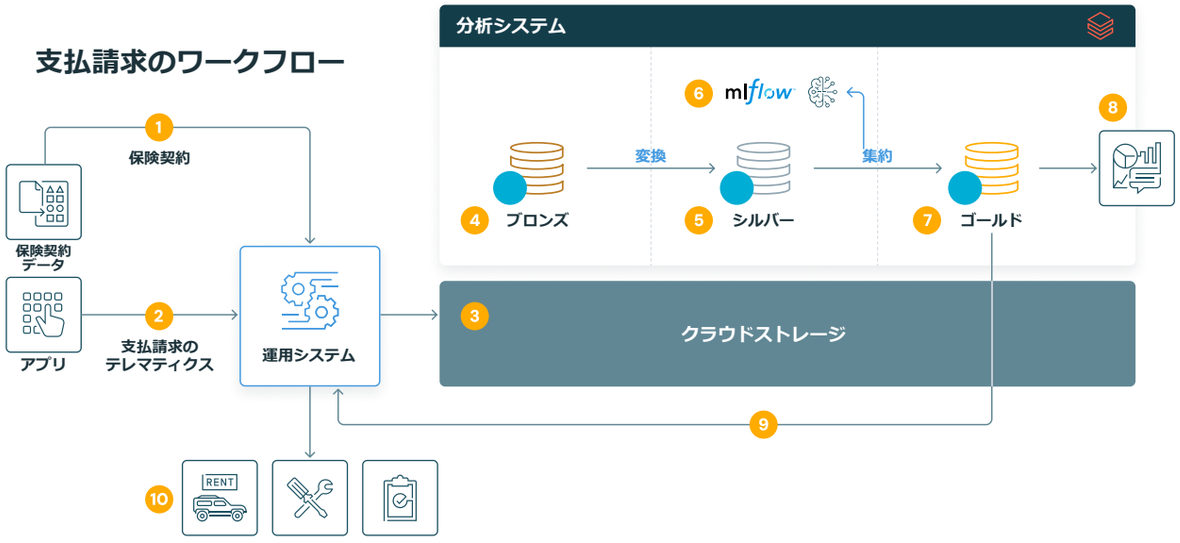
現在の経済状況において、保険会社はますます多くの課題に直面しています。保険会社はデータを活用し、加速度的に**イノベーション**を進める必要に迫られています。個人向け損害保険会社にとって、これは**パーソナライゼーション**とお客さまの囲い込みに一層注力することを意味します。ブランドロイヤルティはかつてないほど低下しており、お客さまはより割安な保険料や総合的により良い体験を求め、継続的に買い物をするため、離脱リスクが高まっています。不正請求の増加は利益率をさらに低下させます。保険会社はコストを削減し、リスクをより適切に管理するための新たな方法を見つける必要があります。

保険金請求処理プロセスの自動化と最適化は、時間の節約と人的資源への依存度の低下を通じてコストを大幅に削減できる分野の 1 つです。さらに、データと高度な分析から得られる知見を効果的に活用することで、リスクにさらされるリスクを全体的に大幅に減らすことができます。

「スマート請求」ソリューションアクセラレータの動機はシンプルです。レイクハウスによって請求処理プロセスを改善し、迅速な決済、処理コストの削減、不正の可能性がある請求に関する迅速なインサイトの取得を実現することです。レイクハウスパラダイムを導入することで、現在のアーキテクチャを簡素化し、将来的な組織全体への拡張を可能にします。関連アセットはこちらからご覧いただけます。

リファレンスアーキテクチャとワークフロー

一般的な保険金請求のワークフローでは、Guidewire のようなオペレーションシステムと Databricks のような分析システムとの間で、ある程度のオーケストレーションが行われます。次の図は、ある自動車保険会社のワークフローの例です。



スマート請求のリファレンスアーキテクチャとワークフロー

保険金請求処理プロセスの自動化と最適化には、お客さまと業務システムとのやり取りや、分析に利用できるさまざまな情報源について深く理解することが必要です。

この例では、お客さまは主にモバイルアプリケーションを通じて対話し、そこから請求書類を提出したり、既存のケースのステータスをモニターしたりすると想定しています。このタッチポイントは、顧客行動に関する重要な情報を提供します。もう一つの重要な情報源は、車両に搭載された IoT デバイスです。テレマティクスのデータを業務システムや分析システムにストリーミングすることで、運転行動やパターンに関する貴重なインサイトを得ることができます。その他の外部データソースには、車両の特性（メーカー、モデル、年式）、ドライバーの特性、エクスポージャー／補償（限度額、免責額）といった従来のデータカテゴリーを補完する天候や道路状況のデータが含まれる場合があります。

特に、信用情報機関のような従来の情報源からのデータがない場合、追加的なデータソースへのアクセスはますます重要になります。信用情報機関のクレジットスコアは通常、リスクモデリングの基礎となり、ドライバーのエクスポージャーを評価し、最終的に保険料に影響を与えます。一方、モバイルアプリケーションや IoT デバイスからのデータは、顧客行動をよりパーソナライズされた形で把握することができ、それを使って、ある人物に関連するリスクのより正確な指標を作成することができます。リスクモデリングとプライシングに対するこのような**行動ベース**のアプローチは、超個別化された顧客体験を提供するために不可欠です。

Databricks データインテリジェンスプラットフォームを搭載したレイクハウスアーキテクチャは、将来を見据えた保険金請求処理のプロセスをサポートするために必要な全ての機能とサービスを組み合わせた唯一のプラットフォームです。ストリーミングから機械学習、レポーティングまで、Databricks は明日の保険業界がエンドツーエンドのソリューションを構築するための最高のプラットフォームを提供します。

以下のステップは全体的な流れを表しています。

- ポリシーデータをインジェストする。
- テレマティクスデータは IoT センサーから継続的にインジェストされる。保険金請求者は、モバイルアプリを通じて保険金請求データを提出する。
- 運用データは全てクラウドストレージにインジェストされる。
- これを「未加工データ」として Delta ブロンズテーブルに段階的にロードする。
- データをさまざまなデータ変換を経て整理し、洗練する。
- 学習済みモデルを用いてデータをスコアリングする。
- 予測値をゴールドテーブルにロードする。
- 保険金請求のダッシュボードを視覚化のためにリフレッシュする。
- 得られたインサイトはオペレーションシステムにフィードバックされる。Guidewire からデータを引き出し、次善の策をリアルタイムで Guidewire に戻すフィードバックが提供され、どの請求が優先されるべきかを把握できる。
- 保険金請求の意思決定ワークフローは、これらの生成されたインサイトを使用して、ケースを適切にルーティングする。(例 : 修理費の承認、レンタル料の払い戻し、当局への警告など。)

レイクハウスのパラダイムがスマート請求を支援する方法

レイクハウスアーキテクチャは、全てのデータペルソナ (データエンジニア、データサイエンティスト、分析エンジニア、BI アナリスト) が単一のプラットフォーム上で共同作業を行うことを可能にします。あらゆるビッグデータワークロードとパラダイム (バッチ処理、ストリーミング、DataOps、ML、MLOps、BI など) を単一のコラボレーションプラットフォームでサポートすることで、アーキテクチャ全体が大幅に簡素化され、安定性が向上し、コストが大幅に削減されます。

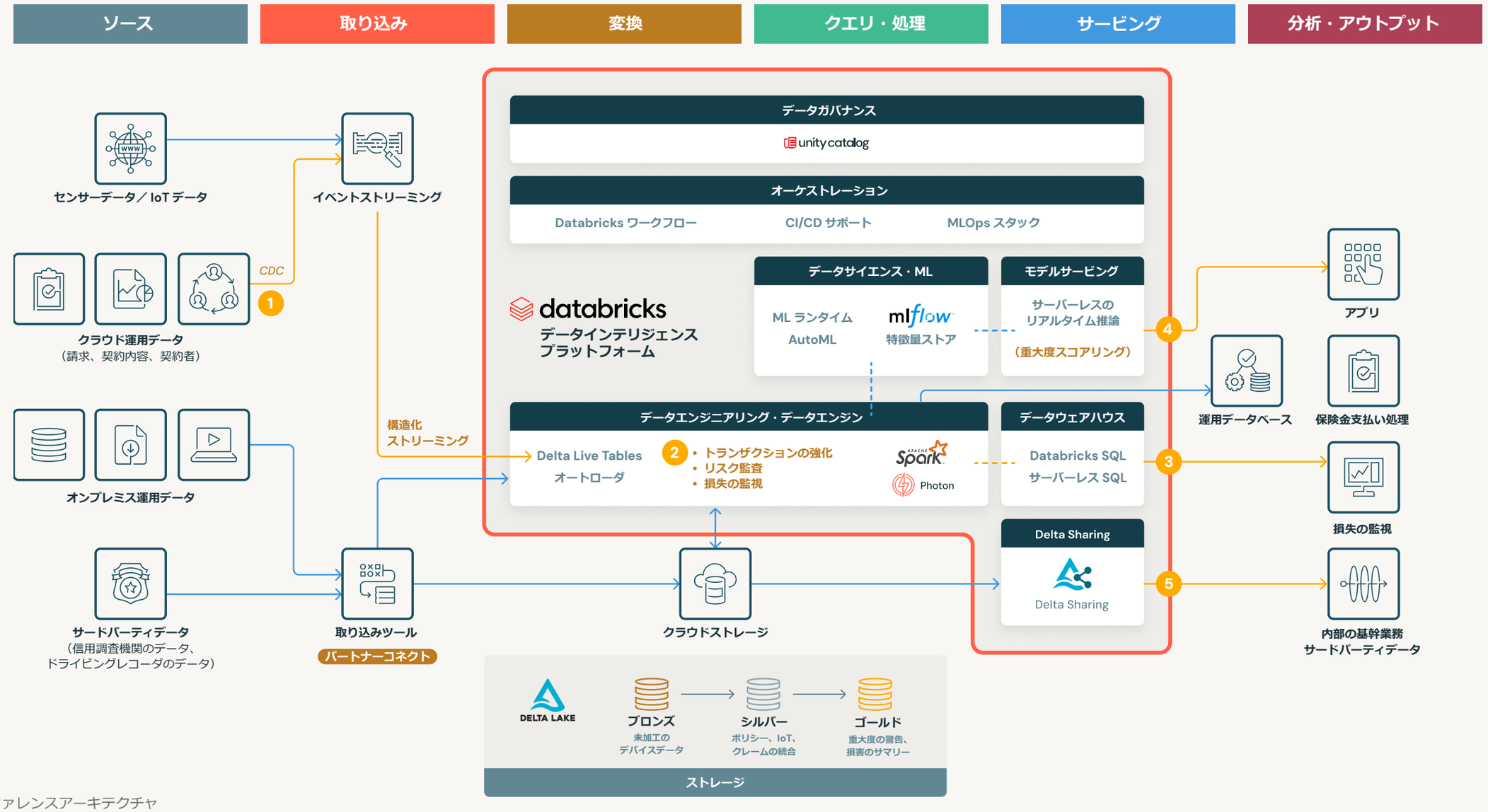
Databricks Delta Live Tables (DLT) パイプラインは、ワークロードを迅速に開発・実装するためのシンプルで宣言的なフレームワークを提供します。また、出力の整合性を保証するためのきめ細かな制約によるデータ品質管理をネイティブにサポートします。

ML と AI のワークロードは、MLflow で簡単に作成および管理でき、再現性と監査が容易になります。MLFlow は、実験からモデルのデプロイ、提供、アーカイブまで、モデルのライフサイクル全体を簡素化します。ML は、テキスト以外の非構造化データ (画像、音声、動画など) を含むあらゆる種類のデータに対して実行できます。このソリューションでは、車両の損傷を評価するためにコンピュータビジョン機能を使用します。

さらに、Databricks SQL は、キュレーションされ集約されたデータをクエリするための高速で効率的なエンジンを提供します。これらのインサイトは、数分以内にパッケージ化され、インタラクティブなダッシュボードを通じて提供されます。

Unity Catalog は、ファイル、テーブル、機械学習モデル、ダッシュボードを含む全てのデータと AI 資産に対して、検索、発見、ワークロードの自動リネージを組み込んだマルチクラウドの一元的なガバナンスソリューションを提供します。

以下の図は、一般的な保険業界のユースケースにおけるレイクハウスのリファレンスアーキテクチャを示しています。



保険業務のリファレンスアーキテクチャ

DLT とマルチタスクワークフローによるデータの取り込み

保険金請求処理プロセスの自動化は、インジェストとデータエンジニアリングワークフローの最適化から始まります。下図は、構造化、半構造化、非構造化など、一般的なデータソースをまとめたものです。ソースによっては更新が遅いものもあれば、更新が速いものもあります。さらに、追加を必要とする追加的なソースもあれば、インクリメンタルな更新を提供し、ゆっくりと変化するディメンションとして扱わなければならないソースもあります。

保険金請求処理で使用されるデータセットの例

DLT はデータ処理パイプラインを簡素化し、運用することができます。このフレームワークは、ストリーミングソースからの取り込みを容易にする Auto Loader、データ量の急激な変化に対応する効率的な自動スケーリング、タスク失敗時の再起動による耐障害性のサポートを提供します。

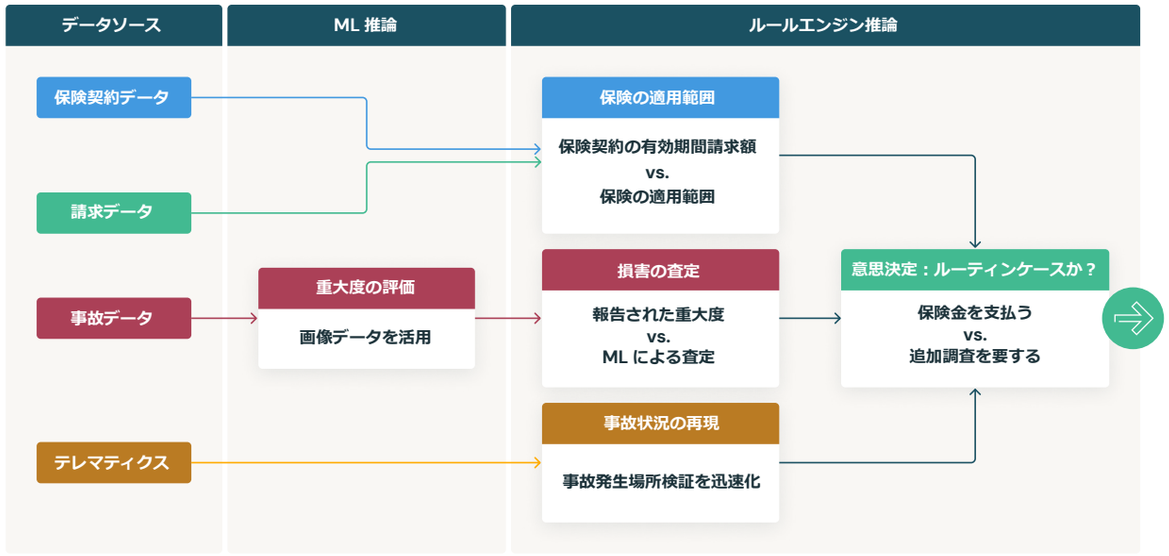
Databricks Workflows は、複数のタスクやワークロード（ノートブック、DLT、ML、SQL など）に対応できます。ワークフローはタスク間のリペア＆ランやコンピュート共有をサポートし、堅牢でスケーラブル、かつコスト効率の高いワークロードを実現します。さらに、ワークフローはスケジュールや REST API を介したプログラム呼び出しによって簡単に自動化できます。

ML とダイナミックルールエンジンを用いたインサイト生成

ML の活用は、これまで知られていなかったパターンを発見し、新たなインサイトを浮き彫りにし、疑わしい活動にフラグを立てるために不可欠です。しかし、ML と従来のルールベースのアプローチを組み合わせることで、さらに強力なものになります。

保険金請求処理プロセスの中で、ML はいくつかのユースケースに使用できます。一例として、自動車保険の請求で提出された画像の評価とスコアリングにコンピュータビジョンと ML を使用できます。モデルは損害の妥当性と重大性にフォーカスしてトレーニングできます。この場合、MLFlow はエンドツーエンドの MLOps 機能により、モデルのトレーニングと提供プロセスを簡素化するうえで非常に重要です。MLFlow は REST API を通じてサーバーレスでモデルを提供します。トレーニングされたモデルは、ボタンをクリックするだけで運用可能になり、本番環境に移行できます。

一方、ルールエンジンは、既知の運用特性や統計的チェックを定義する柔軟な方法を提供し、人手を介さずに自動化・適用することができます。データが事前に設定された期待値に適合しない場合は常にフラグが立てられ、人によるレビューと調査が行われます。このようなアプローチを ML ベースのワークフローに組み込むことで、さらなる監視が可能になり、請求に関する調査担当者がフラグを立てたケースを分析・検討するのに要する時間が大幅に短縮されます。



ML およびルールエンジンによる推論

ダッシュボードを使ったインサイトの可視化

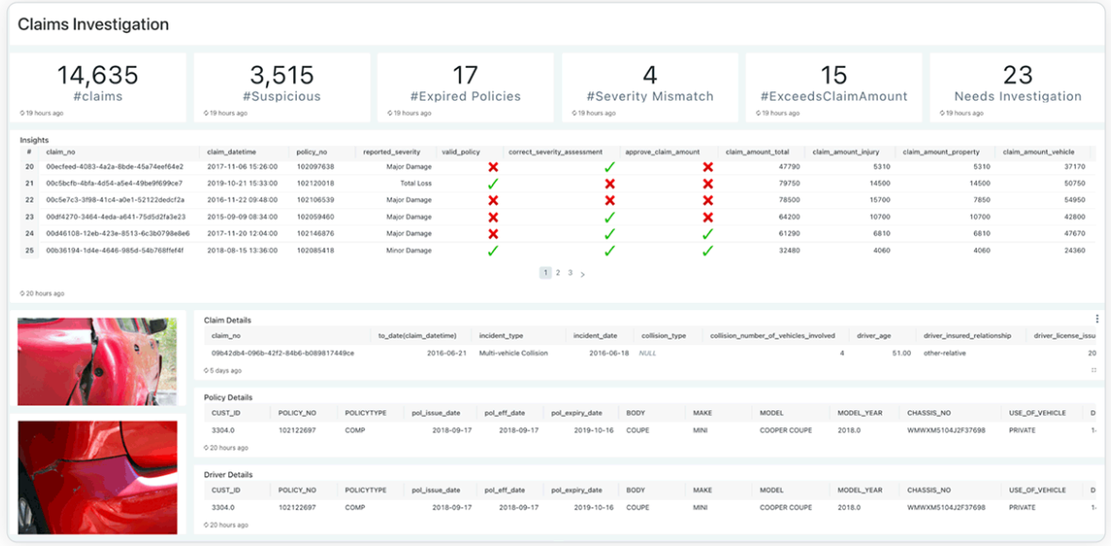
この例では、2 つのダッシュボードを作成し、重要なビジネスインサイトを把握します。ダッシュボードの内容は以下のとおりです。

- **損害概要ダッシュボード**：事業全体を大まかに把握できる
- **保険金請求調査ダッシュボード**：保険金請求の詳細を表示して特定のケースの詳細を把握できる

最近の傾向を分析することは、次のような類似のケースを検討するうえでさらに役立ちます。

- 損害率。損害率は、保険金支払額と調整費を総収入保険料で割ったものです。
  例：個人向け自動車保険の平均的な損害率（対人、対物、対物賠償の全ての補償を合計したもの）は約 65% です。
- 被害の程度別にインシデントの種類をカウントしたサマリーの可視化
- さまざまな特徴量／次元に関する傾向線
- ポリシーの地理的分布

保険金請求調査ダッシュボードでは、保険金請求に関する全ての関連情報が提供されるため、迅速な調査を可能にします。これにより、人間の調査員が特定のケースまで掘り下げて、損傷した車両の画像、保険の請求と契約内容、ドライバーの詳細、テレマティクスデータによる車両の走行経路、報告されたデータと査定されたデータインサイトを照合できます。



保険金請求調査ダッシュボード

ML 推論とルールエンジンを使用してパイプラインで自動スコアリングされた最近の保険金請求を提供します。

- 緑色のチェックは、自動査定が請求の説明と一致していることを示す。
- 赤色の✕印は、さらに手作業で調査する必要がある不一致を示す。

まとめ

保険会社が他社との差別化を図るためには、イノベーションとパーソナライゼーションが不可欠です。Databricks は、サードパーティのツールやサービスと容易に統合できる、オープンでセキュアかつ拡張可能なアーキテクチャにより、イノベーションを可能にし、加速させるデータインテリジェンスプラットフォームを保険会社に提供します。このソリューションアクセラレータは、このパラダイムが保険金支払業務にどのように適用できるかを示しています。さらに、Databricks のエコシステムは、データチームとビジネス関係者のコラボレーションを可能にするさまざまな機能を提供し、ビジネス上の意思決定をサポートし、収益に具体的な価値をもたらすインサイトを生成・共有します。

この例で使用したパイプラインの設定、モデル、サンプルデータなどの技術的なアセットは、こちらのページまたは、Git から直接アクセスできます。

セクション 4.3

金融サービスにおけるバッチ処理のデザインパターン

ワークフロー自動化の基盤構築

執筆者 : Eon Retief

はじめに

世界中の金融サービス機関（FSI）は、市場の変動や政情不安から法規制の変更に至るまで、これまでにない課題に直面しています。企業はデジタル変革プログラムの加速を余儀なくされ、重要なプロセスを自動化して運用コストを削減し、対応時間を短縮する必要があります。しかし、通常、データは複数のシステムに分散しているため、こうした取り組みを実行するために必要な情報にアクセスすることは容易ではないのが実情です。

デジタル変革が進んだビジネスにおいて、データ主導のユースケースを数多くサポートできるサービスのエコシステムを構築することは、不可能なタスクのように思えるかもしれません。このブログでは、最新のデータスタックの重要な側面の 1 つであるバッチ処理に焦点を当てます。一見、時代遅れのパラダイムに見えるバッチ処理が、なぜデータアーキテクチャの重要かつ実行可能なコンポーネントであり続けるのかを説明します。また、FSI が定期的なワークフローをサポートするためのインフラを構築する際に直面する重要な課題の解決に Databricks がどのように役立つかをご紹介します。

バッチによる取り込みが重要な理由

この 20 年間、世界的なインスタント社会へのシフトにより、組織は事業運営とエンゲージメントモデルの見直しを余儀なくされています。デジタルファースト戦略は、もはやオプションではなく、生き残るために不可欠です。お客さまのニーズや要求は、かつてない速さで変化し、進化しています。即座に満足したいという欲求は、リアルタイムの処理と意思決定をサポートする機能の構築がますます注目されています。この新しいダイナミックな世界において、バッチ処理はまだ適切なのか、という疑問が生じるかもしれません。

リアルタイムシステムやストリーミングサービスは、FSI がエッジの不安定な市場環境に機敏に対応するのに役立ちますが、通常、バックオフィス機能の要件を満たすものではありません。ほとんどのビジネス上の意思決定は、反応的なものではなく、むしろ考慮された戦略的な推論を必要とします。定義上、このアプローチでは、一定期間にわたって収集された集計データの体系的なレビューが必要です。このような状況でのバッチ処理は、集約された膨大な量のデータを処理するための最も効率的で費用対効果の高い方法です。さらに、バッチ処理はオフラインで行うことができるため、運用コストを削減し、エンドツーエンドのプロセスをより詳細に制御できます。

金融の世界は変化していますが、既存企業も新興企業も、中核となるビジネス機能を強化するためにバッチ処理に大きく依存し続けています。レポーティングやリスク管理、異常検知や監視のいずれにおいても、FSI は人的ミスを減らし、デリバリーのスピードを上げ、運用コストを削減するためにバッチ処理を必要としています。

開始するには

高い高度からの全体の概要や大まかな視点から始めると、ほとんどの FSI は、オンプレミスのシステム、クラウドベースのサービス、さらにはサードパーティのアプリケーションに散在する多数のデータソースを持つことになります。これら全ての接続に対応するバッチ取り込みフレームワークを構築するには、複雑なエンジニアリングが必要で、保守チームにとってはすぐに負担になります。これは、変更データの取り込み（CDC）、スケジューリング、スキーマの進化などを考慮する前の話です。このセクションでは、金融サービス向けのレイクハウスアーキテクチャ（LFS）とそのパートナーのエコシステムを活用することで、これらの主要な課題に対処し、アーキテクチャ全体を大幅に簡素化できることを示します。

レイクハウスアーキテクチャは、全ての分析およびサイエンス的なデータワークロードをサポートする統合プラットフォームを提供するように設計されています。図 1 は、最新のデータエコシステムをサポートする他のプラットフォームとの統合を容易にする、分離設計のリファレンスアーキテクチャを示しています。レイクハウスでは、データのソース、ボリューム、速度、宛先に関係なく動作するインジェストおよびサービングレイヤーを容易に構築できます。

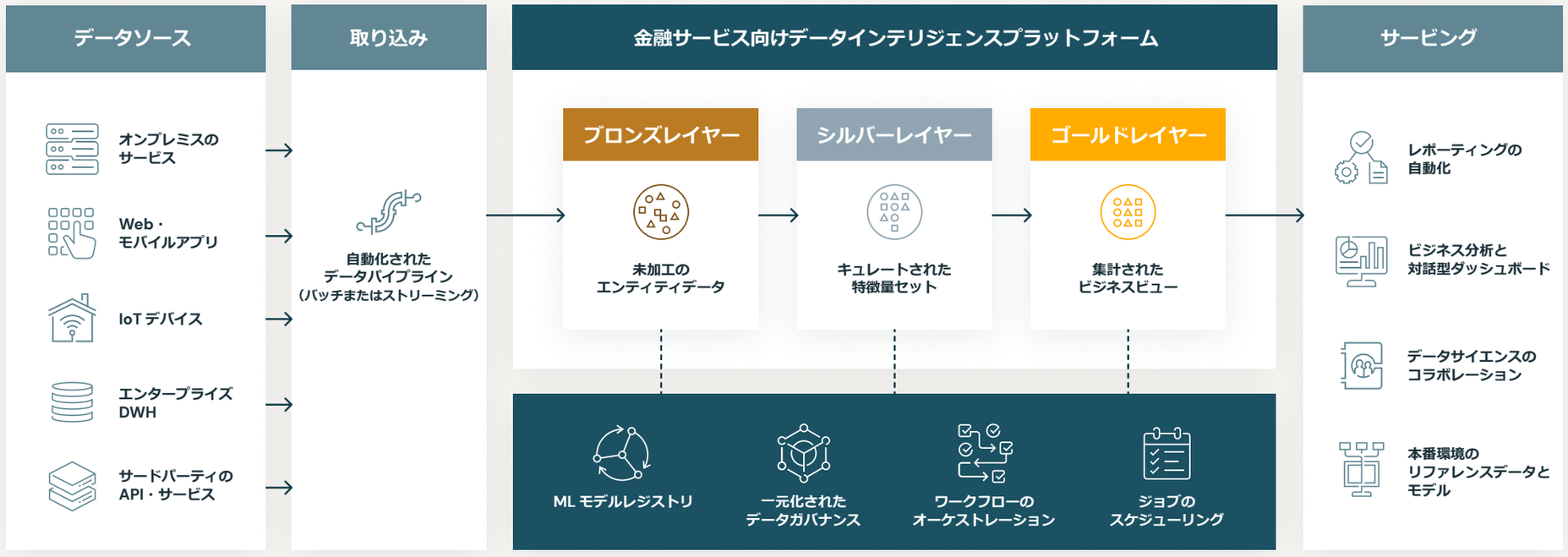


図 1: 金融サービス向けレイクハウスのリファレンスアーキテクチャ

LFS のパワーと効率性を実証するために、保険の世界に目を向けます。一般的な保険金請求のワークフローの基本的なレポート要件を検討します。このシナリオでは、保険会社は、保険金請求プロセスによって引き起こされる次のような主要なメトリクスに関心を持つ可能性があります。

- 有効契約数
- 保険金請求件数
- 保険金支払額
- エクスポージャー総額
- 損害率

さらに、不審な保険金請求の可能性や、インシデントの種類と重大度による内訳を確認したい場合もあります。これらの指標は全て、1) 保険契約、2) お客さまからの保険請求、の 2 つの主要なデータソースから簡単に計算できます。保険契約と請求の記録は通常、エンタープライズデータウェアハウス（EDW）と業務用データベースの組み合わせで保存されます。主な課題は、これらのソースに接続してデータをレイクハウスに取り込み、Databricks のパワーを活用して目的のアウトプットを計算することです。

幸いなことに、LFS の柔軟な設計により、さまざまな SaaS テクノロジーやツールのクラス最高の製品を活用して、容易に特定のタスクを処理できます。Databricks の保険金請求分析のユースケースで考えられるソリューションの1つは、バッチインジェストプレーンに Fivetran を使用することです。Fivetran は、多数のデータソースに接続し、Databricks データインテリジェンスプラットフォームに直接データを配信するためのシンプルでセキュアなプラットフォームを提供します。さらに、CDC、スキーマの進化、ワークロードスケジューリングをネイティブでサポートします。図 2 に、このユースケースに対する実用的なソリューションの技術アーキテクチャを示します。

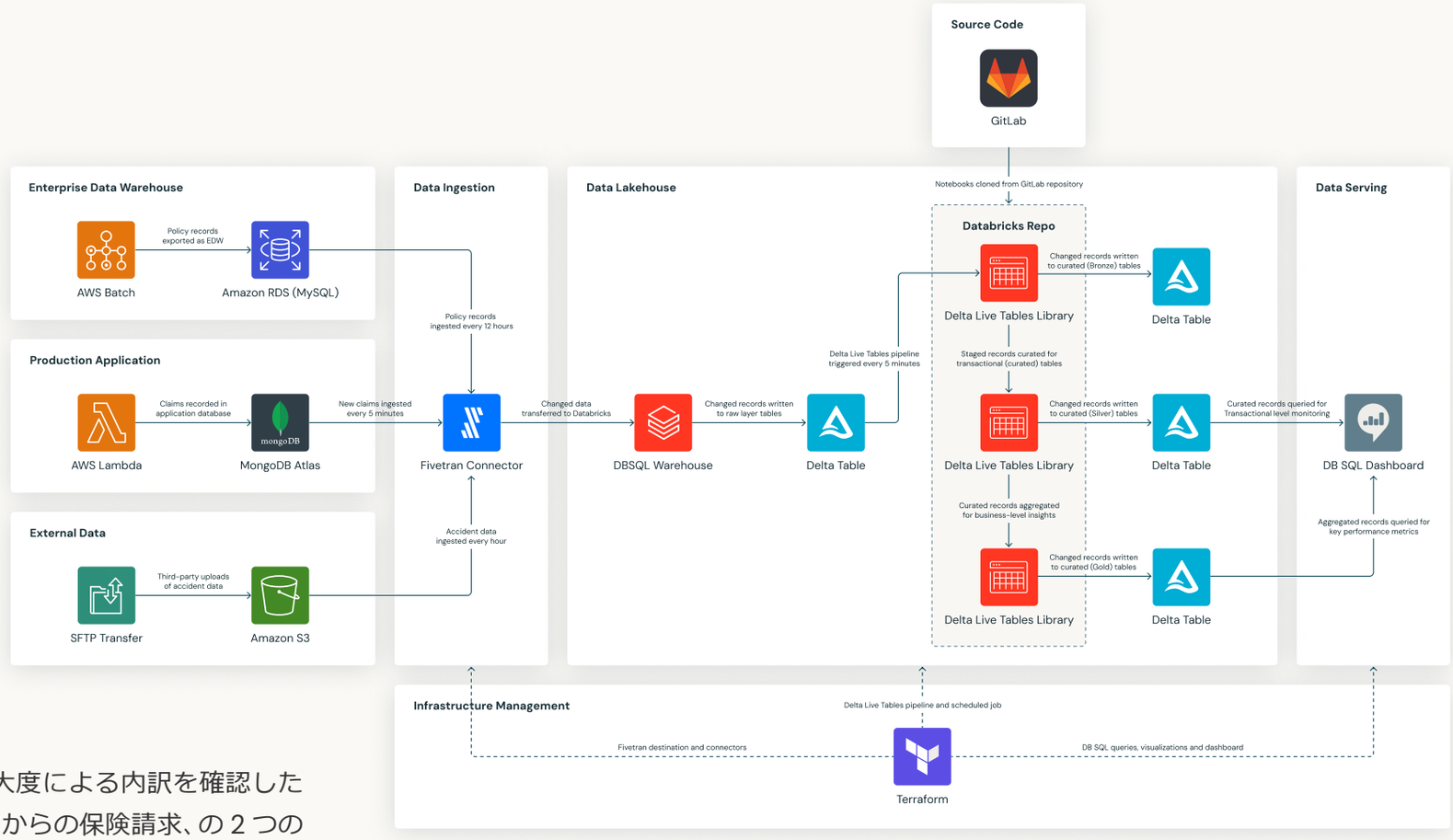


図 2 - シンプルな保険請求ワークフローの技術アーキテクチャ

データが取り込まれ、LFS に配信されると、エンジニアリングワークフロー全体に Delta Live Tables（DLT）を使用できます。DLT は、複雑なワークフローを自動化し、データ品質管理を実施するための、シンプルでスケーラブルな宣言型フレームワークを提供します。DLT ワークフローからの出力、つまりキュレーションされ集約されたアセットは、Databricks SQL（DB SQL）を使用して照会できます。DB SQL は、LFS にデータウェアハウスを導入し、ビジネスクリティカルな分析ワークロードを強化します。DB SQL クエリの結果は、使いやすいダッシュボードにパッケージ化され、ビジネスユーザーに提供されます。

ステップ 1: 取り込みレイヤーの作成

Fivetran で取り込みレイヤーを設定するには、2 段階のプロセスが必要です。第一に、データを配信するいわゆる配信先を設定し、第二に、ソースシステムとの 1 つ以上の接続を確立します。パートナーコネクトインターフェースは、Fivetran と Databricks ウェアハウスを接続するためのシンプルなガイド付きインターフェースで、最初のステップを行います。Fivetran はウェアハウスを使用して未加工のソースデータを Delta テーブルに変換し、結果を Databricks データインテリジェンスプラットフォームに格納します。図 3 と図 4 は、パートナーコネクトと Fivetran のインターフェースから新しい接続先を設定する手順を示しています。

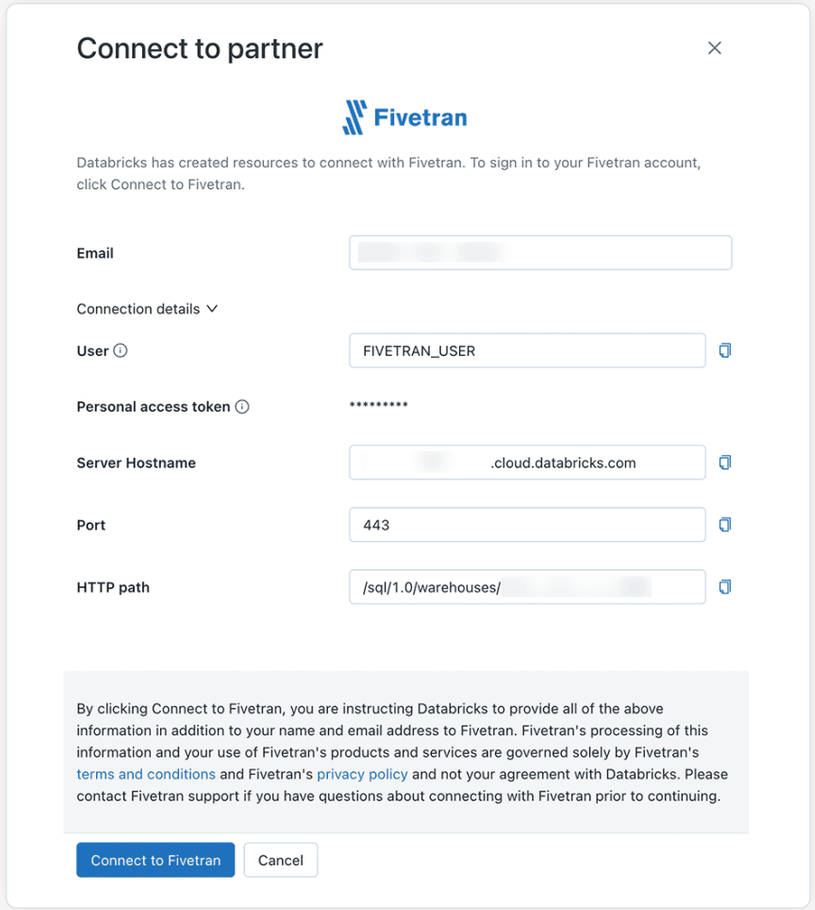


図 3 : 新しい接続を作成するための Databricks Partner Connect インターフェース

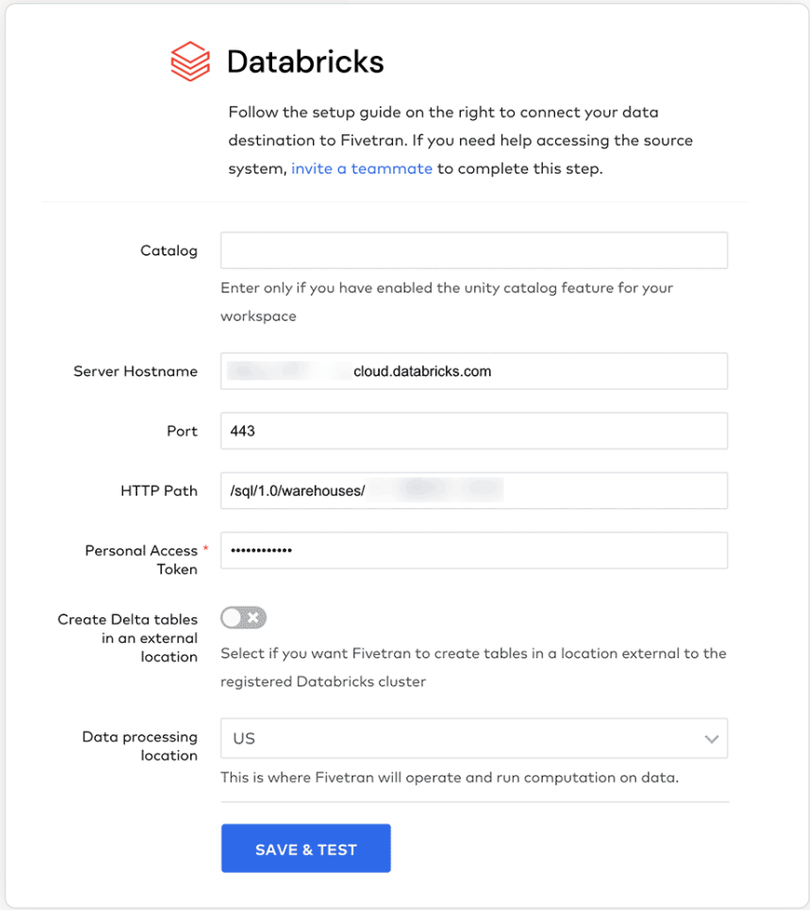


図 4 : 新しい接続先を確認するための Fivetran インターフェース

次のステップでは、Fivetran インターフェースに移動します。ここから、複数の異なるソースシステムへの接続を容易に作成・設定できます（サポートされている全ての接続の完全なリストについては、公式ドキュメントを参照してください）。この例では、3 つのデータソースを考えます。1) ODS（Operational Data Store）または EDW（Enterprise Data Warehouse）に保存されたポリシーレコード、2) オペレーショナルデータベースに保存された保険金請求レコード、3) blob ストレージに配信された外部データ。このように、Fivetran では 3 つの異なる接続を設定する必要があります。それぞれについて、Fivetran のシンプルなガイドプロセスに従って、ソースシステムとの接続を設定します。図 5 と図 6 は、データソースへの新しい接続を設定する方法を示しています。

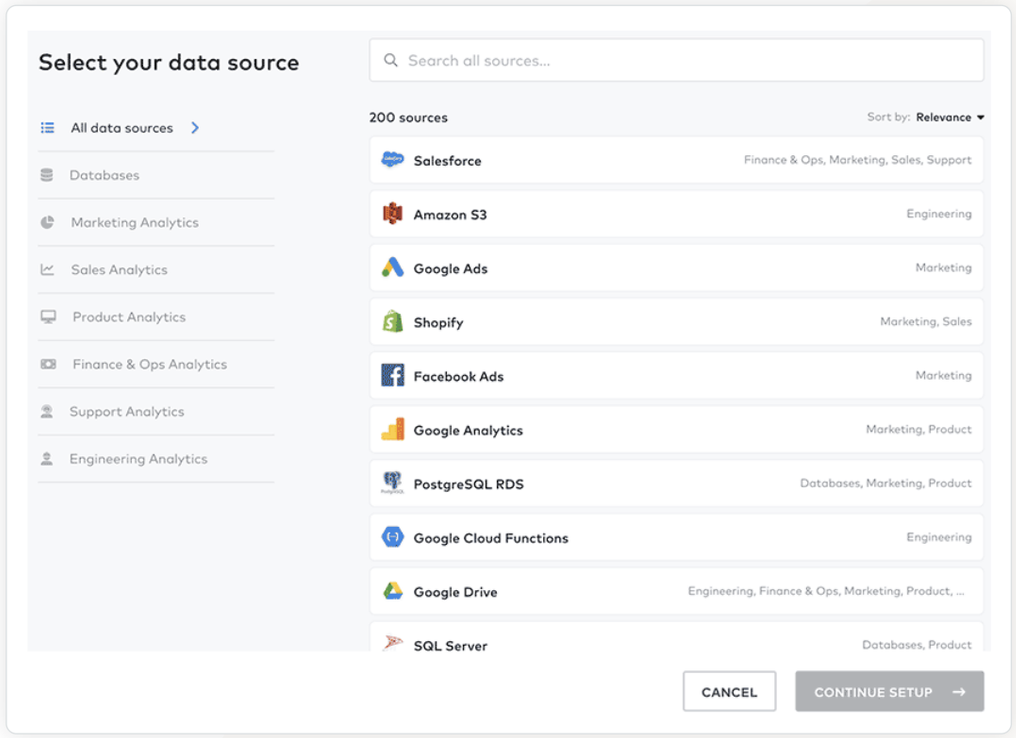


図 5 : データソースタイプを選択するための Fivetran インターフェース

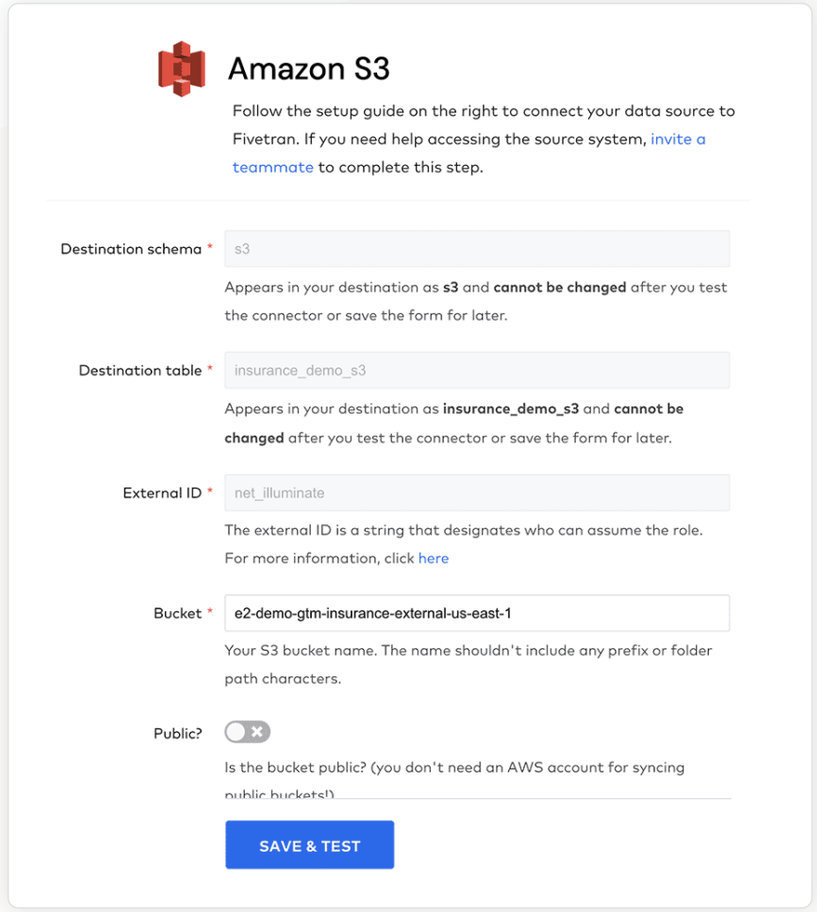


図 6 : データソース接続を構成するための Fivetran インターフェース

接続が検証されると、さらに接続を設定できます。設定する重要なオプションの 1 つは、Fivetran がソースシステムに新しいデータを問い合わせる頻度です。図 7 では、Fivetran が同期頻度を 5 分から 24 時間の範囲で簡単に設定できるようにしたことがわかります。

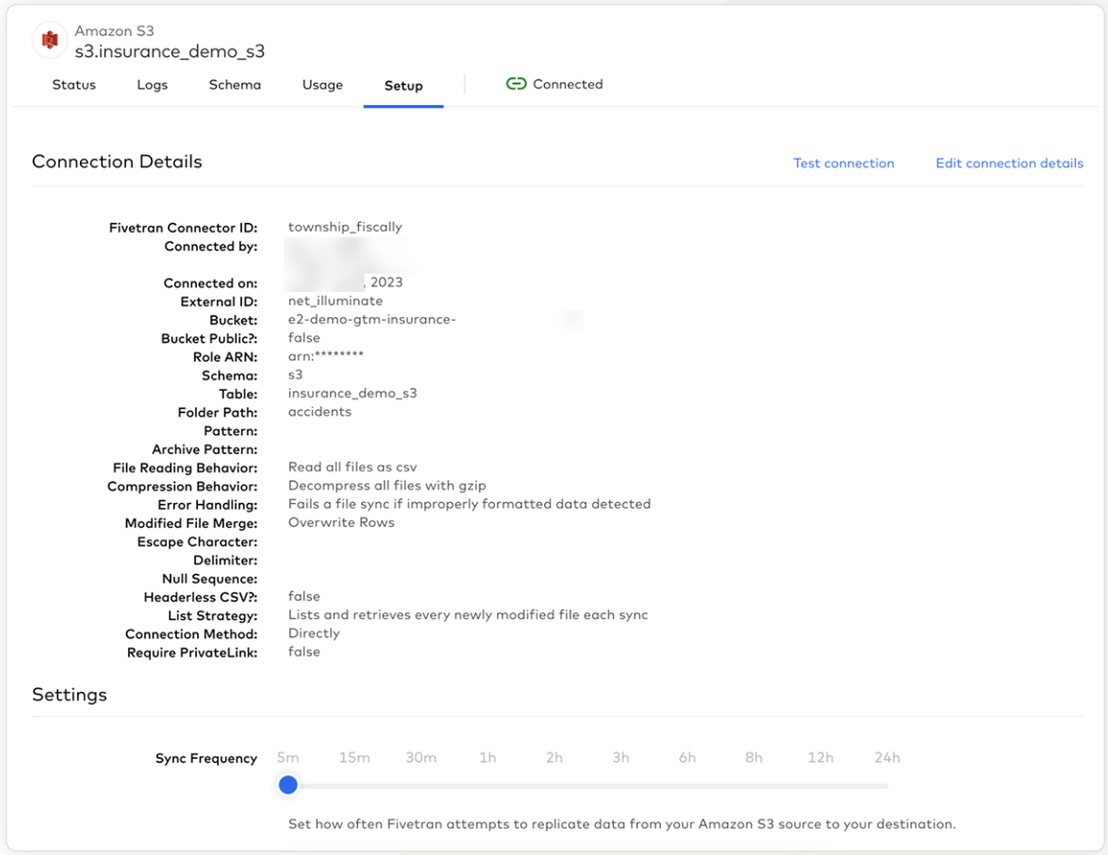


図 7 : Fivetran コネクタの構成概要

Fivetran は、接続が認証されると、直ちにソースシステムからデータを照会して取り込みます。データは Delta テーブルとして保存され、Databricks のカタログエクスプローラから見ることができます。デフォルトでは、Fivetran は全てのデータを Hive メタストアの下に格納します。新しい接続ごとに新しいスキーマが作成され、各スキーマには少なくとも 2 つのテーブルが含まれます。1 つはデータを含むテーブル、もう 1 つは取り込みサイクルごとのログを含むテーブルです（図 8 参照）。

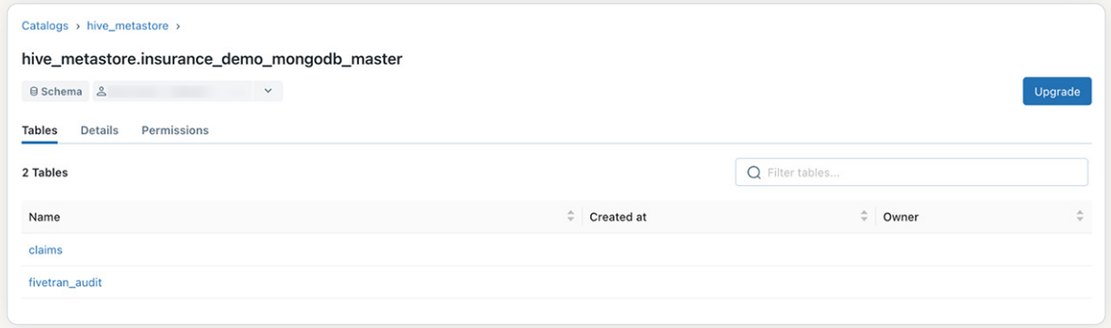


図 8 : Fivetran によって Databricks ウェアハウスに作成された接続例のテーブルの概要

データを Delta テーブルに格納することは、大きな利点です。Delta Lake はきめ細かなデータバージョニングをネイティブでサポートしており、各取り込みサイクルをタイムトラベルできます（図 9 参照）。DB SQL を使用して特定のバージョンのデータを照会し、ソースレコードがどのように変化したかを分析できます。

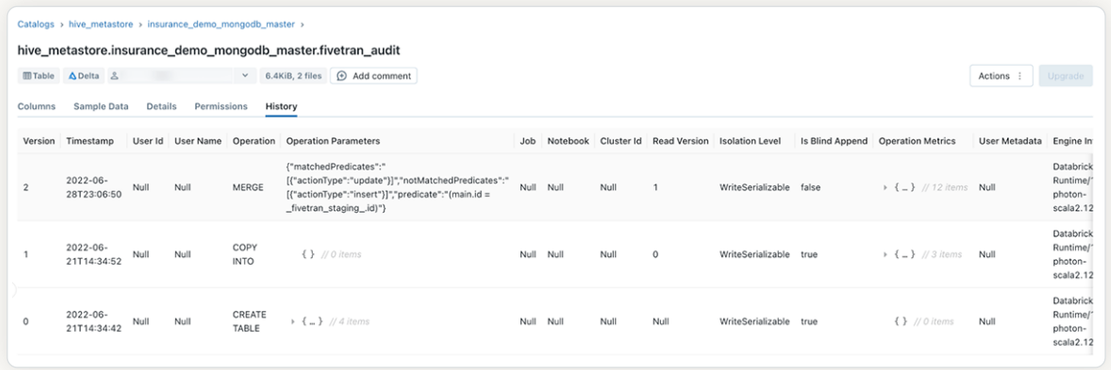


図 9 : Fivetran 監査テーブルに加えられた変更を示す履歴のビュー

ソースデータに半構造化または非構造化値が含まれている場合、これらの属性は変換処理中に平坦化されることに注意することが重要です。これは、結果がグループ化されたテキストタイプの列に保存されることを意味し、これらのエンティティは、キュレーションプロセスで DLT を使用して分離し、解凍して個別の属性を作成する必要があります。

ステップ 2：ワークフローの自動化

レイクハウスのデータで Delta Live Tables（DLT）を使用して、シンプルで自動化されたデータエンジニアリングワークフローを構築できます。DLT は、詳細な特徴量エンジニアリングのステップを指定するための宣言的なフレームワークを提供します。現在、DLT は Python と SQL の両方の API をサポートしています。この例では、Python API を使用してワークフローを構築します。

DLT の最も基本的な構成要素はテーブルの定義です。DLT は全てのテーブル定義に問い合わせて、データがどのように処理されるべきかの包括的なワークフローを作成します。例えば Python では、テーブルは関数定義と `dlt.table` デコレータを使って作成されます（以下の Python コードの例を参照）。このデコレータを使用して、生成されるテーブルの名前、テーブルの目的を説明するコメント、テーブルのプロパティを指定します。

```
@dlt.table(
   name             = “curated_claims”,
   comment          = “Curated claim records”,
   table_properties = {
       “layer”: “silver”,
       “pipelines.autoOptimize.managed”: “true”,
       “delta.autoOptimize.optimizeWrite”: “true”,
       “delta.autoOptimize.autoCompact”: “true”
   }
)
def curate_claims():
   # Read the staged claim records into memory
   staged_claims = dlt.read(“staged_claims”)
   # Unpack all nested attributes to create a flattened table structure
   curated_claims = unpack_nested(df = staged_claims, schema = schema_claims)

   ...
```

特徴量エンジニアリングの指示は、標準の PySpark API とネイティブの Python コマンドを使用して関数本体内で定義します。次の例では、PySpark が保険金請求のレコードとポリシーテーブルのデータを結合して、保険金請求の単一のキュレーションビューを作成する方法を示しています。

```
    ...

    # Read the staged claim records into memory
    curated_policies = dlt.read("curated_policies")
    # Evaluate the validity of the claim
    curated_claims = curated_claims \
        .alias("a") \
        .join(
            curated_policies.alias("b"),
            on  = F.col("a.policy_number") == F.col("b.policy_number"),
            how = "left"
        ) \
        .select([F.col(f"a.{c}") for c in curated_claims.columns] + [F.col(f"b.
{c}").alias(f"policy_{c}") for c in ("effective_date", "expiry_date")]) \
        .withColumn(
            # Calculate the number of months between coverage starting and the
claim being filed
            "months_since_covered", F.round(F.months_between(F.col("claim_date"),
F.col("policy_effective_date")))
        ) \
        .withColumn(
            # Check if the claim was filed before the policy came into effect
            "claim_before_covered", F.when(F.col("claim_date") < F.col("policy_
effective_date"), F.lit(1)).otherwise(F.lit(0))
        ) \
        .withColumn(
            # Calculate the number of days between the incident occurring and the
claim being filed
            "days_between_incident_and_claim", F.datediff(F.col("claim_date"),
F.col("incident_date"))
        )

    # Return the curated dataset
    return curated_claims
```

DLT の大きな利点の一つは、データ品質標準を指定し、適用できることです。各 DLT テーブルに対して、テーブルのコンテンツに適用されるべき詳細なデータ品質制約を期待値として設定することができます。現在、DLT は 3 つの異なるシナリオの期待値をサポートしています。

デコレータ	説明
`expect`	期待値に反する記録の保持
`expect_or_drop`	期待値に反する記録は削除
`expect_or_fail`	いずれかのレコードが制約に違反した場合、実行を停止

期待値は、1 つまたは複数のデータ品質制約で定義できます。各制約は、説明と評価する Python 式または SQL 式を必要とします。`expect_all`、`expect_all_or_drop`、`expect_all_or_fail` デコレータを使用して、複数の制約を定義できます。それぞれのデコレータは、制約の説明をキーとし、それぞれの式を値とする Python 辞書を期待します。以下の例は、上記の retain と drop のシナリオに対する複数のデータ品質制約を示しています。

```
@dlt.expect_all({
    "valid_driver_license": "driver_license_issue_date > (current_date() -
cast(cast(driver_age AS INT) AS INTERVAL YEAR))",
    "valid_claim_amount": "total_claim_amount > 0",
    "valid_coverage": "months_since_covered > 0",
    "valid_incident_before_claim": "days_between_incident_and_claim > 0"
})
@dlt.expect_all_or_drop({
    "valid_claim_number": "claim_number IS NOT NULL",
    "valid_policy_number": "policy_number IS NOT NULL",
    "valid_claim_date": "claim_date < current_date",
    "valid_incident_date": "incident_date < current_date",
    "valid_incident_hour": "incident_hour between 0 and 24",
    "valid_driver_age": "driver_age > 16",
    "valid_effective_date": "policy_effective_date < current_date()",
    "valid_expiry_date": "policy_expiry_date <= current_date()"
})
def curate_claims():
    ...
```

複数の Databricks ノートブックを使用して、DLT テーブルを宣言できます。メダリオンアーキテクチャに従うと仮定すると、例えば、ブロンズ、シルバー、ゴールドのレイヤーを構成するテーブルを定義するために、異なるノートブックを使用できます。DLT フレームワークは、複数のノートブックで定義された命令を消化し、単一のワークフローを作成できます。図 10 は、請求例の完全なワークフローを示しています。DLT は 3 つのソーステーブルから開始し、ビジネスで使用する 13 のテーブルを提供する包括的なパイプラインを構築します。

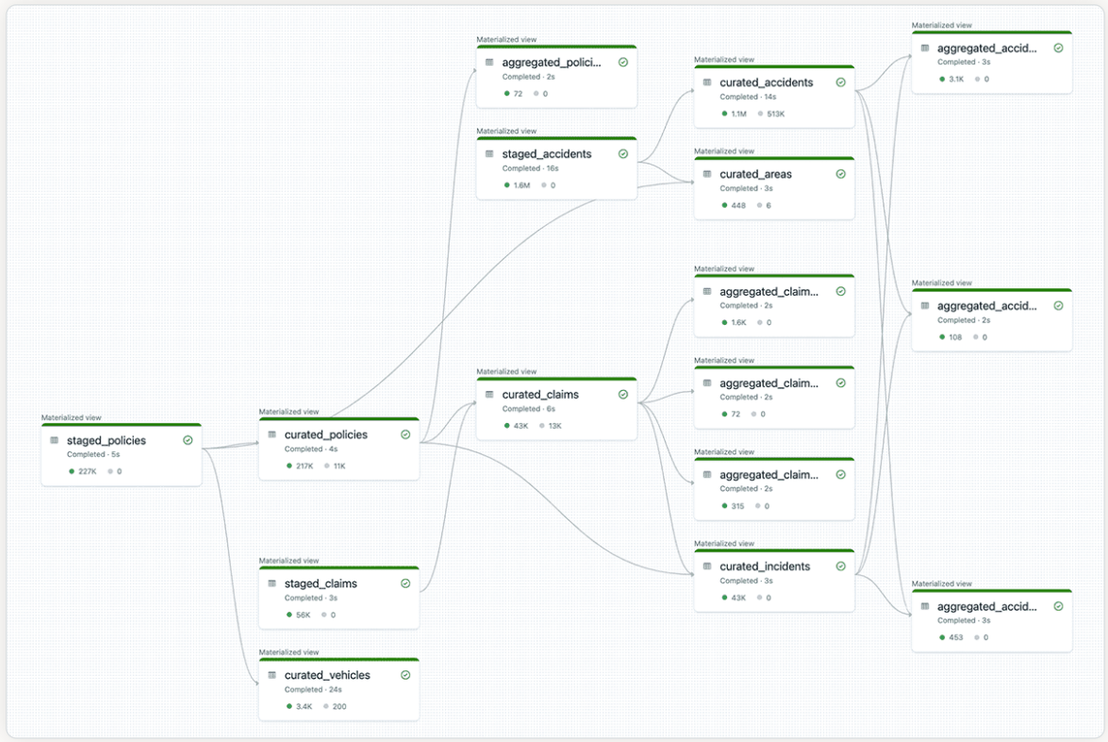


図 10 : 完全な Delta Live Tables (DLT) ワークフローの概要

各テーブルの結果は、必要なエンティティを選択することで閲覧できます。図 11 は、管理された請求テーブルの結果の一例です。DLT は、データ品質管理から得られた結果のハイレベルな概要を提供します。

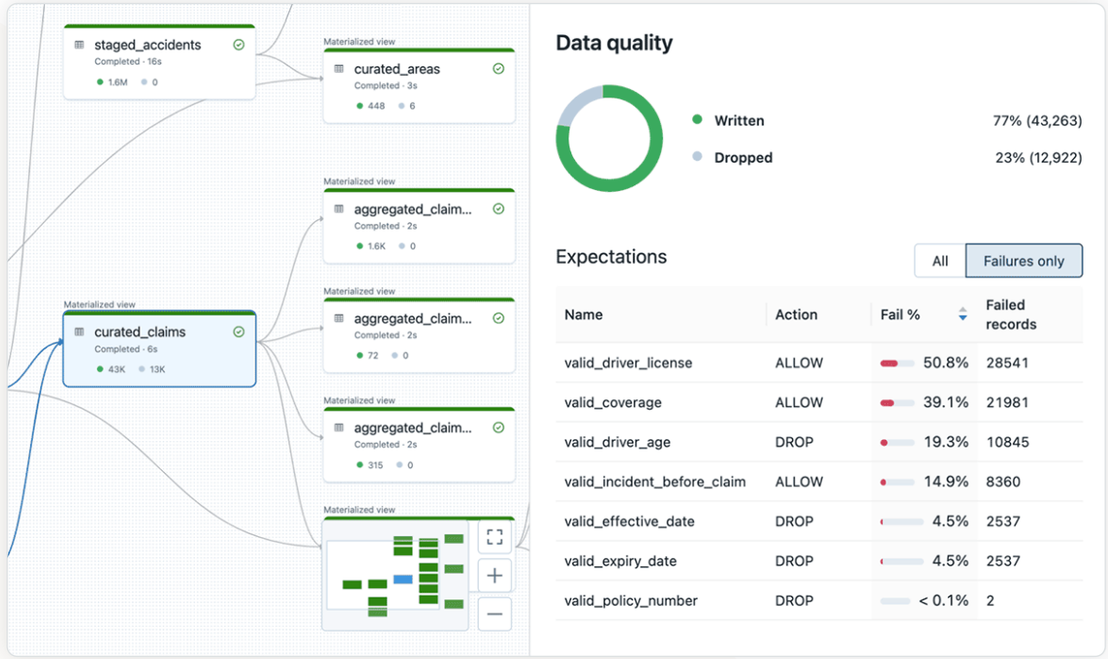


図 11 : Delta Live Tables (DLT) テーブルエンティティの詳細表示と関連するデータ品質レポートの例

データ品質期待値の結果は、イベントログをクエリすることでさらに分析できます。イベントログには、ワークフローパイプラインに定義された全ての期待値に関する詳細な指標が含まれています。以下のクエリは、期待値に合格したレコード数または不合格になったレコード数を含む、最後のパイプライン更新からの主要な指標を表示するための例を示しています。

```
SELECT
  row_expectations.dataset AS dataset,
  row_expectations.name AS expectation,
  SUM(row_expectations.passed_records) AS passing_records,
  SUM(row_expectations.failed_records) AS failing_records
FROM
  (
    SELECT
      explode(
        from_json(
          details :flow_progress :data_quality :expectations,
          “array<struct<name: string, dataset: string, passed_records: int,
failed_records: int>>”
        )
      ) row_expectations
    FROM
      event_log_raw
    WHERE
      event_type = ‘flow_progress’
      AND origin.update_id = ‘${latest_update.id}’
  )
GROUP BY
  row_expectations.dataset,
  row_expectations.name;
```

繰り返しになりますが、Delta 履歴ログを見ることで、各 DLT テーブルに加えられた変更の完全な履歴を見ることができます（図 12 参照）。これにより、時間の経過とともにテーブルがどのように変化していくかを理解し、パイプラインが失敗した場合に更新の完全なスレッドを調査できます。

Columns | Sample Data | Details | Permissions | **History**

Version	Timestamp	User Id	User Name	Operation	Operation Parameters	Job	Notebook	Cluster Id	Read Version	Isolation Level	Is Blind Append	Operation Metrics	User Metadata	Engine Info
12	2023-01-22T19:10:09	Null	Null	WRITE	{ … } // 1 item	Null	Null	Null	11	WriteSerializable	false	{ … } // 3 items	Null	Databricks-Runtime/dlt:11.0-delta-pipelines-1eca0d9-750b289-9ea72db-custom-local
11	2022-12-09T11:48:23	Null	Null	WRITE	{ … } // 1 item	Null	Null	Null	10	WriteSerializable	false	{ … } // 3 items	Null	Databricks-Runtime/dlt:11.0-delta-pipelines-ed5cc83-e81c5c7-17c692e-custom-local
10	2022-11-08T19:48:31	Null	Null	WRITE	{ … } // 1 item	Null	Null	Null	9	WriteSerializable	false	{ … } // 3 items	Null	Databricks-Runtime/dlt:11.0-delta-pipelines-de92f9e-8a33b70-a333f54-custom-local

図 12 : Delta Live Tables（DLT）テーブルエンティティに加えられた変更履歴の表示

さらに、変更データキャプチャ（CDC）を使用して、ソースデータセットの変更に基づいてテーブルを更新できます。DLT CDC は、SCD（Slow- changing dimensions）タイプ 1 および 2 のテーブルの更新をサポートします。

DLT パイプラインをトリガーするバッチプロセスには、2 つのオプションがあります。Databricks の Auto Loader を使用して、ソーステーブルに新しいデータが到着するたびにインクリメンタルに処理するか、設定した時間または間隔でトリガーするスケジュールジョブを作成するかです。この例では、5 分ごとに DLT パイプラインを実行するスケジュールジョブで、後者を選択しました。

アウトプットの運用

データを段階的に効率的に処理する能力は、方程式の半分でしかありません。DLT ワークフローからの結果は、運用化され、ビジネスユーザーに提供されなければなりません。この例では、DLT パイプラインからのアウトプットを、アドホック分析または対話型ダッシュボードを通じてあらかじめ用意されたインサイトを通じて利用ができます。

アドホック分析

Databricks SQL（DB SQL）は、レイクハウスアーキテクチャ上に効率的で費用対効果の高いデータウェアハウスを提供します。これにより、SQL ワークロードをソースデータに対して直接実行でき、他の方法よりもコストパフォーマンスが最大 12 倍向上します。

DB SQL を活用することで、分類および集計されたテーブルに対して特定のアドホッククエリを実行できます。例えば、総露出量を計算するキュレーションされたポリシーテーブルに対してクエリを実行することができます。DB SQL クエリエディタは、このようなクエリを構築し実行するためのシンプルで使いやすいインターフェースを提供します（次の例を参照）。

```
SELECT
  round(curr.total_exposure, 0) AS total_exposure,
  round(prev.total_exposure, 0) AS previous_exposure
FROM
  (
    SELECT
      sum(sum_insured) AS total_exposure
    FROM
      insurance_demo_lakehouse.curated_policies
    WHERE
      expiry_date > '{{ date.end }}'
      AND (effective_date <= '{{ date.start }}'
        OR (effective_date BETWEEN '{{ date.start }}' AND '{{ date.end }}'))
  ) curr
  JOIN
  (
    SELECT
      ...
```

DB SQL クエリエディタを使用して、異なるバージョンの Delta テーブルに対してクエリを実行することもできます。例えば、特定の日時の請求記録を集計したビューに対してクエリを実行することができます（以下の例を参照）。さらに DB SQL を使用して、異なるバージョンの結果を比較し、それらのステート間で変更されたレコードのみを分析できます。

```
SELECT
  *
FROM
  insurance_demo_lakehouse.aggregated_claims_weekly TIMESTAMP AS OF '2022-06-
05T17:00:00';
```

DB SQL は、サーバーレスコンピューティングエンジンを使用するオプションを提供し、コストを最小限に維持しながら、クラウドインフラの構成、管理、スケーリングの必要性を排除します。また、代替 SQL ワークベンチ（DataGrip など）と統合できるため、アナリストは任意のツールを使用してデータを探索し、インサイトを取得できます。

ビジネスインサイト

これで、DB SQL クエリを使用してクエリ結果の上にリッチなビジュアライゼーションを作成できます。これらをパッケージ化し、インタラクティブなダッシュボードを通じてエンドユーザーに提供できます（図 13 参照）。

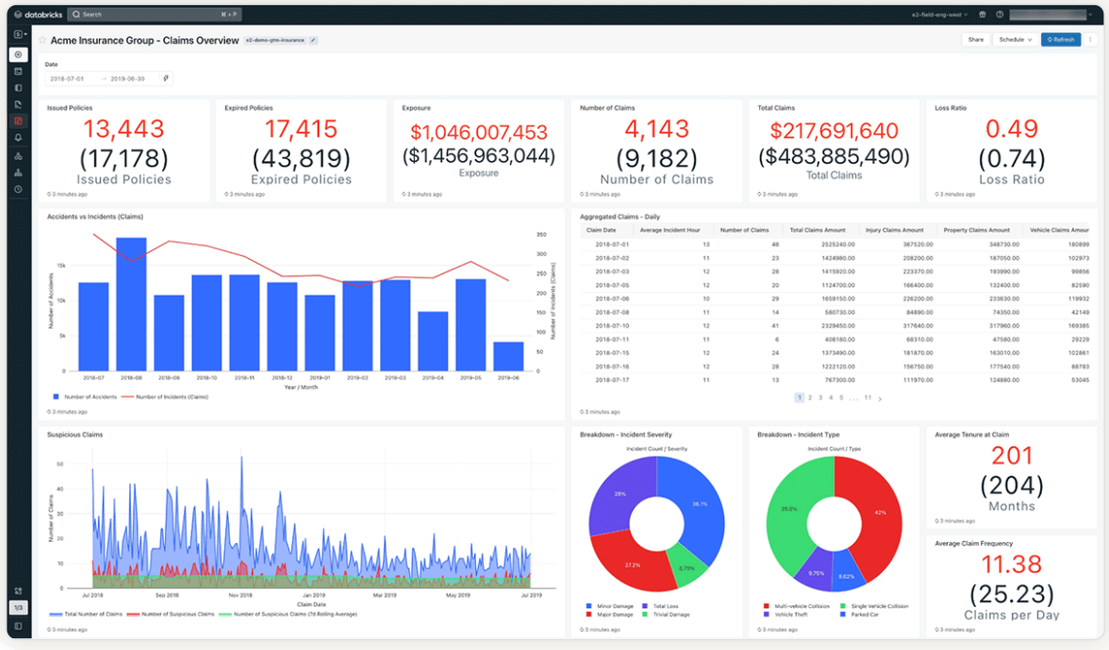


図 13 : Delta Live Tables（DLT）テーブルエンティティの結果に基づいて構築された運用ダッシュボードの例

このユースケースでは、主要な指標、ローリング計算、ハイレベルなブレイクダウン、および集計ビューを集めたダッシュボードを作成しました。このダッシュボードでは、保険金請求処理の完全な要約を一目で見ることができます。また、特定の日付範囲を指定するオプションも追加しました。DB SQL は、実行時にクエリに値を代入できるさまざまなクエリパラメータをサポートしています。これらのクエリパラメータはダッシュボードレベルで定義できるため、関連する全てのクエリが適宜更新されるようになります。

DB SQL は、Power BI、Tableau、Looker をはじめとする数多くのサードパーティの分析・BI ツールと統合しています。Fivetran で行ったように、パートナーコネクトを使用して外部プラットフォームと DB SQL をリンクできます。これにより、アナリストは DB SQL と Databricks データインテリジェンスプラットフォームのパフォーマンスを犠牲にすることなく、ビジネスに最適なプラットフォームでダッシュボードを構築し、提供できます。

まとめ

ペースが速く、変動が激しい現代の金融の世界において、バッチ処理は、ストリーミングやリアルタイムサービスの機能や利点に対抗できる、最新のデータスタックの重要な一部であり続けています。このブログでは、金融サービス向けのレイクハウスアーキテクチャとそのパートナーのエコシステムを利用して、複雑なバッチ処理ワークロードをサポートするシンプルでスケーラブルかつ拡張可能なフレームワークを構築する手法を、保険金請求処理の実例を交えて見てきました。Delta Live Tables（DLT）と Databricks SQL（DB SQL）を使用することで、無限にスケーリング可能なアーキテクチャを持ち、変化する要件に柔軟に対応し、時の試練に耐えることができるデータプラットフォームを構築できます。

インフラの設定や構成など、サンプルパイプラインについて詳しくは、GitHub リポジトリを参照するか、デモ動画をご覧ください。

セクション

05

お客さま導入事例 : Databricks 導入による実際の成果

セクション 5.1

InMobi : 顧客とブランドの効果的なつながりを促進

広告に対して消費者は、関連性があり、パーソナライズされたコンテンツを求めています。特に、モバイルデバイスに表示される広告は、時間が限られているため。短時間で消費者の注意に引き、エンゲージメントを促進することが重要です。InMobi は、リアルタイムの顧客データを利用して、ターゲットを絞ったモバイル広告やロックスクリーン体験を提供し、これを実現しています。しかし、データ処理の要件が 1 時間当たり 20 テラバイト以上に増加するにつれ、マルチクラウドデータウェアハウスの運用コストが急増し、また、そのプロプライエタリな性質がサイロを生み出してコラボレーションとデータ共有を妨げていました。

InMobi は、マルチクラウドのデータウェアハウス（DWH）から Databricks に移行することで、さまざまなワークロード（データウェアハウス、AI、分析）を統合し、オペレーションを効率化し、エンジニアがより価値の高いタスクに注力できるようにすることで、オペレーションの俊敏性と効率性の向上を実現しました。レイクハウスに移行して以来、同社はマルチクラウドのデータウェアハウスを使用していたときと比べて総所有コスト（TCO）の大幅な削減だけでなく、組織全体の生産性を向上させ、新製品の市場投入までの時間を短縮しています。

Databricks は、価格性能を最適化するという私たちの要求に真っ向から応えてくれました。レイクハウスのおかげで、さまざまなワークロードのパフォーマンスを犠牲にすることなくコストを削減し、現在と将来にわたってデータと AI の運用を最適化できるようになりました。

InMobi 共同創業者兼グループ CTO
Mohit Saxena 氏

32%
マルチクラウド DWH と比較して TCO を 32% 削減

15%
マルチクラウド DWH と比較して クエリを 15% 高速化

20%
サプライヤーレポーティングの効率が 20% 向上

複雑なレガシーインフラとマルチクラウド DWH の困難な管理

InMobi は、ターゲティング広告を提供し、ブランドが効果的かつコスト効率の高い方法で消費者にリーチし、エンゲージすることを支援します。関連性の高い広告を正確に配信するには、膨大な量のデータが必要です。InMobi では、複数のクラウドデータウェアハウスを追加してオンプレミスの Hadoop システムを拡張し、さまざまな問題に対処してきました。しかし、処理が必要なデータ量の急激な増加（1時間あたり 20TB のデータ）に対応するためにレガシーシステムの上に追加構築し続けた結果、多くの課題を抱えたマルチクラウドデータウェアハウスを生み出し、極めて複雑で、障害の発生や、拡張に伴う極端なコスト増加、データのサイロ化によるデータの共有とコラボレーションの制限などの問題を引き起こしていました。InMobi のチームは、このシステムのままでは技術革新のスピードが遅くなり、貴重なエンジニアリングのリソースが保守モードに拘束されてしまうことに気づきました。

「構築したデータインフラは機能しましたが、非常に複雑でオーバーヘッドが発生し、コア製品から焦点が外れてしまいました。当社の優秀なエンジニアがお客さまにより多くの価値を生み出すためには、シンプルで統合されたシステムが必要でした。」

InMobi プラットフォームエンジニア部門シニアディレクター
Madhan Sundaram 氏

InMobi が求めていたのは、複数の問題を解決できる単一のシステムでした。この目標を達成するために、分散されたシステムを単一のプラットフォームに統合し、エンジニアが ML や大規模言語モデル（LLM）の開発など、より価値の高いタスクに集中できるようにする必要がありました。そこでチームは、データウェアハウスと AI のワークロードを単一のプラットフォーム上で統合する Databricks データインテリジェンスプラットフォームに注目しました。

レイクハウスへの移行でデータ、アナリティクス、AI を統合

InMobi には有能なエンジニアが揃っていますが、生産性と運用の俊敏性という点で、貴重な教訓を得ました。Sundaram 氏は次のように述べています。「環境のメンテナンスに多くの時間を費やしており、ビジネスのサポートやコラボレーションに支障をきたしていることに気づきました。戦略的になり、データをもっと効率的に運用する方法を特定したいと考えました。」現在のマルチクラウドデータウェアハウスと自社構築の見通しを慎重に評価した結果、Databricks データインテリジェンスプラットフォームが、インフラの複雑さを軽減して開発者の生産性を向上させると同時に、可能な限り最高の価格性能を実現するという目標に最も合致していると判断しました。

選定後、チームは Databricks と提携し、移行計画プロセスを開始しました。既存システム上に構築された 10 年以上にわたるカスタマイズと 1 ペタバイトを超えるデータにより、InMobi は移行が複雑になることを知っていました。ETL 側だけでも、オープンソースの Apache Spark™ からの移行には 150 のパイプラインと 8 つのチームが関わっていました。また、サプライヤーやお客さまへの情報提供に支障が出ないように、約 300 のレポートダッシュボードを移行する必要がありました。このプロセスをナビゲートするために、Databricks は InMobi と緊密に連携し、各チームあたり 2 名の社内エンジニアをサポートに割り当てて支援しました。また、Databricks の実装パートナーである Celebal Technologies とシームレスな移行に必要な最適化について話し合いました。

その結果、Databricks は InMobi のマルチクラウドデータウェアハウスからレイクハウスへの移行を容易に支援することができました。Sundaram 氏は次のように述べています。「技術的な差別化を図るのに役立つ業界初の機能を活用できるという点で、Databricks は私たちに優位性を与えてくれます。移行の際のコンピュートリソースの価格性能の最適化おいて、Databricks の専門知識が発揮されました。Databricks は、最適化を推進するためにオーナーシップを持ち、透明性が高く、教育的な対応をしてくれました。」

パーソナライズされたモバイル広告をあらゆるお客さまに

効率的で統合されたレイクハウスアーキテクチャにより、InMobi は堅牢な顧客データを最大限に活用し、よりスマートでパーソナライズされたモバイル広告を提供できるようになりました。さまざまなチームが、アドホック分析に Databricks Notebook を活用し、レイクハウスのサーバーレスデータウェアハウスである Databricks SQL を使用してビジュアライゼーションを行う Power BI を活用し、次世代 AI プラットフォームの構築に MLflow を活用しています。また、異常検知に Delta Live Tables を使用することで、SLA を 50% 改善し、コストを 80% 削減するという大きな成功を収めています。Unity Catalog のおかげで、データのサイロ化とデータの発見可能性も問題ではなくなりました。Unity Catalog を使用することで、テーブルとカラムレベルでアクセスを管理できるようになり、データの完全なリネージが取得されるため、データがどこから来たのか、古くなっているかどうかを可視化できるようになりました。分析と AI のニーズを満たすように設計されたプラットフォームにより、InMobi チームは、分析と AI のニーズを満たすように設計されたプラットフォームを利用して、顧客にインサイトをより効率的に提供するために、大規模言語モデル（LLM）などの新しいテクノロジーに積極的に取り組んでいます。「エンドユーザーが質問して必要な情報を見つけやすくするために、LLM の導入を検討し始めました。レイクハウスアーキテクチャは、ジョブを自動的に実行するため、この取り組みを容易にします。これにより、当社のチームは、専門知識がなくても質問するだけで、コンテキストに沿った回答をすぐに得ることができるようになります。」（Sundaram 氏）

ビジネスへの影響という点では、移行後、あらゆる面で測定可能な多くの改善が見られました。インフラコストが以前より 34% 削減されただけでなく、以前のデータ環境と比較してクエリ速度が 15% 向上し、ジョブの失敗が 20% 減少しました。TCO は、マルチクラウドデータウェアハウスを使用していたときと比べて 32% 削減され、ETL パイプラインの実行コストも 24% 削減されました。さらに定性的には、システムの信頼性が全体的に向上し、お客さまからの評価も高まっています。

「私たちの実験速度は大幅に向上しました。Databricks はコラボレーションを簡素化し、レイクハウスが提供する統合アプローチによって、新しい機能や製品を提供する際の効率性、生産性、コンプライアンスを向上させることができました。」

InMobi 共同創業者兼グループ CTO Mohit Saxena 氏

InMobi は、Databricks データインテリジェンスプラットフォームの統合プラットフォームに移行することで、モバイル広告領域のイノベーションに注力し、お客さまと社内のエンドユーザーの両方に価値をもたらすリアルタイムのパーソナライゼーションを提供できるようになりました。

その他の導入事例を読む

Delta Lake で、データのクエリの改善とデータ量の増大に対応できるようになりました。昨年はトラフィックとデータが 80% 増加したため、迅速な拡張が不可欠です。

Akamai エンジニアリングマネージャー
Tomer Patel 氏

セクション 5 . 2

Akamai : Delta Lake で大規模なリアルタイム分析を実現

Akamai は、広範かつ高度に分散されたコンテンツデリバリーネットワーク（CDN）を運営しています。Akamai の CDN は、世界 135 か国以上、1,300 を超えるネットワークに設置された約 345,000 台のサーバーを使用し、メディア、商取引、金融、小売、その他さまざまな業界の大企業のインターネットトラフィックを中継しています。インターネットトラフィックの約 30% が Akamai のサーバーを経由しています。Akamai はまた、クラウドセキュリティソリューションも提供しています。

2018 年、Akamai は Web セキュリティ分析ツールの提供を開始しました。このツールは、Akamai のお客さまに幅広いストリーミングセキュリティイベントを評価し、それらのイベントの分析を実行するための単一の統合インターフェースを提供するものです。この Web 分析ツールは、Akamai の顧客がセキュリティイベントに関連した情報に基づいたアクションをリアルタイムで実行できるよう支援します。Akamai は、Web 分析ツールに Delta Lake と Databricks データインテリジェンスプラットフォームを活用することで、膨大な量のデータをストリーム配信し、お客さまに提供する厳格な SLA を満たすことができます。

1 分未満

これまで 15 分かかっていた
インジェスト時間が 1 分未満に短縮

85% 以上

85% のクエリの応答時間が
7 秒以下に高速化

膨大な量のデータの取り込みとストリーミング

Akamai の Web セキュリティ分析ツールは、セキュリティイベントに関連するデータを 1 秒あたり約 10GB インジェストしています。小売業の顧客が多数のセールスを行なう際や、ブラックフライデーやサイバーマンデーのような大規模なショッピングの日には、データ量が大幅に増加する可能性があります。Web セキュリティ分析ツールには、分析目的で数ペタバイトのデータを保存します。これらの分析は、Akamai のお客さまを保護し、お客さま自身でセキュリティイベントを探索および照会できるようにするために行なわれます。

Web セキュリティ分析ツールは当初、Hadoop 上で Apache Spark™ を実行するオンプレミスのアーキテクチャに依存していました。Akamai はお客さまに、攻撃が発生してからツールに表示されるまで 5～7 分という厳しいサービスレベル契約（SLA）を提供しています。Akamai はこの SLA を満たすため、インジェストとクエリの速度の向上を求めていました。Akamai のエンジニアリングマネージャーである Tomer Patel 氏は次のように述べています。「お客さまが攻撃の状況を把握できるよう、データは可能な限りリアルタイムである必要があります。クエリ可能なデータをお客さまに迅速に提供することが重要です。パフォーマンスと SLA を向上させるため、オンプレミスから移行し、レイテンシを数分ではなく数秒にしたいと考えていました。」

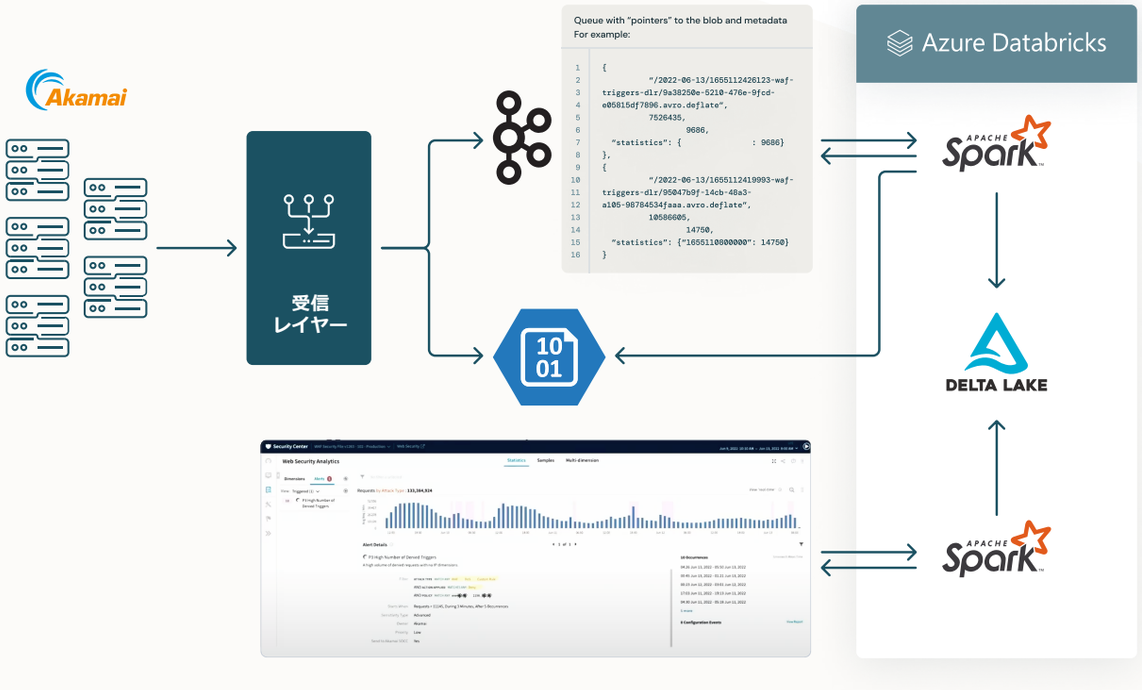
数社と概念実証を行った後、Akamai は Spark と Databricks データインテリジェンスプラットフォームをベースにしたストリーミング分析アーキテクチャを選択しました。「当社の規模と SLA の要求から、Databricks が最適なソリューションであると判断しました。ストレージの最適化とデータキャッシングを考慮すると、他のソリューションを採用した場合、同じレベルのパフォーマンスは得られません。」（Patel 氏）

スピードの向上とコスト削減

現在、Web セキュリティ分析ツールはデータを取り込んで変換し、クラウドストレージに保存し、Kafka 経由でファイルの場所を送信します。その後インジェストアプリケーションとして Databricks Job を使用します。Databricks データインテリジェンスプラットフォームの基盤となるオープンソースのストレージフォーマット、Delta Lake は、Web セキュリティ分析データのリアルタイムクエリをサポートします。Delta Lake はまた、Akamai の迅速な拡張を可能にしています。「Delta Lake を利用することで、データのクエリを改善できるだけでなく、データ量の増加にも対応できるようになりました。昨年はトラフィックとデータが 80% 増加したので、迅速に拡張できることが不可欠です。」（Patel 氏）

Akamai はまた、非常に高速なクエリパフォーマンスを提供する Databricks SQL（DBSQL）と Photon を使用しています。さらに Patel 氏は、Photon によってクエリパフォーマンスが大幅に向上したと付け加えています。全体として、Databricks のストリーミングアーキテクチャと DBSQL および Photon を組み合わせることで、Akamai はリアルタイム分析を実現し、リアルタイムのビジネス利益につなげています。

Patel 氏は、Delta Lake がオープンソースであることを好ましいと考えています。「Delta Lake がオープンソースであり、その背後に大きなコミュニティがあるということは、全てを自社で実装する必要がないということです。他のユーザーが遭遇したバグの修正や、プロジェクトに提供された最適化から利益を得ています。」Akamai は Databricks と緊密に協力し、Delta Lake が Akamai の定義したスケールとパフォーマンスの要件を満たすことができるよう努めました。これらの改善はプロジェクトに還元され（その多くは Delta Lake 2.0 の一部として利用可能になりました）、Delta Lake を実行する全てのユーザーが、実際の本番環境で大規模にテストされたテクノロジーから利益を得ることができるようになりました。



規模、信頼性、パフォーマンスに対する厳しい要件への対応

Databricks データインテリジェンスプラットフォーム上で Spark Structured Streaming を使用することで、Web セキュリティ分析ツールは膨大な量のデータをストリーム配信し、Akamai のお客さまに低レイテンシのリアルタイム分析サービスを提供できます。これにより Akamai は、攻撃発生から 5 ～ 7 分という SLA の範囲内で、セキュリティイベントデータをお客さまに提供できます。「私たちが重視しているのは、とにかくパフォーマンスです。プラットフォームのパフォーマンスとスケーラビリティが、私たちの原動力です。」（Petal 氏）

Databricks データインテリジェンスプラットフォームにより、セキュリティイベントデータの取り込みにかかる時間は 1 分未満になりました。Petal 氏は次のように述べています。「取り込み時間を 15 分から 1 分未満に短縮できたことは、非常に大きな改善です。お客さまは、セキュリティイベントデータをより早く見ることができ、何が起こっているのかを正確に把握し、全てをフィルタリングできます。」

Akamai の最優先事項は、お客さまに優れた体験と迅速なレスポンスを提供することです。現在までに、Akamai はセキュリティイベントデータの約 70% をオンプレミスのアーキテクチャから Databricks に移行しており、その結果お客さまのクエリおよび応答時間の SLA が大幅に改善されました。「Databricks への移行により、お客さまは応答時間を大幅に改善し、クエリの 85% 以上が 7 秒以内に完了するようになりました。」このようなリアルタイムのデータを提供することで、Akamai はお客さまが常に警戒し、最適なセキュリティ設定を維持できるよう支援できます。

その他の導入事例を読む

セクション 5.3

Quartile : 最大の eコマース広告プラットフォームへ

Quartile は、世界最大の eコマースクロスチャネル広告プラットフォームです。10 以上のチャネルから 5 千を超えるの広告アカウントにサービスを提供しています。Quartile のプラットフォームは、特許取得済みの 6 つの機械学習技術を基盤に構築され、Google、Facebook、Amazon、Instacart、Walmart などの eコマース広告を自動化・最適化します。Quartile は、最先端のテクノロジーとマーケティングの専門家を組み合わせ、お客さまのビジネス目標に合わせた戦略を立案します。Quartile のファネル最適化アプローチは、世界中の何千もの販売者から信頼を得ており、複数のチャネルにおける販売と広告の可能性を、完全に統合されたスケールで最大限に引き出しています。

Databricksデータインテリジェンスプラットフォームのおかげで、私たちの技術チームは新しいソリューションの市場投入までの時間を短縮し、質の高いデータセットでお客さまにご満足いただけるようになりました。Databricks とレイクハウスアーキテクチャがなければ、最大の eコマースクロスチャネル広告プラットフォームになることは、はるかに困難だったはずです。

Quartile CEO
Daniel Knijnik 氏

80%
従来のデータベースと比較して
データストレージが 80% 削減

10 倍高速化
7.5 時間かかっていた入札の最適化を
45 分に短縮

膨大な量のデータと拡張に必要なパフォーマンスの不足が課題

Quartile は、60 日以上のアトリビューションを含む販売データの統合とレポーティングにおける重要なニーズにより、複数の広告チャネルのデータの保存と処理に課題を抱えていました。以前のアーキテクチャでは、Quartile が処理するデータサイズに対応できませんでした。チームは 1 つのジョブで 10TB を超えるデータをバッチ処理しており、データレポーティングに必要な全ての変換を適用していたため、サーバーが利用できず、データポイントの配信が遅れていました。広告のパフォーマンス向上を担うこのプロセスだけでも、個々のジョブが毎日最大 7.5 時間稼働するなど、深刻なパフォーマンス問題を抱えていました。

テクノロジーの進化、レガシーからモダンデータスタックへ

企業の進化の一環として、Quartile のデータアーキテクチャは、Azure クラウド上で稼働する従来の SQL データベースから、Databricks データインテリジェンスプラットフォームを基盤とする新しいソリューションへと成長し、新たなレベルの成熟度に達しました。このモダナイゼーションは、社内のいくつかの領域に直接的な影響を及ぼし、Quartile のお客さまにも明らかなメリットをもたらしました。Delta Lake によってデータの信頼性が高まり、パフォーマンスが向上し、コストが削減されました。従来の SQL データベースから Databricks にデータを移行する際、主に Delta のバージョンコントロールと Parquet 圧縮による最適化によって、データ量が大幅に削減され、ストレージが 90TB から約 18TB に削減されました。

レイクハウスアーキテクチャは、Quartile のデータスタックの進化を可能にしました。新しいデータがクラウドストレージに到着した際に、Databricks Auto Loader を利用してそのデータを段階的かつ効率的に処理します。同時に、Delta Lake をオープンフォーマットのストレージレイヤーとして使用することで、データレイク上での信頼性、セキュリティ、パフォーマンスを向上させます。また、Databricks SQL を使って簡単にクエリやダッシュボードを作成できるようにし、これによりデータエンジニアとデータアナリストの協力を円滑に行えるようにしています。さらに、Databricks Workflows を導入することで、全ての機能を安定かつスケーラブルな方法で結びつけ、効果的なデータ処理を実現しています。

お客さまに最高の体験を提供するために、Quartile は正確なデータを取得できなくてはなりません。これは重要な課題ですが、Spark のユーザー定義関数を使用することで、並列処理のパワーを利用して必要な処理を必要な数だけ分割することができるため、容易に処理できます。ソリューションを拡張するために、Terraform を利用して全てのワークスペースをデプロイし、新しいクラスタとジョブを容易にスピンアップし、組織全体で正しい標準が守られていることを確認しています。

パートナー企業の広告運用を支援

Quartile のお客さまにとって、キャンペーンに関連する売上、コスト、その他の指標を分析できる一元化されたソリューションを持つことは極めて重要です。Quartile では、堅実なデータエンジニアリングを活用し、Databricks と Power BI を統合することで、ポータルに直接ダッシュボードを埋め込んでいます。これにより、クライアントが複数のチャネルにまたがるマーケティングキャンペーンを設定したり、パフォーマンスの変化をフォローアップしたりするための単一の場所を提供できます。また、オブジェクトストレージ上の Delta Lake のファイル形式を活用することで、従来のデータウェアハウスに保存されるデータよりも 80% 安価な、より小さなデータストレージフットプリントを維持できます。さまざまなチャネルのデータを組み合わせることができるため、既にいくつかのお客さまに役立っています。例えば、SmartyPants は Quartile と提携して以来、100% 以上の成長を遂げています。

しかし、それだけではありません。Quartile は、Databricks の機械学習ペルソナを利用して実装された、広告のパフォーマンスを向上させるアルゴリズムの特許も取得しています。データスタック全体を構築する中央のレイクハウスアーキテクチャを持つことで、Quartile の開発者の業務はよりシンプルになり、革新的なソリューションの開発に集中できるようになり、お客さまにより良い結果をもたらすことができるようになりました。別の例として、OfficeSupply は Quartile と提携して最初の 1 年間ですばらしい成果を上げました。個々のジョブのパフォーマンスが向上したことで、Google 広告の収益が 67% 増加し、商標用語による Google ショッピングのクリック数が 103% 増加しました。以前は 7.5 時間かかっていたジョブが、レイクハウスアーキテクチャでは 45 分で実行できるようになりました。

今後、Quartile は Databricks との提携を継続し、モダンデータスタックを構築し、新しいソリューションの統合とテストを行っていきます。これには、より高度なデータ品質チェックのための Delta Live Tables、お客さまに独自のデータを送信するための Delta Sharing、お客さまがすぐに始められるようにするための Data Marketplace などが含まれます。Quartile は、この分野で初の広告最適化アルゴリズムのためのクロスチャネル学習を開発するという大胆な目標を掲げており、Databricks はこれらのイノベーションの中心となる見込みです。

データブリックスについて

データブリックスは、米国サンフランシスコに本社を置き、世界中に拠点を持つデータとAIの企業です。Apache Spark、Delta Lake、MLflow のオリジナル開発メンバーによる創業以来、データの活用によって難題解決に挑む組織の支援に取り組んできました。Databricks データインテリジェンスプラットフォームは、コムキャスト（Comcast）、コンデナスト（Condé Nast）、Grammarly をはじめ、フォーチュン 500 企業の 50% を含むさまざまな業界の 10,000 社以上におけるデータの統合、民主化、分析と AI に活用されています。

X（旧 Twitter）、LinkedIn、Facebook での情報発信も行っております。

無料トライアルを開始

お問い合わせ先：
Databricks.com/jp/company/contact

databricks