ORACLE

テクノロジー概要：
Oracle GoldenGateプラットフォームを使用した動的なデータ・ファブリックと信頼性のあるデータ・メッシュ

信頼性があり、台帳に基づく、待機時間が短いストリーム・
エンタープライズ・データ・アーキテクチャの中核を成す原理と属性

2021年1月、バージョン2.1

免責事項

本書は情報提供のみを目的としており、記載した製品機能の実装およびアップグレードの計画を支援することのみを意図しています。マテリアルやコード、機能の提供をコミットメント（確約）するものではなく、購買を決定する際の判断材料になさらないでください。本書に記載されている機能の開発、リリース、および時期については、弊社の裁量により決定されます。製品アーキテクチャの性質上、コードが大幅に不安定化するリスクなしに、本書に記載されているすべての機能を安全に含めることができない場合があります。

本書の目的

本書の対象読者は、データ・ファブリック、データ・メッシュ、Oracle GoldenGateのストリーム・データ・プラットフォームに関心のあるテクノロジー担当エグゼクティブとエンタープライズ・データ・アーキテクトです。本書の構成および内容は、一般的なエンタープライズ・データ管理ツールおよびパターンに精通していることを想定していますが、Oracle GoldenGate、データ・ファブリック、データ・メッシュの概念に馴染みのない個人の方にも適しています。

本書のおもな目的は、（1）新たなデータ管理機能、（2）信頼性のあるリアルタイム・データ・メッシュの重要な属性の詳細、（3）Oracle GoldenGateを使用してそのような機能をいかに提供できるかを示す簡潔な例について説明することです。

本書は、Oracle Cloud、およびマルチクラウドのエンタープライズ環境に適用されます。

目次

エグゼクティブ・サマリー

ビジネス変革の取組みは、データに関する時代遅れの考え方と旧世代のモノリシックなデータ・ツールによって足止めされています。現代のエンタープライズ・データ資産は分散化とマルチクラウド化が進み、時間とともに無秩序なデータが否応なしに増加し続けています。ビジネス変革の目標を達成するには、適切に管理され、自由にストリームできるデータがビジネス・チームに必要です。

モノリシックなデータ・アーキテクチャは、クラウド内であろうと、オンプレミスであろうと、必要なモジュール性とビジネスの成功に求められる速度を実現できません。動的なデータ・ファブリックと信頼性のあるデータ・メッシュは、新たな思考と最新のテクノロジーを結合することで、エンタープライズ・データ資産の価値をさらに解放する新しい道筋を示します。この新しいアプローチを採用すると、データ・サプライ・チェーンでスマートな自動化が実現し、複雑性が緩和されるため、より素早いイノベーション・サイクルとデータ運用コストの削減が可能になります。

エンタープライズ規模のすべてのビジネスにはデータ資産があります。数百、あるいは数千ものアプリケーション、データ・ストア、データ・レイク、および分析機能によって、ビジネスが運営され、多くの事業部門にまたがる意思決定が促進されている場合があります。既存の市場を破壊し、新たなビジネス機会を創出することによって、データ資産からより多くの価値を引き出すことに成功した企業が市場の成功者となります。成功を収めるデジタル変革の取組みは、強固な戦略と優れた実行力（すなわち優れたスタッフとプロセス）に基づいていますが、新たな効率性を作り出し、新たなイノベーションを推進し、新たなビジネス機会を切り開く上で、テクノロジーが常に重要な役割を果たしてきました。

動的なデータ・ファブリックと信頼性のあるデータ・メッシュは、企業がデータ資産を構築し、管理するための全く異なる素晴らしいアプローチを提供します。この新しいアプローチは、データ・プロダクト思考を優先し、データの分散化を十分に取り入れた新しいタイプのエンタープライズ・データ・アーキテクチャであり、リアルタイムのストリーム・データ・プラットフォーム内に信頼性のある適切なデータを保存するように構築されています。中核を成すのは、動的なストリーム・データと価値ベースのデータ・プロダクト管理に重点を置く、業界が信頼を寄せるデータ・ファブリックの基盤です。データ・メッシュの概念もこのアプローチの中心的な側面ですが、データの整合性、正確性、信頼性を確保できる技術的基盤も備えています。

この新しいタイプのデータ・アーキテクチャにより、より素早いイノベーション・サイクルと運用コストの削減が可能になります。このアプローチの早期採用者と先駆者により、現在次のような意義深い多くの利点を享受できる可能性があることが証明されています。

- **データのバリュー・チェーンの完全な透明性** – ‘データ・プロダクト思考’のベスト・プラクティスを適用
- **業務系データの99.999 %以上の可用性** – レプリケーションにマイクロサービス・ベースのデータ・パイプラインを使用
- **10倍高速なイノベーション・サイクル** – ETLから連続変換およびロード（CTL）に移行
- **データ・エンジニアリングの最大70 %の削減** – ノーコードのセルフサービス式データ・パイプライン・ツールを使用

常時運用に関連するコストを削減すると同時に、データを競争上の優位性として活用する各事業部門でイノベーションの強化を図ることもできるソリューションを見つけることは非常に難しく、ここではまさにそのような機会を提供しています。

企業のニーズを満たす動的なデータ・ファブリック

2000年代前半に最初に普及したデータ・ファブリックは、当初はインメモリ・オブジェクト・グリッドと深く関連していました。その後、Forresterがより汎用的なデータ・ファブリック・ソリューションについて記述し始め、2013年までにはデータ・ファブリックは本格的な調査カテゴリとなりました[i]。データ・ファブリックの概念は浸透し、Gartnerは“*データ・ファブリックはデータ管理の未来である*”とさえ宣言しました[ii]。現在、データ・ファブリックのトピックは広範なデータ・テクノロジーに適用されています。

一般的に、データ・ファブリックをすべて網羅する単一ツールが存在しないことは誰もが理解しています。むしろ、データ・ファブリックを採用する組織にとって、データ・ファブリックとは、多くのデータ統合とガバナンスの‘方式’にまたがり、統一された1つのソリューションを実現する設計の概念です。Forresterはデータ・ファブリックを次のように定義しています。“*新たに出現したユースケースに対応する統一されたインテリジェントなエンド・ツー・エンドの統合型プラットフォームを実現します。その中核を成すのは、動的な統合、分散型マルチクラウド・アーキテクチャ、グラフ・エンジン、および分散型インメモリおよび永続メモリ・プラットフォームでイノベーションを活用することによって迅速にユースケースを実現する機能です。データ・ファブリックは、プロセス統合、変革、準備、キュレーション、セキュリティ、ガバナンス、およびオーケストレーションの自動化に重点を置くことで、ビジネスを成功に導くための分析とインサイトを素早く実現します*[iii]。”

オラクルは、データ・ファブリックと次の完全なポートフォリオにおいて、唯一の[iv]リーダーとして認識されています。

クラウドネイティブな一般的なプラットフォーム・データ・ファブリック	マルチクラウドとオンプレミス向けの最善のデータ・ファブリック
• 分析と自律型DB向けのセルフサービスETL • OCI Data Catalog、OCI Data Integration、OCI Data Flow • OCI GoldenGateとOCI向けOracle Stream Analytics • Oracle Integration CloudとOracle Cloud SQL	• Oracle Data Integrator（ETL、品質、メッセージングを提供） • Oracle GoldenGateとStream Analytics • Oracle Big Data SQL（データ連携） • Oracle Data Visualization（データ準備）

オラクルのポートフォリオに含まれるOracle GoldenGateプラットフォームは、分散化されたマルチクラウド・エコシステム向けのリアルタイム・レプリケーション、ストリーム、時系列分析、インメモリ・データ処理を重視する動的なデータ・ファブリックの概念に明確に重点を置いています。

信頼性のあるデータ・メッシュに対するオラクルの概念

‘データ・メッシュ’は、1990年代に3Dラミネートとデジタル顕微鏡検査を表す用語として使用され[v]、2000年代前半までは、TCP/IPネットワークの仕組みを説明する用語として使用されていました[vi]。2000年代後半のデータ管理のコンテキストでは、データ・メッシュは、Web Ontology Language（OWL）やResource Description Framework（RDF）といったセマンティックWebの早期の取組みを説明する文書[vii]でまず普及し始めました。
その頃の2007年、Wikipediaでも“データ・メッシュ”の非公式な定義を“*断続的にのみアクティブな異機種ネットワーク間の（中略）任意のポイントから任意の別のポイントにデータを転送するためのネットワーク*”と記載していました[viii]。

最近では、エンタープライズ・データにおけるデータ・メッシュの概念が、2016年のGartnerのレポート、『*Maverick* Research:Revolutionizing Data Management and Integration With Data Mesh Networks*』に記されています[ix]。その後、2019年にThoughtWorksの記事、『*How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh*』[x]でその概念が復活し、より広く理解されるようになりました。

お分かりのように、データ・メッシュという用語の起源はさまざまですが、本書の目的においては、直近の参考文書であるThoughtWorksの記事を採用します。その記事では、“*データ・メッシュ・プラットフォームは、相互運用性のための一元化されたガバナンスと標準化の下で、意図的に設計された分散型データ・アーキテクチャであり、統一された共有のセルフサービス・データ・インフラストラクチャによって実現される*”と説明されています。データ・メッシュの主眼は、最重要事項として‘ドメイン・データ・プロダクト’に重点を置き、‘不変なデータセット’を製品として使用してデータの分散化を促進することです。

データ・メッシュのこの新しい概念化における伝統とDNAは、ドメイン駆動設計（DDD）やイベント・ソーシングといったマイクロサービス設計思考の概念に由来しています。ThoughtWorksのリーダーたち（Martin Fowler氏他）は、長きにわたり分散型ソフトウェア、アジャイル開発、マイクロサービス・メッシュ設計パターンにおけるリーダーでした。この思考の伝統が、データ・メッシュの枠組みを形作る一部の素晴らしい側面（分散化、モノリスの分解、データ・プロダクト思考など）を伝えてきましたが、いくつかの重要な点（最終的な整合性、開発者の経験則への依存、ペタバイト規模のデータ管理の物理的性質）が見落とされる原因にもなっています。

本書のオラクルの定義は、これまでのデータ・メッシュの定義に基づきつつ、厳格なデータ整合性（信頼性のあるデータ・トランザクション）、データのガバナンスと検証可能性（データの妥当性検査）、およびエンタープライズ規模の要求（ミッション・クリティカルなデータのペタバイト規模のデータ移動）の各要素をさらに重視しています。この集約された定義では、**信頼性のあるデータ・メッシュとは、成果（データ・プロダクト）、マルチクラウド環境でのITの俊敏性（メッシュ）、信頼性のあるあらゆる種類のデータ（ポリグロット・データ・ストリーム）、および迅速なビジネス・イノベーション・サイクル（イベント駆動型のデータ台帳の使用）に重点を置いたデータ・アーキテクチャのアプローチです。**

データ・メッシュを説明する言葉として選択され得るすべての言葉のうち、‘メッシュ’という言葉は馴染みがあり、情緒的で、独自に要件を満たしています。馴染みのある他のテクノロジーの‘メッシュ’（メッシュWiFi、スマート・ホーム・メッシュ、5Gメッシュ、サービス・メッシュ/Kubernetesなど）と同じく、データ・メッシュの反復的な中心属性は、ネットワーク化され、分散化されたアーキテクチャの原型です。メッシュは根本的に、集中型のモノリシックなアーキテクチャを否定しています。

データ・メッシュには、本書で説明する以外にも多くの重要な属性があります。しかしながら、過去との技術的な違いを理解するとともに、この新しいアプローチを採用することで、すべてのアプリケーション、サービス、分析機能が本質的に分散化され、マルチクラウド化される今後に適切に備えることができる理由を理解するための主軸となるのは、この“メッシュ性”です。

ファブリック/メッシュとの橋渡し役となるGoldenGate

1990年代に創業したサンフランシスコの新興企業であるGoldenGateでは、Tandem NonStopデータベースからネットワーク接続された、稼働中のATM（現金自動預け払い機）にビジネス継続性とデータの高可用性を提供することが当初の目的でした。

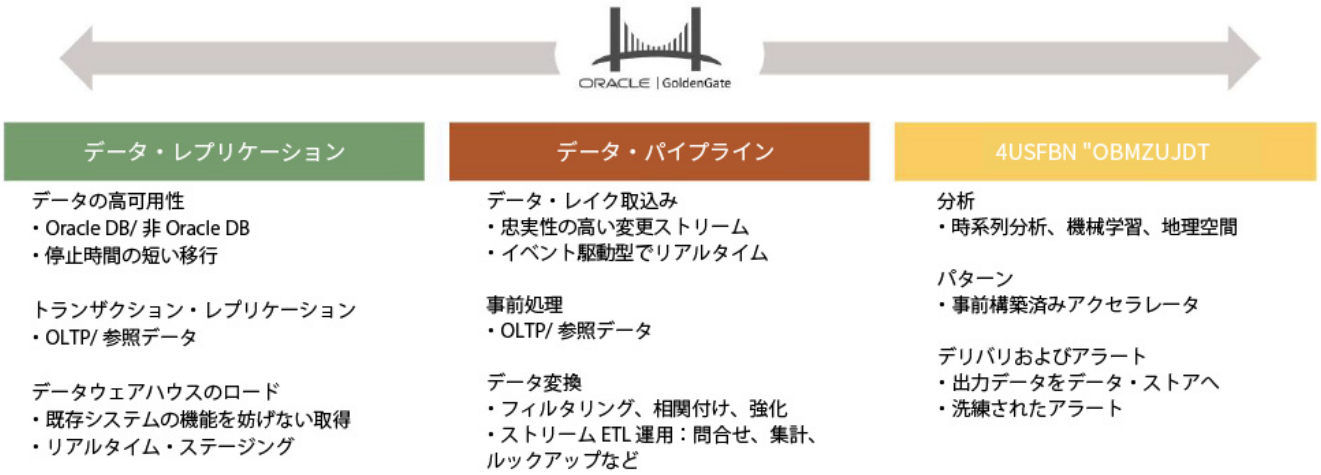


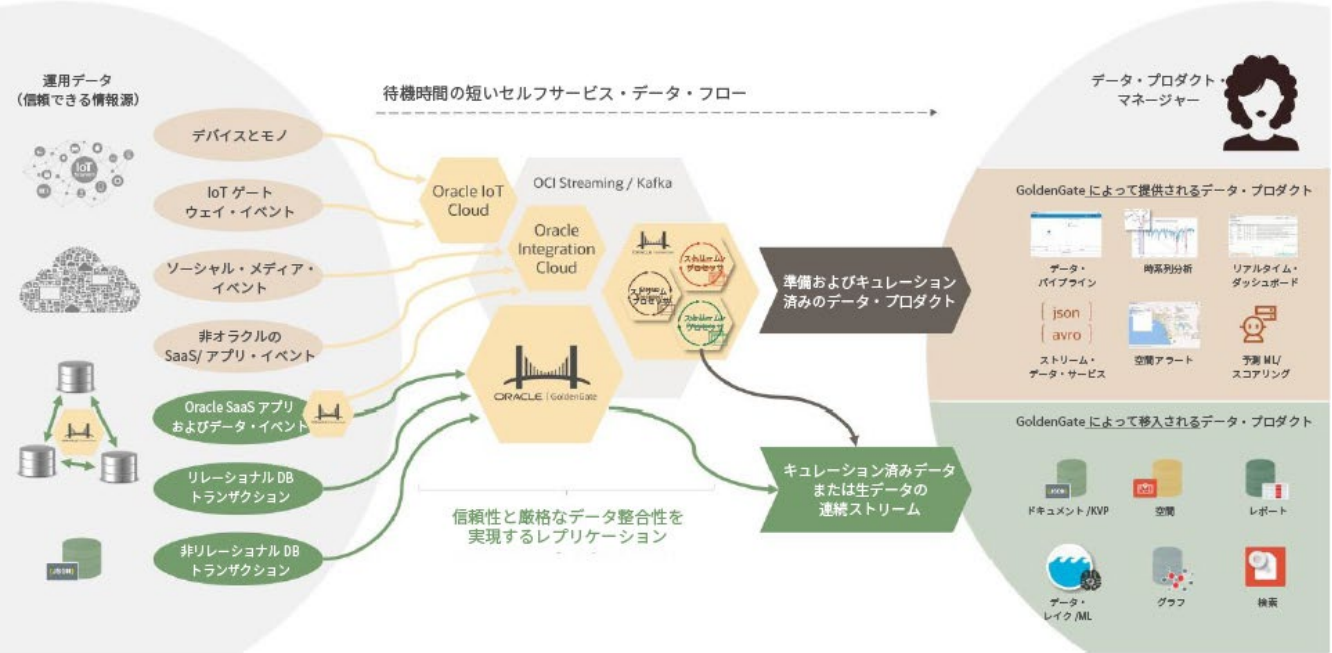
図1：GoldenGateプラットフォームの機能

Oracle GoldenGateソフトウェアは、現在もなおNonStop、DB2 iSeries、LUWなどのデータベースや、メインフレーム、SQL Serverにビジネス継続性とデータの高可用性を提供しており、Oracle Database Maximum Availability Architectureの‘プラチナ層’サービス・レベルの頂点でもあります。何千ものグローバルな銀行、小売業者、通信事業者、医療会社などが、Oracle GoldenGateという信頼性のある基盤上で業務系データ・プラットフォームを実行しています。

GoldenGateは基本的に、データ・イベントを検知し、非常に短い待機時間でネットワーク全域に転送できるリアルタイム・データ・レプリケーション・ツールです。GoldenGateテクノロジーは、業務系データベース、停止時間の少ないデータ移行、オンラインのマルチアクティブ・データ・ストア、クラウドへのリアルタイムのデータ取込み、データ・レイク、データウェアハウスなどの地理的なシャーディングに使用されます。2015年以来、GoldenGateはポリグロット・ビッグ・データとnoSQLデータ・ペイロードにますます重点を置くようになり、ネイティブなマイクロサービスの‘サービスとして’のデプロイメント向けに完全にリファクタリングされています。

2018年にGoldenGateプラットフォームは、順序付けされたデータ処理をナノ秒スケールに保ちながら、1秒あたり数十億イベントにまでスケーリングする堅牢な複合イベント処理（CEP）コア・エンジンを備えたStream Analyticsとデータ・パイプラインを追加しました。このイベント・エンジンは、非常に強力なセマンティックを変換や分析で使用し、大規模パラレル・プロセス（MPP）のためにオープンソースのApache Spark上で実行できます。

GoldenGateはこれらの新機能を使用して、高価値のデータ・プロダクトをデータ・コンシューマに直接提供します。これまでは、GoldenGateは主に待機時間の短い生データをデータ・パイプラインやその他のデータ・プロダクトに提供するために使用されていました。現在は、生データをプッシュできるほか、高価値のデータ・プロダクトを提供できます。



GoldenGateによって提供されるデータ・プロダクト	GoldenGateによって移入されるデータ・プロダクト
• データ・パイプラインとストリーム・データ・サービス • 時系列分析と本番環境のML/AIスコアリング • 地理空間アラートとリアルタイム・ダッシュボード	• データ・マートまたはOLAP/ROLAP分析 • データ・レイクとクラウド分析 • ドキュメント、KVP、検索、およびグラフ・データ・ストア

図2：GoldenGateによるデータ・プロダクトの作成と移入

ビジネス・リーダーには、今後より多くのストリーム・データとリアルタイム・イベント、より高いデータ可用性、より強固なトランザクション・ガバナンスが必要となります。信頼性のあるデータ・ファブリックは、信頼できるデータソースから、下流のデータ・レイクやデータを消費する分析ツールに至るまでのあらゆる場所で、厳格なデータ整合性を確保する必要があります。GoldenGateは立証済みの信頼性のあるデータ・ファブリック・プラットフォームであり、データ・メッシュの未来の要となります。

次のステップとして、本書の残りの部分で以下について詳しく説明します。

- **変化のとき** – 変わる必要のある時代遅れの思考と文化的な変化を調査し、この新しいデータ・アーキテクチャのアプローチへの足掛かりとなる次世代の技術イノベーションを評価します
- **新しいパラダイムの原理** – データ・プロダクト思考、分散化されたメッシュ・ネットワーク、エンタープライズ・データ台帳、ポリグロット・データ・ストリームという4つの原理を簡潔にまとめます
- **信頼性のあるデータ・メッシュとしての動的なデータ・ファブリック** – 動的なデータ・ファブリックを実現する属性と特徴、およびデータ・メッシュによって、信頼性と100 %の整合性を提供するデータ・ストリームをいかに実現できるかについて詳しく見ていきます
- **Oracle GoldenGateの説明** – 既存のGoldenGateテクノロジーの製品特性を動的なデータ・ファブリックと信頼性のあるデータ・メッシュのパターンに対応付けます

変化のとき

35年間主流のアーキテクチャとして普及してきた集中型のモノリシックなデータ・アーキテクチャは、変化の機が熟しています。モノリシックなデータ・アーキテクチャは、数十年間重要であり続けたあらゆるソフトウェア・パターンと同じく、成功を収めたと確信を持って言うことができます。実際、おそらくすべての企業のIT組織がすでに多くのデータ・モノリスを本番環境で実行するほど成功を収めています。このようなデータ・モノリスには、オンプレミスまたはパブリック・クラウドのバッチETLツール、業務系データ・ストア（ODS）、エンタープライズ・データウェアハウス（EDW）、データ・レイクが含まれます。

> *35年以上の間、データ・モノリスは、エンタープライズ・データ・アーキテクチャを検討する際の主流のパラダイムであり、そのパラダイムは今まさに変わろうとしています。*

より広範なソフトウェア開発エコシステムでは、変化の風はすでに吹いています。過去のモノリシックな従来型ソフトウェアを超えたいという願望が、マイクロサービスやKubernetesなどのサービス・メッシュ・アーキテクチャの幅広い採用に直接表れています。俊敏性、モジュール性、変化への耐性を高め、イノベーション・サイクルを短縮したいと考えるビジネスや開発者の熱意によって、エンタープライズ・ソフトウェアが記述される方法は様変わりしました。

データの世界では、モノリシックなデータ・アーキテクチャには同じ（モノリシックなアプリケーションの）基本特性が数多くあり、そのことが現在、データのモダナイゼーションとビジネス変革に欠かせない重要な改善の妨げとなっています。従来のモノリシックなデータ管理の属性として以下が挙げられます。

ITアーティファクトとしてデータを扱う	*データはアプリケーション機能の副産物として扱われるため、ドメイン・モデリングのセマンティックをITライフサイクルで繰り返し行う必要があります*
モノリシックで集中型	*ハブアンドスポーク方式のアーキテクチャはITエコシステムを支配しますが、各アプリ、ETL、マート、ウェアハウス、レイクなどは、各々をデータ環境の中心と捉えているため、ITは、互換性のないツールを使用してデータの統合と連携を図るプロジェクトに絶えず資金を提供する必要があります*
ウォーターフォール型DataOps/DevOps	*データベース、ETL、マート、ウェアハウス、レイクなどの“データ環境”では、ウォーターフォール型の運用に対する強い先入観が依然として存在します。開発のアジャイル手法は、反復可能なCI/CDライフサイクルのアプローチをモノリスで提供できませんでした*
バッチ処理中心	*ほとんどの分析（OLAP）ドメインでは、データの移動の大部分は依然として、ビジネス自体のイベントではなく、スケジューラ（時間）によって引き起こされるバッチ指向です*
（分離された）OLTPとOLAP	*さまざまな運用アプリケーション（OLTP）と分析アプリケーション（OLAP）は通常、組織的、政治的、技術的に分離されているため、ITの摩擦（遅延）、データ・ドメイン・セマンティック問題（データの信頼性低下）、最小公分母のソリューション（少ないイノベーション）がもたらされます*

従来型のツールとアーキテクチャ設計を変更することは必要かつ重要ですが、おそらくそれよりも重要なのは、データの本質的価値についての思考を広く範例的に変えることも必要であるという点です。一部のビジネス・リーダーは、次第にデータを資産と見なし始めています。実際、データは資本の一種です。これは、“データは新たな石油である”、“データは新たな金である”といった比喩ではありません。データは、経済の教科書に記載される資本の文字通りの定義を満たしています。資本は、天然資源ではなく、生成された財産です。単に地面から掘り出すのではなく、投資して作り出す必要があります。何よりも、データ資本は他の財産やサービスを産出する上で不可欠な要素です。この新しい思考のパラダイムでは、データとは価値の有形資産であり、その価値は、データを保持する多くの分散化されたデジタル・インフラストラクチャ全体で、まとまりのある方法で管理される必要があります。

前進するためには、パラダイム・シフトが必要であるという結論に私たちは到達しました。このパラダイム・シフトは、私たちの考え方（思考へのアプローチ）だけでなく、ソリューション領域で使用するツールやテクノロジーにも影響を及ぼすに違いありません。

モデルの危機：データ・モノリス

有能なエンタープライズ・アーキテクトが皆知っているように、ソフトウェアの問題を解決する特効薬や魔法の解決策は存在しません。テクノロジーにはすべて、トレードオフ、コスト、結果が伴います。モノリスとメッシュについても同様です。本書はデータ・メッシュ（およびデータ・ファブリック）という新たに出現した領域を説明することに重点を置いているため、つまるところ、データ・モノリスが抱える多くの問題について説明することが必要になります。

パラダイム・シフトは、主要なパラダイムが新たな現象と相容れなくなったときに発生し、これにより新たな理論、すなわちパラダイムの採用が促進されます[xi]。35年以上の間、データ・モノリスは、エンタープライズ・データ・アーキテクチャを検討する際の主流のパラダイムであり、そのパラダイムは今まさに変わろうとしています。パラダイム理論の言葉では、‘モデルの危機’は、確立された主流のパラダイムが、組織に立ちはだかる新たな現実に適切に対処できなくなったときに起きるサイクルの3番目のステップです。

2015年以降、パブリック・クラウド・テクノロジーが急速に採用されたことで、無秩序なデータの増加が引き起こされました

現代のITでは、企業の‘データ資産’（企業のデータ資本を保持するデータ・ストアとデータ中心のアプリケーションのセット）がかつてないほど分散化されているという新しい現実が出現しています。2015年以降、パブリック・クラウド・テクノロジーが急速に採用されたことで、無秩序なデータの増加が引き起こされ、データ資本がさまざまな物理ロケーション、データ・ストア、ネットワークに分散されるようになりました。しかしながら、古いデータ・モノリスはこの新しい現実にとって理想的ではありません。

典型的なモノリシックなデータ管理は、（a）互いに数百または数千キロメートル離れたインフラストラクチャにまたがって、データがまとまりのある方法で管理され、（b）さまざまなデータ形式（ポリグロット）が使用され、非常に短い待機時間で接続され続ける必要があるマルチクラウド・エコシステムには適していません。これらの課題は、元々分散化されているデータを管理および制御する作業に影響を及ぼします。以下のようなデータ・モノリスは、このような分散化が進んだ新しいデータの課題にはあまり適していません。

- **従来型のETLとデータ準備ツール** – これらは、通常は集中型の処理エンジンで処理する必要があります。‘プッシュ・ダウン’や‘E-LT’処理を行うという主張にだまされないでください。そのような機能を主張するツール・ベンダーのほとんどは、いまだにステージングやワークロードのために、データを単一のインフラストラクチャ・ハブに集める必要があり、これはモノリスの従来の定義にほかなりません。‘E-LT’についての主張は、多くの場合、組込みの
  専有エンジンではなく、データ・ストア・ベースの処理エンジン（データベース、Hadoop、Spark、サーバーレス・クラウドのオプションなど）をツールが使用していることを表しています。Oracle Data Integratorなどの少数のバッチ処理ツールが、‘エージェント’として実行され、開発者がワークロードをさまざまなエンジンやロケーションに移動できるようにします。
- **従来型のデータ・レプリケーション・ツール** – このツールは長年の間、業務系データ・アーキテクチャの一部でした。1990年代にETLツールに代わるイベント駆動型ツールとして、チェンジ・データ・キャプチャ（CDC）ツールが使用され始めました。CDCツールの従来型アーキテクチャはデータ・モノリスのため、メタデータとシステム・フレームワークのために‘ハブ型’のデプロイメントと緊密に統合されたデータベースが必要です。このような設計は、本書で説明する最新のメッシュ型デプロイメントには適していません。
- **業務系データ・ストア（ODS）** – 一般的に従来型のKimballアーキテクチャやData Vaultアーキテクチャの一部として使用されるODSは、アプリケーション・データを収集し、保管するために使用されます。このアプリケーション・データは、通常は下流のデータ・マートまたはデータウェアハウスによって使用されます。ODSの通常のデータ
  入出力フローは、従来型のETLツールを使用してバッチ処理で行われるのが一般的です。
- **エンタープライズ・データウェアハウス（EDW）** – いろいろな意味で、EDWはデータ・モノリスの典型的な定義です。EDWの概念化（Kimball、Inmon、Data Vaultなど）は数多く存在しますが、共通の‘サブジェクト領域’とデータ・ドメイン一式を持つ、企業のレポート作成における信頼できる唯一の情報源であるという考えが、数十年の間主流でした。実際は、ほとんどの組織が結局のところ、特定の事業部門に合わせたデータウェアハウスとデータ・
  マートを数多く持つのが通例です。
- **データ・レイク（リザーバ、ポンド、スワンプ、レイク・ハウスなど）** – ‘モノリス’といえば、データ分析のメインフレームともいえるビッグ・データです。Hadoopエコシステムの初期の形態では、ストレージの緊密な統合が計算処理に事実上必要であり、200近いさまざまなApacheフレームワークとJavaフレームワークを実行することが求められました。それよりも新しいデータ・レイクでは、依然としてテクノロジーを特定のクラウド・ベンダーやビッグ・データ・テクノロジー・スタックに一元化することが求められます。現代企業はおそらく、EDWの場合と同様に、1つの巨大な集中型データ・レイク・テクノロジーに頼るのではなく、複数の異なるデータ・レイク・
  テクノロジーを組み合わせて実行しています。

多くのデータ・ストアはいずれも、ある程度はモノリシックであるため、全体的なエンタープライズ・データ資産には、必要に迫られて非常に多くのデータ・ストア（一部は古く、一部は新しいデータ・ストア）が常にあることになります。データの準備、パイプライン、統合の各ツールは、DevOps、DataOps、およびCI/CD（継続的インテグレーション/継続的デリバリ）のためにモノリスである必要はなく、モノリスであるべきではありません。データ・モノリスは、一夜でなくなることはなく、時には、モノリスのアーキテクチャが推奨される状況に陥る場合もあります。組織はマイクロサービスとサービス・メッシュを採用してアプリケーションのモノリスから移行していますが、データ・モノリスは依然として進歩の障壁であり、データ駆動型の大規模なビジネス変革で避けることができない事項を解放するために克服しなければならないものです。

古い思考法と前提の廃止

“この方法でいつもやってきたから”という言葉が、変化を避けるための言い訳としてあまりにも頻繁に使われてきました。新しいパラダイムを使用して前進するには、確立されている思考方法を一部変える必要があります。このように考え方を変えることは、複雑なビジネス組織の文化では通常は体現することが極めて困難です。動的なデータ・ファブリックまたは信頼性のあるデータ・メッシュを成功させるためには、データ管理に関する以下の重要な前提を改める必要があります。

古い前提1：データを副産物/ITアーティファクトとして扱う

アプリケーション開発における過去数十年間の一般認識では、データは、直接管理および制御される資産としてではなく、ソフトウェアの副産物として扱われていました。ことの始まりはおそらく、物理的なデータ（ドキュメントやデータベース表内のデータ）を物理的に最適化して、古いテクノロジーで効率的かつ永続的に処理する必要性が頻繁に発生したことによる現実的な決断でした。物理的なデータの物理的な形態は、多くの場合、ソフトウェア・アプリケーションがインメモリに保持するデータの論理的な形態とは異なりました。このような‘インピーダンス・ミスマッチ’により、アプリケーション開発者が使用するデータ（オブジェクト）とデータ・エンジニアが使用するデータ（物理的構造）の間に論理的な隔たりが生成されます。

ただし現在は、‘インピーダンス・ミスマッチ’問題はさらに複雑化しています。2020年代のデータ構造とデータ形式の分野は、2000年代前半と比較してはるかに複雑です。‘ポリグロット永続化’と‘コンバージド・データベース’の戦略が業界で広く採用されたということは、通常はビジネス・データが、リレーショナル、Key/Valueペア、ドキュメント、グラフ、時系列構造、オブジェクト構造といった多様な形式にフォーマットされるようになったことを意味しています。データのインメモリ表現と物理的な永続的形式はどちらも、これまで以上にポリグロットです。

形式に依存しない方法でデータを管理することが緊急に必要であることを認識するようになったのは偶然ではありません。データをある種の‘アプリケーションの排出物’として管理するのではなく、ドメインとしてデータの本質を概念化する必要性が発生しています。そうすることで、物理的形式やインメモリ形式にかかわらず、データが表現する論理的な考え方をより効果的に管理できるようになります。そのため私たちは、データをITの副産物として管理するという思考を改め、データをビジネス・ドメインの付属品として、およびライフサイクル内で管理され、保守される一種の製品として捉え始めています。

古い前提2：OLTP→ETL→OLAPの断片化

数十年もの間、業界はバッチ指向のデータ・アーキテクチャに依存してオンライン・トランザクション処理（OLTP）データ・ストアをオンライン分析処理（OLAP）データ・ストアに連結していました。スケジューラ駆動型のETLツールは、ファイルを移動し、日次/週次バッチ・ジョブを実行することで、データを移動します。OLTPにおけるデータ・オブジェクト、データ・モデル、および統合の基本的な意味は、分析におけるデータ・オブジェクト、データ・モデル、機械学習機能、データ統合の意味とかけ離れています。分析データをアプリケーションの信頼できる情報源に再び関連付ける方法として私たちが知っているのは、プログラム、マッピング、ストアド・プロシージャ、スクリプト、データ・パイプライン（および時には関連付けが解除されたデータ・カタログ）が大まかに関連付けられた複雑なグラフを使用するやり方のみです。これは、分散アーキテクチャでデータを管理する方法としては根本的に破綻しています。

思考を改めるには、OLTPの信頼できる情報源と連携したデータ・ファブリックと同じものを使用して、分析フレームワーク内でデータの連携と管理を行うことができれば、どれほど簡単であるかを考える必要があります。インピーダンス・ミスマッチを最小限に抑えながら、トランザクションの統一性と整合性を保てば、データの移動に必要なデータ・パイプラインの著しい複雑性が大幅に低減されるでしょう。動的なデータ・ファブリックと信頼性のあるデータ・メッシュにより、OLTPドメインと分析ドメインが同じ基盤で大規模に相互運用される青写真が提供されるはずです。

古い前提3：データ・プロジェクトのウォーターフォール・プロセス

OLTPプロジェクトと分析プロジェクトは、従来、異なる職務を担う事業部門やIT組織の間で分離されているため、通常はプロジェクトを推進するスタッフやプロセス手法を統一できません。そのため、大規模なデータ管理の取組みにアジャイル開発手法を取り入れることは極めて困難です。つまり、アジャイル手法によって提供される従来の最適化は適用されないか、組織間で共有されません。これらの組織間の問題に輪をかけているのは、何層にもおよぶDevOpsおよびCI/CD（継続的インテグレーション/継続的デリバリ）の複雑性を抱えるモノリシックなデータ・ツールが、極めて大規模なデータ管理で依然として使用されているという事実です。

アジャイル開発手法が、（マイクロサービスとサービス・メッシュに向けた）モノリシックなソフトウェア・アーキテクチャについての思考の大変革を最終的に後押ししたのと全く同じ理由で、データ管理でアジャイル手法を推進したいという熱意が、データ管理アーキテクチャの再考（データ・ファブリックとデータ・メッシュなど）を後押ししています。データ管理における俊敏性を最大化するには、データ・アーキテクチャをより小規模で、動的で、疎結合されたモジュール式のコンポーネントに分解することが必要です。

古い前提4：バッチ優先の設計思考

ITデータ管理は当初から、データ移動およびデータ処理のワークロードのためのバッチ処理を中心に展開されてきました。もちろん、何十年も前からリアルタイム・システムはありますが、バッチでデータを移動し、処理するという中心的なパラダイムに‘追加’される機能として存在してきました。バッチ処理が今すぐなくなるわけではありませんが、データ・アーキテクトとデータ・モデル作成者は、‘バッチ優先’設計の前提を改めるときが来ました。将来を想像したとき、最新でないデータがデバイスに到着するのを何時間も待つことは考えられません。‘ストリーム優先’のITインフラストラクチャを計画し、モデル化することが現在可能であり、それが動的なデータ・ファブリックと信頼性のあるデータ・メッシュを計画する上での基盤となります。

古い前提5：ハブアンドスポーク型アーキテクチャに対する先入観

メインフレームとマイクロコンピュータ（DEC VAXなど）に常にリモート端末があったように、ハブアンドスポーク型アーキテクチャは、当初からコンピューティングの決定的なパターンでした。パブリック・クラウド・コンピューティングのデータ・レイク概念が台頭するなかで、私たちは現在もなお、データとワークロードを1つの集中型の物理ロケーションに集約したいという衝動に自然に駆られています。しかしながら、世界は隠喩的に縮小し、グローバル展開するビジネスは増加しているため、エンタープライズ・データは必然的に、さまざまなアプリケーション、クラウド、エッジ・デバイス、オンプレミス・システムにまたがる無数のインフラストラクチャでさらに断片化されます。

このような古いハブアンドスポークの前提を改めれば、多数のエンタープライズ・データ・プロデューサとデータ・パイプラインが、DAG（有向非巡回グラフ）すなわちメッシュとして、自然な状態に効率的にモデル化される未来を作り上げることができます。ワークロード（データの取込み、台帳イベント、変換、ストリーム分析など）はグラフの任意のノードで発生することが可能です。モノリシックなハブ（ETLツール、EDW、データ・レイクなど）は必ずしもなくなるわけではありませんが、概念上、これらのモノリスは、より広範なデータ・ファブリックの単なる小さい一部分となります。ファブリック/メッシュが導入されたら、俊敏性を向上してモノリスの代わりとなるものを試すことができるよう、ITを解放します。データ・モノリスは、数十年前の古いITの‘ゴミ’[xii]や‘技術的負債’になるのではなく、ビジネス・ニーズが発生した場合に簡単に繰り返し適用できるよう、ある時点で発展させることができます。‘ハブ/モノリスのグラフ’はデータ・メッシュと同じものではないことに注意してください。

古い前提6：ストレージ中心のモデリング

データは動的です。すべてのデータはイベントとして素早く生成されます。極めて価値の高いデータは通常は移動中であり、生成されてから数ミリ秒以内に利用されます。しかし歴史的に見て、データはデータ・モデルとデータ・アーキテクチャ全体で、アルミニウム製のハード・ディスク・ドライブのどこかに1と0として存在する静的なものであるかのように扱われています。データに対するこのストレージ中心のアプローチは、データの属性が必然的に変わるなかで、徐々に柔軟性に欠ける融通の利かないアプローチとなります。統合の支出に関連するコストの中で高い割合を占めるようになるのは、ドキュメントのペイロード、アプリケーション・アップグレード、ETLマッピング、EDWのディメンション変更におけるこの‘データ・ドリフト’です[xiii]。

移動中のデータについての避けられない未来に備えるには、トランザクション、イベント・ログ、データ・アーティファクトを、本質的に一時的な場所に存在するオブジェクトとしてモデル化するというデータ思考を改める必要があります。チェス盤のように、任意の時点における駒の場所は1つのスナップショットにすぎないため、一連の移動（イベントなど）によって、チェスの対局の全容が実際に分かります。同様に、エンタープライズ・データの全体像を完全に把握するには、別個のビジネス・オブジェクトのスキーマとデータ価値がどちらも時間とともに変わるといったデータの動的な性質を事実に基づきモデル化できなければなりません。

なぜ今なのか：実現化テクノロジー

1つの発明やプログラム言語がデータ・ファブリックやデータ・メッシュのイノベーションを促進するわけではありません。大規模なアーキテクチャの変化が、稲妻のように発生する1つの飛躍的進歩によって起きることはまれです。多くの場合、そのような変化は、実現化テクノロジーの普及が進み、使用が容易になるにつれて、小規模な一連のイノベーションや段階的な改善から徐々に現れてきます。これらの実現化テクノロジーは、システム規模の新しいアーキテクチャの可能性が出現する基盤としての役割を果たします。

先ほどのセクションでは、改めるべき古い思考法と前提を数多く説明しましたが、数々の重要なテクノロジーの段階的な改善についても、詳細を特筆すべきでしょう。このような基盤となる技術のイノベーションにより、数年前でさえも全く不可能だった方法で、動的なデータ・ファブリックと信頼性のあるデータ・メッシュが実現します。

実現化テクノロジー1：サービス・メッシュ、Kubernetesなど

2015年以降、Apache Stormによるソフトウェア開発エコシステムを取り入れたマイクロサービス・アーキテクチャの大変革は、DevOpsとCI/CD（継続的インテグレーション/継続的デリバリ）にアプローチする方法も一変させました。KubernetesやOpenShiftなどのツールで、ワークロードのコンテナ化（Dockerなど）と‘サイドカー・パターン’（分散管理向け）の高負荷のアプリケーションを組み合わせることにより、これまで全く不可能だった驚くべき方法で、アジャイルなDevOps手法を適用できるようになりました。たとえば、パッチ適用、アップグレード、コード・インジェクション、再プラットフォーム化、自動スケーリングといった従来型のDevOpsタスクは、現在は通常、サービス・メッシュのpod管理制御を使用して、これまでよりもはるかに低リスクで簡単に実行できます。

データ・ファブリックやデータ・メッシュといった次世代のデータ管理概念が進化するなか、サービス・メッシュ・テクノロジーのアプリケーションは、より素晴らしい俊敏性、改善されたDevOps、CI/CD、および優れたDataOpsプロセス手法の基盤を実現するための基本的な手段です。

実現化テクノロジー2：成熟したSDN（Software Defined Networking）

効果的なマルチクラウド運用を遂行するITの能力にSDN（Software Defined Network）が及ぼす影響は、どれほど誇張してもしすぎることはありません。ドメインSDNが台頭する以前は、謎に包まれたネットワークはもっぱら、高度な専門知識を持ち、物理的なスイッチ、ゲートウェイ、ルーター、ロードバランサ、VPNハードウェアなどを扱うネットワーク・エンジニアの領域でした。効果的でセキュアなネットワーク構築は現在も依然として非常に困難な課題ですが、たいていは主要なクラウド・ベンダーのソフトウェア層内から管理できるようになりました。セキュアなドメイン、テナンシー、IPSec VPNトンネル、VNC（仮想ネットワーク）、プライベート・エンドポイント/リバース接続などの全域で、ネットワーク・トラフィックをすべて、APIとクラウド・ユーザー・インタフェースを使用して調整できることは驚異的です。

このSDNの簡素化により、万能型のITスタッフが、セルフサービスのネットワーク・タスクをより多く担うことができるようになったことで、かつてないほど分散化されたインフラストラクチャ全域でデータ/アプリケーションを管理しやすくなるという大きな変化がもたらされました。すべてのデータセンターでデータとワークロードを極めて柔軟に移動できることにより、動的なデータ・ファブリックと信頼性のあるデータ・メッシュに対する（i）需要が促進され、（ii）新機能が強化されています。

実現化テクノロジー3：CDCレプリケーションのマイクロサービス

チェンジ・データ・キャプチャ（CDC）ツールはデータ・ストア内のイベントを検知し、データ・レプリケーション・ツールはデータをLAN/WANに移動します。これらのツールは、常にではありませんが、頻繁に一緒に使用されます。このようなツールは当初、他のデータ統合ツールと同じく、その領域につきものの欠点をすべて抱えたモノリシックなアプリケーションでした。しかしながら2016年以来、一部のツール（Oracle GoldenGateについては本書の第4部で詳しく説明します）は、RESTベースのAPIによって100 %駆動され、完全にカプセル化された軽量サービスを持ち、メタデータのために外部データ・ストアに依存しない真のマイクロサービス基盤へと変化してきました。技術的に見ると、この基盤のリファクタリングによって、サービス・メッシュが強化され、DevOps、CI/CD、およびアジャイル開発の可能性に著しい影響が及びます。

実現化テクノロジー4：DBレプリケーションにおけるポリグロットとACIDペイロード

1990年代に遡ると、データ・レプリケーション・ツールは、ACID（原子性、整合性、独立性、永続性）に対応したデータベースをソースとする整合性の高いデータ・トランザクションのレプリケーションに重点を置いていました。論理的なレプリケーションを使用して、大規模な高可用性（HA）、ディザスタ・リカバリ（DR）、Oracle Global Data Services（Oracle GDS）、地理的に分散されたデータ・シャーディングをサポートしていました。このように重点を絞ることには、ACIDのプロパティを必要とするデータ・トランザクション（銀行、フィンテック、電気通信、バックオフィス・ツール、ERPシステムなど）の信頼性を確保できるという重要な利点がありました。データ・レプリケーションを使用して、私たちはデータ整合性が失われないという確信を持ちながら、物理的に非常に離れた距離の間でデータを迅速に移動できます。

2015年以降、ビッグ・データ、データ・レイク、NoSQLドキュメント・ストアの普及が広がり、とりわけ非トランザクション・ワークロードをサポートする分析データ・ストアで、‘データ整合性が厳格でない’多数の有効なユースケースが導かれました。そのため、一部のデータ・レプリケーション・ツール（Oracle GoldenGateを含む）は、非リレーショナル・データのペイロードにも対応するピボットをすでに作成しました。最新のデータ・レプリケーション・ツール（Oracle GoldenGateなど）では、‘ポリグロット・レプリケーション’が可能になりました。つまり、XML、JSON、Avro、ORC、Parquetといった、多くがフォーマットされたペイロードと同様に、厳格なACIDデータ整合性トランザクションにも対応できます。

普及しているオープンソースのメッセージング・フレームワーク（ACIDに対応していないApache Kafkaなど）とは異なり、データ・レプリケーション・ツール（Oracle GoldenGateなど）は、ACIDトランザクションを正しく処理でき、待ち望まれていた水準の信頼性をデータ・メッシュで実現できます。

実現化テクノロジー5：ETLをストリームで実行可能

1990年代以来、ETLツールはモノリシックなアプリケーションを基盤としており、バッチ処理が必要でした。ETLジョブは単一のデータ・イベントから実行されるのではなく、スケジュールに基づき実行されます。もちろん、以前からイベント駆動型メッセージング・エンジンを使用してデータを変換できましたが、データベース・ワークロードに必要な規模で変換を行うことはできませんでした。

2015年頃から、イベント・ストリーム・プロセッサ（Apache Stormはその初期の例）が事業化されたことで、LambdaアーキテクチャやKappaアーキテクチャと呼ばれるビッグ・データ・アーキテクチャの可能性が広がりました。LambdaアーキテクチャとKappaアーキテクチャはどちらも、スケール・アウト・データ・ストリームでETLを提供しますが、Kappaアーキテクチャは、バッチ処理とストリーム処理のワークロードを単一のエンジンに統合します。当初のKappaテクノロジー・スタックは、コンシューマ・データ（ログ・イベント、ソーシャル・メディアなど）に重点を置いていましたが、基盤となるテクノロジーは、現在はエンタープライズ系データ・ワークロードに十分に対応できるものに進化しました。

エンタープライズETLを連続したストリームで実行できるテクノロジーが基本となっており、考え方が一変しています。つまり、スケジューラを待つのではなく、利用可能になったデータ・トランザクションをすぐに変換してロードできるようになりました。スケジューラで必然的に発生する‘時間枠’をなくすことで、データ・パイプライン自体で処理時間枠を管理できるようになりました。
ACID整合性を持つエンタープライズ・データは、通常は代数のセット（joinなど）で処理され、ストリームETLツールは、joinが発生する時系列のデータセットで高い精度を維持できます。

バッチ処理の時間枠を排除することは、長い間、データ統合における難題でしたが、以前は不可能だった大きな規模で実現できるようになりました。

実現化テクノロジー6：無料のMPP DAGとサーバーレス

2015年以降のITにおけるもっとも歴史に残る2つの変化は、パブリック・クラウドとビッグ・データ・テクノロジーの台頭です。これらのマクロトレンドはどちらも、根底となるのはソフトウェア・アプリケーションのワークロードのスケーリングでした。パブリック・クラウドでは、他人のコンピュータを使用するというだけでなく、実際に使用した正確なコンピューティング量に対してのみ課金される‘サーバーレス’コンピューティング・モデルを採用しています。ビッグ・データ・テクノロジーによってもたらされた、Hadoop（MapReduce）から離れてDAG（有向非巡回グラフ）ベースのMPP（大規模パラレル・プロセス）フレームワーク（Apache Spark、Flinkなど）へと向かう変化のおかげで、極めてスケーラブルでインタラクティブなランタイム・エンジンを、Apache gzipパッケージをダウンロードできる誰もが利用できるようになりました。

これまでは、大規模な（数百または数千のプロセッシング・コアを使用する）ワークロードの実行は、高額で途方もない作業でした。しかし現在は、クラウドでホストされる従量課金制のサーバーレス・エンジンのビッグ・データ・フレームワークを使用すれば、クレジット・カードを持つ誰もが数分で作業を開始できます。このような徹底的な簡素化には、‘データ重力’を低減し、データを使用して実験する傾向を増加させる実質的な効果があります。ストレージとMPP演算の相対コストは0円に近づきつつあるため、自然な結果として、データは増加し、ワークロードがさまざまなインフラストラクチャ間を移動する頻度は増します。

大企業のIT組織は元々、レガシー・システムと一連の新しいSaaSクラウド・アプリケーションで構成される異機種混合のインフラストラクチャを持ち、複数のクラウド・プラットフォーム・プロバイダと契約しています。クラウド・コンピューティングの台頭により、データ・プロデューサの断片化が増加しました。この断片化は、かつては単一のデータセンター内の少数のアプリケーションにのみ存在していましたが、そのような日々は永遠になくなりました。

安価で簡素なMPPフレームワークとサーバーレス・コンピューティングの可用性が広がったことで、動的なデータ・ファブリックと信頼性のあるデータ・メッシュに対するかつてないほど桁違いの需要が促進されるでしょう。これらの基本機能により、データ・ファブリック/データ・メッシュがオンデマンドで大規模かつ安価にワークロードを実行するインフラストラクチャも実現します。

重要な目標と具体的な利点

この新しいタイプのデータ・アーキテクチャにより、より素早いイノベーション・サイクルと運用コストの削減が可能になります。早期採用者により、現在次のような意義深い多くの利点を享受できる可能性があることが証明されています[xiv]。

- **データのバリュー・チェーンの完全な透明性** – 'データ・プロダクト思考'のベスト・プラクティスを適用
- **業務系データの99.999 %以上の可用性** – レプリケーションにマイクロサービス・ベースのデータ・パイプラインを使用
- **10倍高速なイノベーション・サイクル** – ETLから連続変換およびロード（CTL）に移行
- **データ・エンジニアリングの最大70 %の削減** – ノーコードのセルフサービス式データ・パイプライン・ツールを使用

常時運用に関連するコストを削減すると同時に、データを競争上の優位性として活用する各事業部門でイノベーションの強化を図ることもできるソリューションを見つけることは非常に難しく、ここではまさにそのような機会を提供しています。データを管理する方法を再考し、最新のデータ・ファブリック/メッシュを導入することで、企業は競争上の利点を得るためにより自由にデータを使用できます。

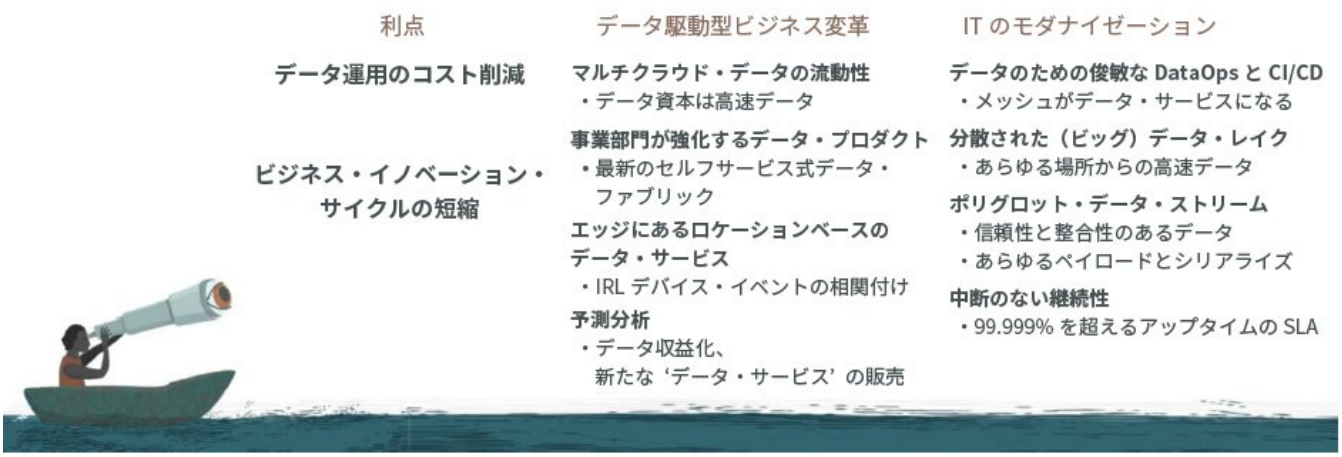


図3:動的なデータ・ファブリックと信頼性のあるデータ・メッシュが導く具体的なビジネス成果

意義あるビジネス変革の達成を目指している組織は、人材とプロセスの変更を重視していますが、テクノロジーも重要な役割を担います。エンタープライズ・テクノロジーは、単体では大規模な変化に影響を与えることはできませんが、文化的な変化と組織的な変化とともに1つの手段として使用すれば、データ主導のビジネス変革を可能にする強力な促進剤になり得ます。

テクノロジーのために見境なくテクノロジーに費やすことができる新規の純予算を持つ組織はほとんど存在しません。コスト削減などの戦術的利益を土台に新たなアプローチを採用することが重要であるのはそのためです。データ・ファブリックとデータ・メッシュの機能は、ITのモダナイゼーションへと向かうもっとも重要な最初の経路であり、既存のデータ資産で業務系データのレディネスと必要なデータ統合、データ・パイプライン、データ・ガバナンスを維持するための従業員の占有面積を縮小することに直接つながる可能性があります。多くのサポート予算と人材を必要とする高額なデータ・モノリスは、段階的になくなる可能性があります。一方、最新のメッシュベースの統合プラットフォームによって、継続性のSLAとポリグロット・データ・パイプライン機能が提供され、分散されたデータ・レイクへの具体的な道筋が示されます。

データ・インフラストラクチャの変革が進行している一方で、データ主導のビジネス変革は現実味を増しています。より迅速で信頼性のあるデータ・フローは、より信頼できる予測分析とデータ収益化機会を意味します。実世界（IRL）の片隅で物理的な機器を運用しているビジネスでは、特別な専有ツールでのみ実行されるデータ・イベントを、全体的なデータ資産で利用できるようになりました。

データ主導のビジネス変革における最終フェーズは、‘データ・プロダクト’の所有権が事業部門（LoB）にあるものの、スマートなデータ・ファクトリがITによって運用されている場合に具体化され始めます。ITは、データ・ファクトリを製品としてLoBに提供し、LoBは自身でセルフサービス方式によってデータ・プロダクトを作成し、収益化します。データとデータ・プロダクトのこの継続的なフローは、複数のクラウド・エコシステムでシームレスに発生します。動的なデータ・ファブリックと信頼性のあるデータ・メッシュにより、企業はデータとデータ・ワークロードを、ビジネスにとってもっとも理にかなっているクラウドに移動できるようになるため、単一のクラウド・ベンダーによって‘ロックイン’されることはなくなります。

データの流動性

データの流動性とは、データをその起点から使用される多くの地点に効率的に移動できることです。データの流動性の本質は、データを新たな用途のために再利用する時間、コスト、労力を削減することです。

より迅速、容易、かつ安価にデータセットを新たな用途のために再利用できるほど、流動性が高まります。

各デジタル・イベントは、アプリケーションから見て、迅速かつ容易に取得できる特定の構造で作成されます。これらの構造を異なる形状と考えてください。表の行として生成される構造もあれば、オブジェクトにバンドルされる属性と値のペアのセットとして生成される構造や、グラフのノードやエッジとして、またはリストに追加されるテキストの行として生成される構造もあります。しかしながら、すべての業務系データの事例、分析システム、AIシステム、その他のアプリケーションでは、独自の目的のためにわずかに異なる形状のデータが必要です。

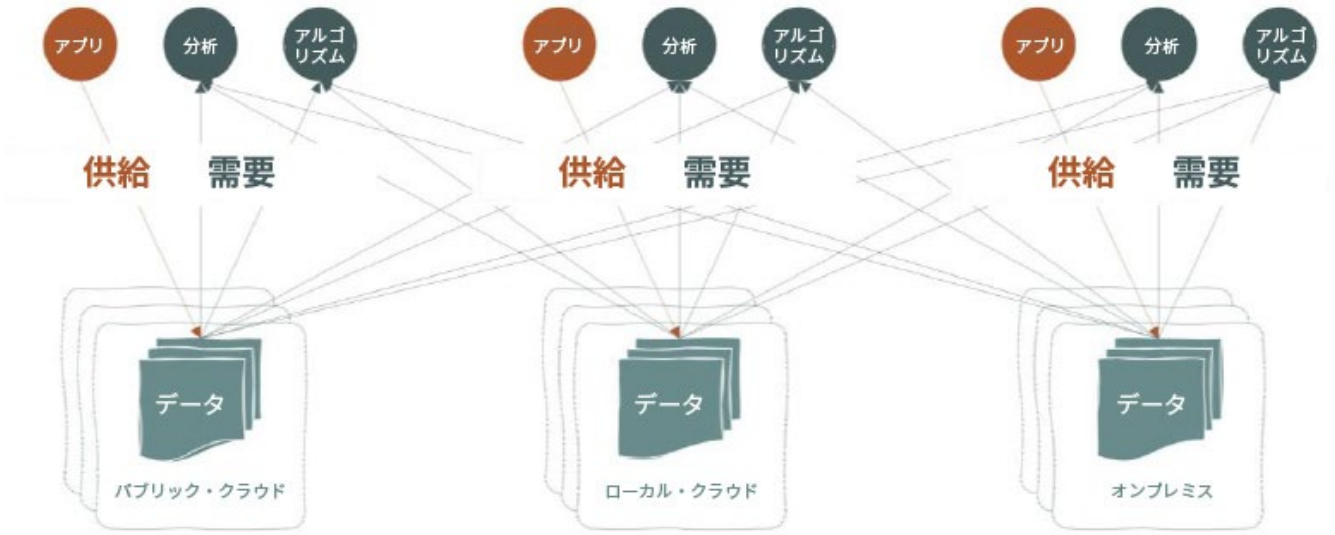


図4：データの供給がいかに迅速かつ容易に需要を満たすことができるかを示す1つの指標としてのデータの流動性

データの流動性を高める上での課題は、データの形状を変えることができるようにすることです。必要な形状のデータをアプリで作成できるようにすると同時に、独自のタスクにとって必要な別の形状でデータが作成された他の使用場所で、そのデータを迅速かつ容易に表示できるようにするという課題は、時間とともに難しくなるばかりです。魔法のランプから出た魔神を戻すことはできません。SaaSと低コストのIaaSクラウドの台頭により、エンタープライズ・データをどこにでも保管して好きなときに移動させることは、かつてないほど安価で容易です。エンタープライズ・データが急増し、分散化するなかで、データ・ファブリックとデータ・メッシュ用のツールは、データ・ガバナンスらしきものを確保できる唯一のデータ管理インフラストラクチャになるでしょう。

そのため、信頼性のあるデータ・メッシュと動的なデータ・ファブリックは、現代のデータ・アーキテクチャのイノベーションとって必要不可欠なツールとなります。主にデータ・ガバナンス、セキュリティ、信頼性のあるデータ・トランザクションを懸念している組織にとって、この動的なデータ・ファブリックは、データを使用する多くの物理ロケーションにデータを移動するデータ・パイプラインの来歴と信頼を保つ上で不可欠です。イノベーションや収益化のためにデータを使用することにより関心がある他の組織では、データの流動性は、新たなビジネス機会を繰り返し迅速に改善しながら、データを他の目的のために素早く再利用するための強力な促進剤となります。かつてはITにとって論理的な俊敏性という利点であったものが、瞬く間にグローバル市場における競争上の優位性という戦術的な強みになるでしょう。

データ・アーキテクチャの新しいパラダイム

前の2つのセクションでは、データ管理の考え方の複数の側面をどのように改める必要があり、いかに複数の主要テクノロジーが促進剤となってこの分岐点が導かれたかについて説明しました。行動の変化とテクノロジーの変化が交わることで、マルチクラウドの動的なデータ・ファブリックと信頼性のあるデータ・メッシュの新たな設計方法が可能になります。これらの変化は以下のようにまとめることができます。

改めるべき古い思考法	実現化テクノロジーの変化
• データを副産物として扱う • OLTP→ETL→OLAPの断片化 • データ・プロジェクトのウォーターフォール・プロセス • バッチ優先の設計思考 • ハブアンドスポーク型アーキテクチャに対する先入観 • ストレージ中心のモデリング	• サービス・メッシュ、Kubernetesなど • 成熟したSDN（Software Defined Network） • マイクロサービスCDC/レプリケーション機能 • 1つのトポロジのACID/ポリグロット向けレプリケーション • データ・ストリームにおけるETL • オープンソースMPP DAGおよびサーバーレス・コンピュート

こうした動作における変化とテクノロジーの変化の状況を受けて、動的でリアルタイムなエンタープライズ・データ・ファブリックと、信頼性のあるエンタープライズ・データと適切に連携されるデータ・メッシュという新しいアプローチが可能になりました。

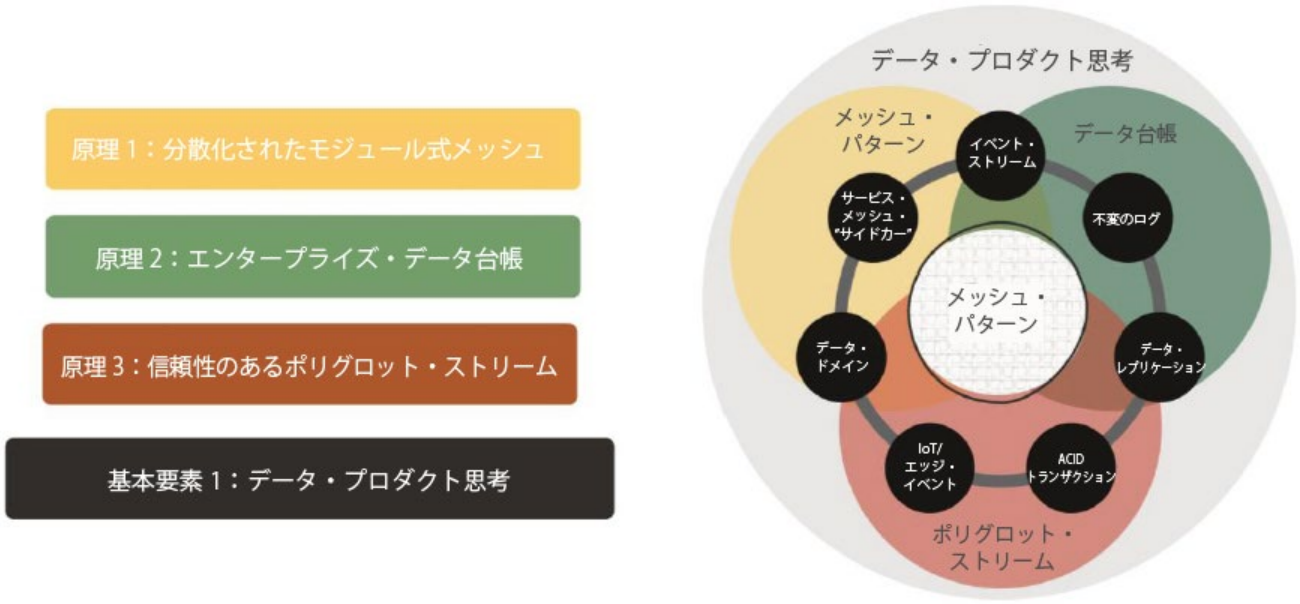


図5：動的なデータ・ファブリックと信頼性のあるデータ・メッシュの重要な原理

この動的なデータ・ファブリックと信頼性のあるデータ・メッシュの概念には、1つの基盤となる‘思考法’と3つのテクノロジー原理があります。データ・プロダクト思考は、このソリューション領域にかかわるあらゆる人、プロセス、テクノロジーに吹き込まれ、浸透する必要がある思考法であるため、基礎を成す要素です。1つ目のテクノロジー原理は、フレームワークとデータ・サービスがデフォルトで分散化され、高度にモジュール化される必要があることです。この分散化は、CICD、アジャイル、および持続可能なマルチクラウド・データ・アーキテクチャを実現する上で非常に重要です。2つ目の原理は、データ/イベント台帳を通して、データ・サービスがデータと可能な限り連携する必要があることです。これらの台帳により、‘データの時間’を行き来できるほか、可能な限り高い忠実性をデータ・イベントで実現できます。3つ目の原理は、ポリグロット・データ・ストリーム、すなわち‘多くの形状’をとることができるストリーム・データをサポートすることです。エンタープライズ・データは、多種多様なデータ形式とトランザクション・セマンティックを持つ可能性があるため、特定の種類のポリグロット・ストリームに特化した分散化された台帳/ストリーム・テクノロジーが数多く存在する可能性があります。データ・プロダクト思考とこれら3つのテクノロジー原理を併せると、新しいタイプのデータ・アーキテクチャへのロードマップが完成します。

基本要素：データ・プロダクト思考

データ・ファブリックまたはデータ・メッシュを作成する目的は、より多くの価値をエンタープライズ・データにもたらすこと、すなわち、ビジネス変革の目標をより適切、迅速、かつ安価に達成できるようにすることに尽きます。データ・プロダクト思考のアプローチは、取り入れなければならない、もっとも基本的で基盤となる考え方の変化です。考え方を修正できなければ、テクノロジーのためのテクノロジーを採用するという落とし穴に極めて容易に陥ります。

一般コンシューマとして、私たちは皆、製品やサービスを毎日のように購入しています。私たちは朝のコーヒーからコンピュータや車といったより大きな買い物に至るまで、製品の価値を日々の生活で本質的に理解します。その価値は、ほとんどの場合、私たちが支払ってもいいと考える価値と全く同じです。ではデータ・プロダクトはどうでしょうか。データが製品であるというのはどういう意味でしょうか。

データ・プロダクト思考は、基本的にデザイン思考[xv]手法の集まりであり、データ・サイエンス・コミュニティから生じたデータ・プロダクト概念です。データ・プロダクトの作成とキュレーションに対する顧客中心のアプローチを定義しており、ビジネスとデータ・コンシューマのニーズを一致させようとするものです。

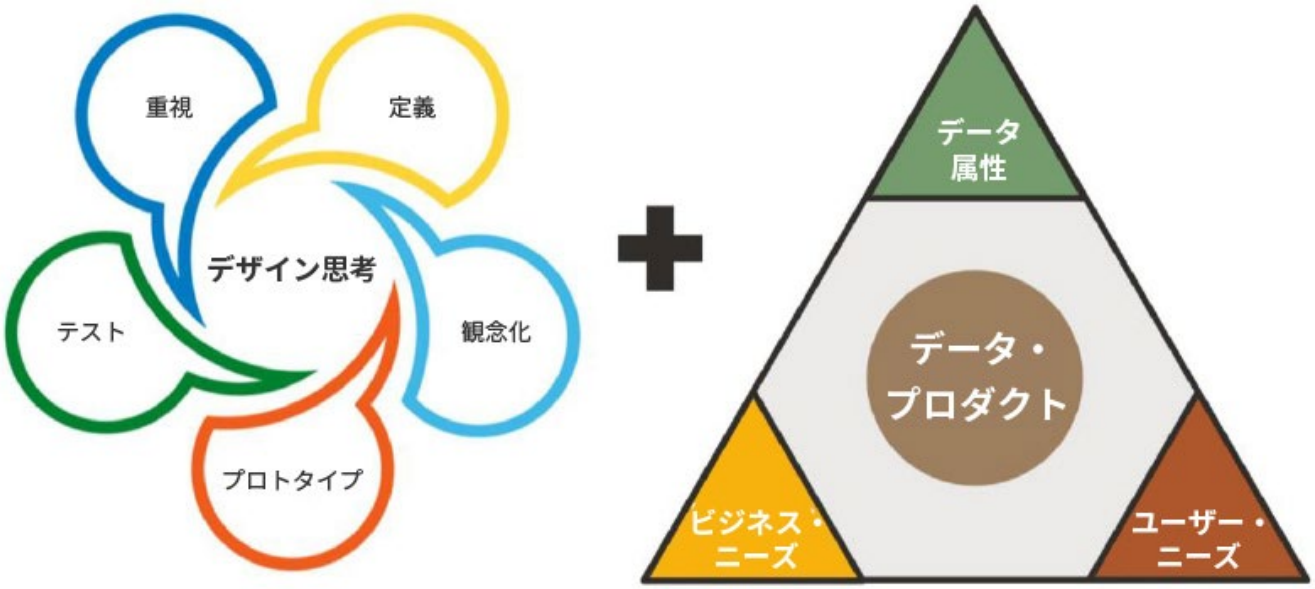


図6：価値を重視したデータ・プロダクト思考は、デザイン思考手法と、データ・サイエンスから生じたデータ・プロダクト概念から影響を受けている

データを明確な価値と暗黙的な価値を持つ資産として十分に活用するようになれば、データは必然的にKPI（キー・パフォーマンス・インディケータ）とSLA（品質保証契約）に連携されます。データを単なるアプリケーション・ワークロードの副産物や、IT部門が遠くの保管庫にある安価なテープ・バックアップのごみ箱に追いやるべきものとして捉えなくなります。データ・プロダクト思考では、データを適切な価値の資産として扱い、データ管理に製品管理プラクティスを適用します。本書では後ほど、製品としてデータを管理する際の主要な4つの属性について説明します。

原理1：分散化されたモジュール式メッシュ

動的なデータ・ファブリックとあらゆるデータ・メッシュの概念の中核は、データ・サービスの分散化された物理アーキテクチャです。データのプロデューサは、エッジ、エッジ・ゲートウェイ、オンプレミスのデータセンター、パブリック・クラウド・ネットワーク、SaaSアプリケーションなど、どこからでもデータを生成できます。動的なデータ・ファブリックは、本質的にデータ・イベントとのリアルタイムの連携を意味しているため、データが生成されるネットワークのどのような場所でも、ファブリック/メッシュのニーズがあるはずです。分散化されたメッシュ（WiFiメッシュなど）のように、コントローラ（コントロール・プレーン）が存在しますが、メッシュ・アーキテクチャでは、メッシュ内の任意のノードは、メッシュ内の別の任意のノードと通信でき、任意のノードにデータを中継できます。そのため、集中型/ハブベースのルーティングは必要ありません。自宅にWiFiメッシュがある場合、まさにそれと同じように機能します。

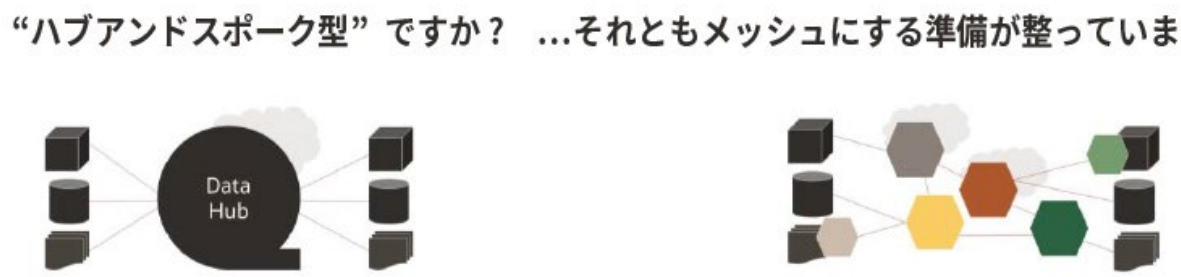


図7：ハブとメッシュのトポロジの概念上の違い

この固有の‘メッシュ状態’は、ストリーム・データを毎回単一の集中型ハブを通さずに、さまざまなパスを介してルーティングできるマルチクラウド・エコシステムで効率的なデータ運用を実現するのに役立ちます。

メッシュがデータそのもののメッシュでない理由

成功を収めているエンタープライズ・データ・アーキテクチャは、データを望む形式で利用するのではなく、そのまま利用します。極めて典型的な大企業には、数十年分のデータ形式が存在し、それらは今すぐなくなることはありません。これは、エンタープライズ・データにおける実情です。‘壊れていなければ修復しない’というITの哲学は広く浸透しています。つまり、レガシー・データは単なる現実にすぎません。
ほとんどのデータは、スキーマがないか、完全に非正規化されており、少数の貴重なデータ形式が自己記述的または自己結合型です。WebスケールでURIにリンクできる特定の表現豊かなデータ標準（RDF（Resource Description Framework）、OWL（Web Ontology Language）など）は、十分に広く浸透していないか、整合性が十分でないため、統一フォーマットとして利用できません。
同様に、ベンダーと企業は通常はどちらも、他に従うのではなく独自の共通語を持つことを好むことが、あまり成功しなかった40年以上におよぶ業界の語彙力によって実質的に証明されています。一部の業界では、基盤となるデータ・モデルは基本的な知的財産（IP）と見なされ、保護されるべきであり、共有されるべきではないと考えられています。
そのため、データ・ファブリックとデータ・メッシュは、データそのものの一種のメタグラフではなく、あらゆる種類のデータを効果的に移動して管理する機能を提供する必須サービス（データ変換、ストリーム、カタログなど）のファブリック/メッシュです。

原理2：エンタープライズ・データ台帳

動的なデータ・ファブリックのコンテキストにおける‘動的’という言葉は、このアーキテクチャのアプローチが持つイベント駆動型の性質を指しています。アーキテクチャの中核的機能としてのデータ・イベントには、ログ・イベント、メッセージ、およびイベントを消費するさまざまな読取り側の場所を記録する台帳が必要です。

エンタープライズ・ソフトウェア・システムのデータ台帳は、数十年の歴史を持ちます。1960年代に専用のトランザクション処理システム（TPS）が作成されました。TPSは実質的にアプリケーションとデータ・トランザクション用のイベント台帳であり、IBMのCICSやOracle TuxedoなどのTPSは現在も稼働し続けています。ほぼすべてのSQLデータベースは、内部オンラインREDOログを中心に構築されているため、データ・イベントの厳格な整合性を実現するトランザクション台帳として機能します。その他の最近のエンタープライズ台帳は、Apache KafkaやPulsarに基づいている場合もあれば、Hyperledgerなどのブロックチェーン・テクノロジーに基づいている場合さえもあります。

台帳ベースのプラットフォームは、多くの場合、次のような特殊なユースケースに対応するように設計されています。

- **アプリケーション・アーキテクチャ** – コンポーネント間の通信で‘イベント・ストア’に依存するイベント・ソーシング・パターンを活用するマイクロサービス・ベースの台帳ソリューションの普及が増加しています

- **インフラストラクチャ・ログの取込み** – エンタープライズ・インフラストラクチャのテレメトリ（運用動作）を収集するイベント台帳。Apache Kafka、Pulsar、AWS Kinesisなどのツールが一般的な例です
- **エンタープライズ規模のデータ・サービス** – 単純なログの取込みとは異なり、企業のユースケースでは、KafkaやPulsarといった同じ台帳もイベントベースのデータ・サービスでより広範に使用されています
- **複数パーティのブロックチェーン台帳** – 透過性と不変性のために採用されたブロックチェーン・テクノロジーが、組織を横断した（B2Bなどの）交換で使用され、極めて信頼性の高い方法でトランザクションを追跡しています
- **分散されたデータベース・トランザクション** – （'シャード・データ'や高可用性などのために）グローバルに分散されたデータベース・トランザクションが、Oracle GoldenGateなどのデータ・レプリケーション・ツールを使用してイベントをレプリケートします

最後のカテゴリである分散されたデータベース・トランザクションは、企業のユースケースにおける極めて重要な側面です。大多数のエンタープライズ・アプリケーションは、整合性が極めて高いリレーショナル・データベースで実行されているため、分散化されたデータの共有へと移行するには、ネットワークの境界を越えてリレーショナルの整合性を維持するACID対応のトランザクションが必要です。Oracle GoldenGateなどのツールは実際、データが数千キロメートルの距離を横断してレプリケートされる場合や、さまざまなポリグロット・データ・ストアでレプリケートされる場合でさえも、データの信頼性と整合性を保証できるように特別に構築された分散トランザクション台帳です。ACIDレベルの整合性というこの要件は、OLEPシステムに関する研究で言及されています。

オンライン・イベント処理（OLEP）

データを中心に据えたエンタープライズ・ドメイン向けの汎用オンライン・イベント処理（OLEP）システムの概念が最初に探求されたのは、Kafkaの考案者の1人であるJay Kreps氏が2013年に発表した独創性に富んだ記事、『The Log:What every software engineer should know about real-time data's unifying abstraction』[xvi]でした。この記事では、さまざまエンタープライズ・データソースからログ・イベントを取得でき、それらのイベントを任意のデータ・コンシューマが容易に利用できるようにするエンタープライズ・データ台帳についての考え方の基礎が提供されています。最近では、Martin Kleppmann氏たちが分散OLEPシステムの一部としてACIDに似たデータ整合性を求めて、この概念をさらに推し進めようと試みました（『Online Event Processing: Achieving consistency where distributed transactions have failed』[xvii]）。この一連の考え方の中心的概念は、イベント・ストアまたはメッセージ・システムを、戦術のための部分的なソリューションだけでなく、エンタープライズ規模のソリューションでより広範に活用することです。

図8：あらゆる種類のデータ・イベントの統一ソースとしてのエンタープライズ・データ台帳

エンタープライズ規模のイベント台帳機能は、必ずしも、会社全体で台帳が1つでなければならないことを意味しているわけではありません。イベント台帳のエンタープライズ性は、一連のエンタープライズ・クラスの機能を意味しています。大量のイベントをスケーリングする機能、柔軟な一連のトランザクション・セマンティックに対応する機能、ペイロード構文、パブリック・クラウドで実行される場合やサービス・メッシュ（Kubernetes、OpenShiftなど）の一部として実行される場合のサーバーレス・オプションなどがその例です。実際、動的なデータ・ファブリックまたは信頼性のあるデータ・メッシュでは、さまざまなエンタープライズ・データ台帳が存在している可能性があり、それらの台帳はおそらく、一意のデータ・ドメインによってセグメント化されるか、組織の異なる事業部門によって運用されています。共有されるカタログやレジストリが、複数の台帳にまたがるデータを連携し、管理する役割を果たすことになります。

原理3：信頼性のあるポリグロット・データ・ストリーム

あらゆるデータ統合ツール（データ・レプリケーション、ETLエンジン、データ・フェデレーションなど）の中心的機能は、データベース・トランザクションの信頼性を常に基盤としてきました。データ整合性を維持する上でのこの信頼性は、非常に重要な領域であり、30億ドル超のエンタープライズ・データ統合ツール市場[xviii]を他の統合方法と切り離してきました。データ・ファブリックは最新の次世代データ統合ツールであるため、データ・ファブリックには信頼性のあるデータの基盤となる機能が本来備わっています。しかしながら、データ・メッシュはデータ・ファブリックとは別に存在できるため、データ整合性を保証せずにデータ・メッシュを構築することは十分に可能です。実際、一部のマイクロサービス中心のデータ・メッシュ概念では、データの信頼性と整合性を、アプリケーション開発者の手に委ねています（段階的なコミット、トランザクションの相殺、およびリカバリ・ロジックのエンコード）。ただし、そのようなアプローチにはリスクが伴います。

本書におけるこの‘信頼性のあるデータ・メッシュ’の概念は、（ACIDプロパティのデータセットで）必要に応じてデータを保証しながらも、高い水準の信頼性を維持する必要のないデータ形式とトランザクション・タイプとも同様に連携できる十分な柔軟性を備えた信頼性のあるツールを使用するという原理に基づいています。

データ信頼性の特性	
一度だけのメッセージ処理	分散システム（メッセージングやデータベース）では、システムを構成する各コンピュータで、互いに無関係に障害が発生する可能性があります。一度だけの処理により、メッセージ・プロデューサがメッセージを何度も送信しようとしている場合に、そのメッセージがエンド・コンシューマに一度だけ送信されるように導くことが保証されます。
イベントの厳格な順序	アプリケーション・プロデューサは通常、第三者の正規形データ構造にデータを永続化します。データが保存される際、トランザクションの正確な順序はデータ整合性にとって極めて重要です。平凡な例として、表間で外部キー・プロパティを持つItemHeader表とItemDetail表が挙げられます。更新がシーケンス通りに実行されないと、更新のシーケンスによって整合性/コミットの失敗が引き起こされる可能性があります。
トランザクションの原子性（A）	イベントの順序と同じく、データ・トランザクションは通常、一般的な“コミット・ポイント”でグループ化されます。これにより、コミットがさまざまなSQL文で構成されているにもかかわらず、データベース・システムはトランザクション全体の原子性を保証できます。原子性のあるトランザクションは他のペイロードに変換され、メッセージ・システム（エンタープライズ・データ台帳）に置かれます。トランザクションが分散システムを移動するなかで、メッセージ・システムは、トランザクションの全体的な原子性を存続できなければなりません。
トランザクションの整合性（C）	本書の目的として、データに参照整合性が必要な場合は、トランザクションそのものの整合性に重点を置いています。リレーショナル・ペイロードがメッシュを移動するなかで、信頼性のあるデータ・メッシュは、リレーショナル・ペイロードの整合性の保証を存続できなければなりません。トランザクション（場合によっては大規模なSQLグループとして）の行レベルおよびシステム・レベルの整合性は、存続および保証される必要があります。
独立性（I）と永続性（D）	同時トランザクションは、不完全なトランザクションが移動中であっても、システムの整合性状態を存続させるために、あたかも連続した独立性が存在するかのように確実に処理されます。永続性を最大限に確保するために、ファブリックまたはメッシュは、ほぼゼロのRPO（リカバリ・ポイント目標）を実現する必要があります。永続性は基本的に、許容可能なデータ損失の尺度です。
ロールバックとリカバリ	上記のいずれかのタイプの信頼性に違反した場合、もしくはパーティションまたはコンシューマで何らかの計画外災害が発生した場合、ファブリック/メッシュ・フレームワークは、データを整合性のある状態にリストアできなければなりません。

通例、データ統合ツールとデータ・ファブリックは、ACIDタイプのデータ・ワークロードの処理に関して、信頼できると考えられています。しかしながら、動的な（分散された、ストリームの）データ・ファブリックやイベント・メッシュでは、データの信頼性を維持することははるかに複雑になります。アーキテクチャが分散化されており、待機時間が短い大量のイベントがネットワーク上でルーティングされているためです。すべての種類のデータ・メッシュがデータの信頼性と整合性を存続させるわけではありません。

したがって、‘信頼性のあるポリグロット・データ’・メッシュとは、データ管理システムが、一度だけの順序通りの処理、原子性、整合性、独立性、永続性、およびリカバリ可能性といった幅広いデータ信頼性の特性を提供できることを意味します。これらの信頼性のある特性は、任意のレプリケーション/メッセージ・サブシステム経由でデータ・プロデューサのイベントを取得するデータ処理パイプラインから、ウィンドウイング、集計、データ変換が発生する可能性があるデータ処理パイプラインに至るまでの、データ処理全体のパイプラインに対応している必要があります。

信頼性のあるメッシュとしてデプロイされる動的なデータ・ファブリック

本書では、特定の種類のデータ・ファブリック、具体的には動的でリアルタイムなストリーム中心のデータ・ファブリックに主眼を置いています。さらに、本書の主眼は特定の種類のデータ・メッシュ、具体的には信頼性と厳格な整合性のあるエンタープライズ・データ・メッシュにも置かれています。これらの特色ある領域では、これまでにないほど動的なデータ・ファブリックと、無敵の信頼性を持つデータ・メッシュに対するオラクルのアプローチに注目が集まっています。データ・メッシュはデータ・ファブリックの外部に（たとえば、専門化したIoTアプリケーションの一部として）存在でき、同様に、データ・ファブリックはリアルタイムである必要や動的である必要はありません。

動的なデータ・ファブリックと信頼性のあるデータ・メッシュのこのような属性を同時に実現することを目指す組織には、以下のような、モノリスからメッシュに移行するために必要な道のりがあります。

図9：動的なデータ・ファブリックと信頼性のあるデータ・メッシュを適用する利点と機会

データ・ファブリックとデータ・メッシュのビジネス価値について、大胆な主張がなされています。目標の達成は簡単ではなく、魔法の解決策は存在しません。動的なデータ・ファブリックと信頼性のあるデータ・メッシュの包括的なアプローチを採用するには、多くの面があります。本書の残りの部分では、動的なデータ・ファブリックと信頼性のあるデータ・メッシュの重要な属性を体系的に見ていきます。取り上げる内容は以下のとおりです。

本セクションの残りの議題

- **データ・プロダクト思考** - ‘製品として’のデータの管理に対する包括的な考え方
- **運用データ・ストアと分析データ・ストアの連携** – エンタープライズ・データ資産全体を網羅するデータ管理アプローチの重要性
- **データ・モノリスの分解** – 分解戦略の紹介
- **データ・ゾーンとデータ・ドメイン** – 分散データ・アーキテクチャ内のドメインとゾーンの違いの説明
- **メッシュの構成要素** – サービスおよびデータの分散化されたメッシュに欠かせない特性の特定
- **おもなユーザー、セルフサービス・ロール** – メッシュ/ファブリック・ツールを使用するユーザーと、予想される種類のセルフサービスの調査
- **イベント駆動型のストリーム・データ・パイプライン** – リアルタイムのデータ変革がビジネス価値に影響を与える仕組み
- **反復可能なデータ・プロダクト・ファクトリ** – データ・プロダクト生産のファブリック/メッシュを重視することがビジネス価値にとって非常に重要である理由の説明
- **DevOps、CI/CD、DataOps** – ファブリックの運用とファブリックのユーザーの関係
- **データ・メッシュのガバナンス** – 高度なファブリック/メッシュに欠かせないおもな特徴と機能

データ・プロダクト思考

データ・プロダクト思考の起源は、設計者とデータ・サイエンティストに馴染みのある概念が交わったことにさかのぼります。設計側では、デザイン思考[xix]と呼ばれる、十分に立証されたユーザー・エクスペリエンスへのアプローチが存在します。デザイン思考は、製品とサービスについてのイノベーションを生み出すための深く研究された反復的な手順であり、世界中の数千もの組織が、Webサイト、アプリケーション、店舗で購入する物理的な製品をはじめとするあらゆるものの設計において使用しています。データ・サイエンティスト側では、データ・プロダクトの概念は、データと機械学習によって支えられるデジタル製品を説明するために定期的に使用されてきました。もっともよく知られているのは、幅広いビジネス読者層に説明することを目指した、Harvard Business Reviewの2018年の記事、『How to Build Great Data Products』[xx]です。

2020年には『Product Thinking 101』[xxi]が執筆されました。この記事は、多くの意味で、デザイン思考のプラクティス領域で採用される目標を形作る上位集合です。プロダクト思考では、多くの中核的なデザイン思考方式を適用しますが、市場分野のもう少し広いコンテキストでは、市場の大規模なコンシューマ・グループの場合は製品プランニングとして適用します。一方、デザイン思考方式（観念化、人間中心設計など）は通常、個人ユーザーのニーズ、目標、および解決策に非常に直接的に結び付きます。このような点で、デザイン思考はプロダクト思考を広く補足するものです。

デザイン思考とプロダクト思考はどちらも、Clayton Christensen氏の“job to be done”（JTBD）[xxii]の概念に大きく依存しています。JTBD理論は、顧客中心の製品設計に対する包括的なアプローチであり、例として、この概念の本質は次のように言い表されます。「*私たちは製品を購入するとき、基本的に仕事を片付けるためのものを'採用'します。その製品が良い仕事をしたら、次に同じ仕事に遭遇したとき、同じ製品を再び採用します。製品が良い仕事をしなかったら、その製品を'お払い箱'にして、問題を解決するために採用できる別のものを探します*[xxiii]。」製品中心のこのような思考は、データ・プロダクト思考の基本的な側面です。たとえば、‘データ・プロダクト’の設計を目指している場合、データがどのような仕事を達成し、データがその仕事でどの程度うまく機能したかを継続的に評価する必要があります。

データ・プロダクトの考え方の中心的枠組みは、データ・サイエンス・コミュニティにその起源があります。データ・サイエンス領域で最初に考え出されたデータ・プロダクトは、もっとも一般的には、“*任意のデータ・サイエンス・アクティビティの出力*”と定義されています。しかしながら、データ・プロダクト思考のより広いコンテキストでは、データ・プロダクトは、“*KPIおよびSLAを使用して管理され、ビジネス成果を直接後押しする任意のデータセット*”と考えることもできます。本書ではこの広義を使用し、製品化の目標を達成するための、KPIとSLAによるガバナンスに重点を置いています。データ・プロダクトの代表的な例は以下のとおりです。

データ・プロダクトの代表的な種類	
分析	• 成果とビジネス決定を後押しするレポートとダッシュボード • データ・サイエンス・リグレッション、包括的レポート、移動中のデータに対するリアルタイムのインサイトが含まれる場合があります
データ・モデル	• データ・ドメイン・オブジェクト（おそらくシリアライズされた、またはUMLダイアグラムとして表されたコンポーネント・モデル） • データ・モデル（リレーショナル、グラフ、オントロジーなど）およびML機能（列/属性など）
アルゴリズム	• 設計ドキュメント、データ・カタログ、RETEエンジン、ETLマッピングなどに取り込まれたビジネス・ルール • 製品化された、またはMLライフサイクル・カタログに取り込まれた機械学習（ML）モデル • バッチETL、ESBドキュメント変換、リアルタイム・ストリーム統合などをはじめとするデータ・パイプライン
データ・サービス/API	• データ・ペイロード（XML、JSON、Avro、Protobuf、専有ツールの形式など、シリアライズされた形式） • API管理：メッセージベースのサブスクリプション向けのPub/SubセマンティックなどをはじめとするREST API • 契約による設計に基づき、APIが（開発者によって）消費される製品であるAPIフレームワーク

重要なのは、上記のすべてのビジネス・アーティファクトが必ずしもデータ・プロダクトと見なされるわけではないということです。データ・プロダクトに欠かせない特性は、‘*…ビジネス成果を後押しする…*’ことであり、すべてのデータ中心のアーティファクトが成果を直接後押しするわけではありません。たとえば、一連のダッシュボードと分析レポートのうち、ほんの一部のみがビジネス決定を後押しするために定期的に繰り返し使用される場合があります。エンタープライズ・データは大量に存在しますが、そうしたデータのほとんどは、本格的なデータ・プロダクトとして管理する必要はないでしょう。

データ・プロダクト・マネージャー

‘データ・プロダクト’という用語と同じように、データ・プロダクト・マネージャーの概念もデータ・サイエンス・コミュニティから生まれました。データ・サイエンスの領域では、ビジネス所有者、エンジニア、およびデータ・サイエンティストの交流を促進できる熟練のメンバーがすぐにチームに必要になりました。他のエンジニアリング組織や製品組織と同じく、データ・サイエンス・チームには、複数の高度なプログラム管理スキルと、製品ライフサイクルを実行し、ビジネス成果を後押しする技能や知識を持つ熟練のメンバーが必要でした。

データ・プロダクト・マネージャーの概念は、幅広い潜在的データ・プロダクトに適用されます。データ・プロダクト・マネージャーは、高度なデータ・リテラシを通常の製品管理スキルセットに取り入れます。一般的な製品マネージャーと同じく、デザイン思考/プロダクト思考の適用、ビジネス事例の構築、戦略の策定、製品ライフサイクルの管理、ロードマップの構築、リリース計画の提示、バックログのコンパイルなどを行うほか、数多くの利害関係者との協働を図ります。加えて、データ・プロダクト・マネージャーは、組織のデータ・プロデューサ、データセットのデータ形式と意味、およびデータ・プロダクトのコンシューマによって下流で消費されるデータに適用できるアルゴリズムや可視化の種類についての技術や知識を備えています。

図10：データ・プロダクト・マネージャーと従来のプロダクト・マネージャーの役割の比較

Rapido Labsの言葉を借りれば、「*従来のプロダクト・マネージャーが、ビジネス、エンジニアリング、およびユーザー・エクスペリエンスが交わる地点で働いているとすれば、データ・プロダクト・マネージャーは、データ、エンジニアリング、およびビジネスを結び付ける場所にいます*[xxiv]。」

データ・プロダクトにおけるKPIとSLA

あらゆる製品と同様、データ・プロダクトは、完全なライフサイクルにおいて計画および管理されなければなりません。データ・プロダクトは、キー・パフォーマンス・インディケータ（KPI）で評価され、品質保証契約（SLA）に遵守する必要があります。データ・プロダクトの各事例では、本来、独自の一意の属性が管理されますが、データ・プロダクト属性の一例として、以下が挙げられます。

データ・プロダクトの属性	
パッケージ化、ドキュメント	• どのように消費されるか。完全に文書化されているか
目的と価値	• 目的の用途は適切に定義されているか • 価値は明確か、暗黙的か • 時間とともに価値が下がるか。どの程度の割合で下がるか
品質、整合性、サービス・レベル	• KPIとSLAは明確か • 異なるデータ・コンシューマの体験は類似しているか • 回数が異なるコンシューマ（毎日、週に一度、月に一度など）の体験は同じか
起源、ライフサイクル、ガバナンス	• ゆりかごから墓場までの反復可能なライフサイクル・ポリシーで完全に管理されているか • データの起源を追跡できるか • 抽出されたデータを説明できるか（アルゴリズムの出力など）

データ・メッシュの概念は、データ・プロダクト思考と深く絡み合うようになりました。データ・プロダクトの起源となる考え方の多くは、データ・サイエンス・コミュニティから生まれましたが、データ・プロダクトという用語は、分析、データ・サービス、従来型のデータ・モデリング・ドメインに及ぶより多くの潜在的データ・プロダクトを含むまでに拡張されました。自社のデータ・ドメインや、そのデータと効果的に連携するために必要なデータ形式とデータ・ツールに深く精通した人材として、新しい種類のデータ中心プロダクト・マネージャーがますます必要とされています。このようなデータ・プロダクト・マネージャーは、高価値のデータをキュレーションし、パッケージ化して、社内ユーザーや社外ユーザーが幅広く消費できるようにするという独自のタスクを担います。同様に、自社のデータの価値を評価し、追跡する組織では、データの完全なエンド・ツー・エンドのライフサイクル全体でKPIとSLAを管理するのはデータ・プロダクト・マネージャーです。

運用データ・ストアと分析データ・ストアの連携

コンピューティングの時代が始まった当初から、私たちはさまざまなワークロードのニーズや特性に応じて、デジタル化されたデータを異なる方法で整理しています。オンライン・トランザクション処理（OLTP）ツールは一般的に、業務系データを扱うツール、すなわち業務を遂行し、あらゆる種類のトランザクション（挿入、更新、削除など）を処理するアプリケーションでした。オンライン分析処理（OLAP）は、大量の各種データを結合する複雑な読取り操作のために最適化され、大規模なデータセットで実行されます。OLAPシステムは多次元データ・モデルと非常に緊密に結び付いていますが、本書では、分析データ・ストアをより一般的に捉え、読取り用に最適化された分析をサポートするために使用される多種多様なデータ構造を含めています。

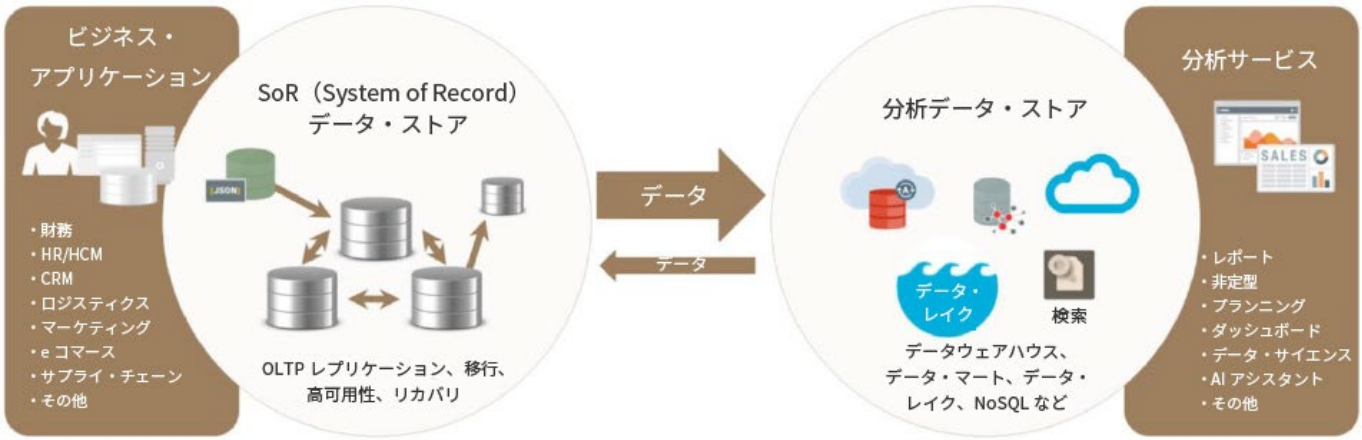


図11：従来から分離されているOLTPデータ・ドメインとOLAPデータ・ドメイン

上述のように、OLTPデータ・ドメインとOLAPデータ・ドメインはこれまでずっと分離されており、両者は単に疎結合されている状態でした。極めて近代的な企業の大半では、OLTPシステムを実行する運用チームは通常、分析システムを設計し、運用するエンジニアとは全く別のIT組織に属します。
たいていの場合、これらのデータ・ドメインを結合する唯一の方法は、一連のETLジョブとストアド・プロシージャです。

OLTPワークロードとOLAPワークロードを全く同じデータ・ストアで一緒に実行できると主張する特殊な少数のベンダーは別として、世界のITシステムの大部分は、依然としてOLTPワークロードとOLAPワークロードを最適化された別個のデータ・ストアに分離しています。同様に、一部の分散化されたソフトウェア・アプリケーションは、‘最終的な整合性’データ・モデルをサポートするように設計されていますが、実際の業務でそのようなアプリケーションが使用されることは現在も非常にまれです。ワークロードに合わせて最適化された、厳格な整合性のあるデータ・ストアは、依然として現代のソフトウェア・アプリケーションの基礎です。Oracle Databaseなどのコンバージド・データベース・システムは、最適化されたポリグロット・データ・ストアを、同じDBMS内で別のプラガブル・データ・ストア・インスタンスとして実行できます。多くの企業はいまだに、データ間に物理的なネットワーク上の分離を設けています。そのため、データ統合を使用してOLTPデータとOLAPデータの連携を実現する必要があります。

古い実証済みのデータ統合方法では、ETLハブアンドスポーク・エンジンが使用されていました。しかしながら、この種のバッチ指向型（スケジューラ駆動型）データ・パイプラインは、OLTPデータセットとOLAPデータセットの分離を悪化させているにすぎません。このようにデータが深く分離されると、データ・ドメインを連携し、データ資産全体で価値あるデータ・ガバナンスを提供することは極めて複雑になります。

動的なデータ・ファブリックと信頼性のあるデータ・メッシュでは、OLTPデータ・ドメインと分析データ・ドメイン間のリアルタイム・ストリーム統合という代替の道筋が示されます。これは、想像するよりもシンプルな可能性があります。

数十年の間、論理レプリケーションが分散OLTPトランザクションの解決策として活用されてきました。Oracle GoldenGateなどのツールは、高価値のデータのACIDプロパティを維持しながら、データ・トランザクションをWide Area Network（WAN）経由でレプリケーションできます。実際、Oracle GoldenGateの主要なユースケースは、オラクルのMaximum Availability Architecture（MAA）の‘プラチナ層’であり、高可用性（HA）とディザスタ・リカバリ（DR）を維持します[xxv]。そのため、信頼性のあるリアルタイム・トランザクションのための基本機能は、すでにOLTPスタックに適切に構築されています。本書の後半では、データ・メッシュ・テクノロジーを、分析ドメインに頻繁に存在するポリグロット・データセットに適用する方法について説明します。

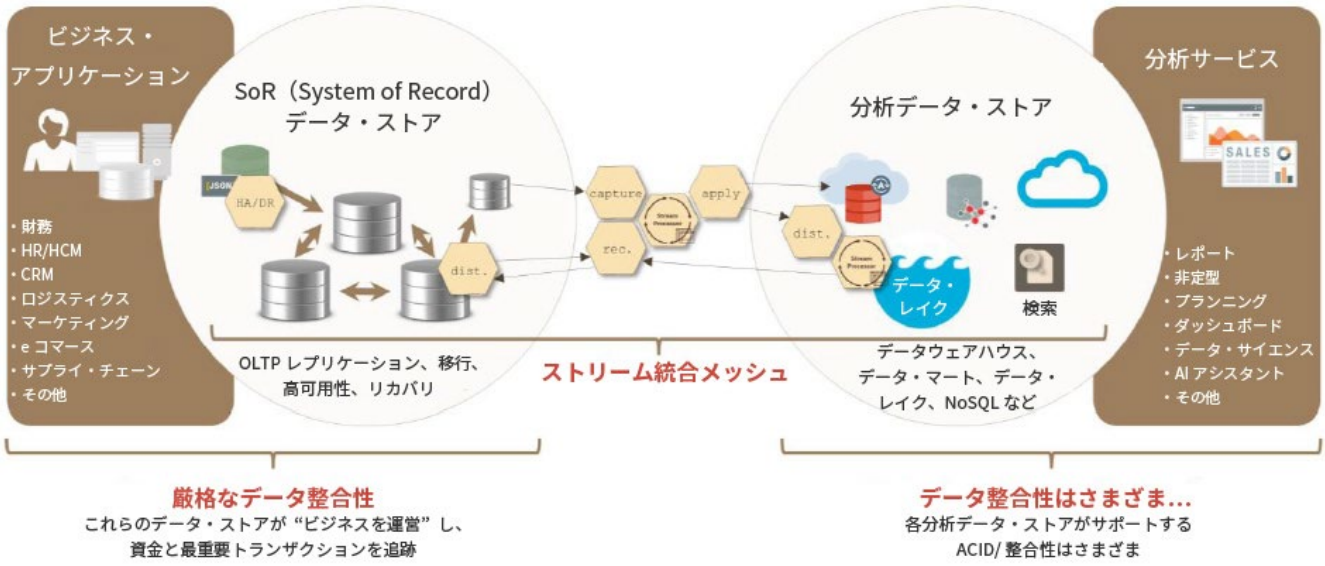


図12：OLTPデータ・ドメインとOLAPデータ・ドメインのセマンティックにまたがることが可能な統合メッシュ

上記のように、ストリーム統合メッシュを適用して、OLTPドメインと分析ドメインを連携させることができます。人体の循環機能のように、ストリーム統合メッシュは移動中のデータを、そのデータを必要とするOLTPシステムとOLAPシステムに継続的に送り出すことができます。これは一方向の流れではありません。データ・イベントは多方向に継続的に流れることができます。データ・フローは、「原理3：信頼性のあるポリグロット・データ」で説明した‘信頼性の特性’を維持しながら、ポリグロット・データ・ストリーム（リレーショナルまたはドキュメント・ペイロード）をサポートできます。この信頼性の特性は、OLTPトランザクションの意味が正確に分析データ・ストアに伝えられるという確信を持ち続けるために絶対不可欠です。

要約すると、リアルタイム・ストリーム統合が、（スケジューラ駆動型のバッチ処理とは対照的な）データ・ファブリックの‘動的’な性質を作り上げ、整合性の高いACIDトランザクションをサポートすることが、（保証のないデータ移動とは対照的な）データ・メッシュの‘信頼性’を作り上げます。このリアルタイム・メッシュを使用すると、OLTPデータ・ストアと分析データ・ストア間でデータを連携し、同期させることができます。

エンタープライズ・データ台帳

本書の前半の「原理2：エンタープライズ・データ台帳」では、Jay Kreps氏とMartin Kleppmann氏が、このエンタープライズ・データ台帳の概念を推進する中心的な2人であることを述べました。この概念が主に、ストリーム・データを取り巻くデータ・アーキテクチャ革命となりました。エンタープライズ・データに関するこの新しい思考法の中心となるのは、ほぼすべてのエンタープライズ・データ・プラットフォームをログを通して把握でき、リアルタイム・イベント・ログがビジネス・データを理解し、扱う上でのまさに中心であるという概念です。

エンタープライズ・データの意味を理解するにはどうしたらいいかを考えてください。過去50年にわたるコンピューティングの歴史のほとんどの期間で、コンピュータ・トランザクションの意味の考え方として二派が存在していました。（メモリのシリアライズにおける）アプリケーション・ビジネス・オブジェクトが信頼できる情報源であるという考えと、信頼できる最終的な情報源を保持するのはデータ・ストレージ・レイヤー（物理ストレージ）であるという考えです。しかしながら、この新しいエンタープライズ・データ台帳の概念には、時間的な要素が加わっており、3つ目の考え方が提示されています。エンタープライズ・データのセマンティックを本当に理解するには、これらのシステムのイベント・ログのシーケンスを見る必要があるという考え方です。

たとえとして、チェスの対局を考えてください。対局を総体的に理解するには、どのような情報が必要ですか。単にチェス盤を見るだけでは、その時点の駒の位置を把握できるにすぎません。同様に、2人の対局者間の会話に耳を傾けるだけでは、チェスの対局は理解できません。実際の対局のストーリー全体を把握するには、位置、対局者の会話、および対局者の推論を理解し、対局が継続している間の駒の完全な移動シーケンスを考慮する必要があります。

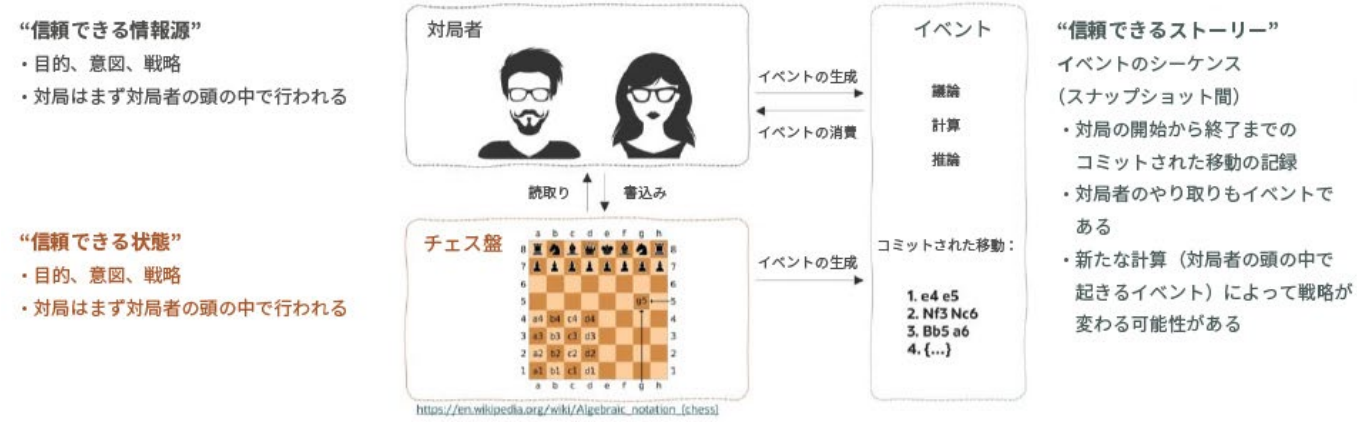


図13：たとえとしてのチェスの対局を理解する

同様に、移動中のエンタープライズ・データの意味を理解するには、（i）アプリケーション・ソフトウェアのログ・イベントを通して信頼できるアプリケーション・ソースを理解し、（ii）データの状態変化に伴うデータ・ストア・ログ・イベントを理解し、（iii）状況に応じてあらゆる時点のデータを理解できるよう、これらのログ・イベントの履歴を保持する必要があります。これらが適切に行われれば、このアプローチによって、動的に変化するデータのストーリーを理解できます。データの動的なストーリーは、意味を理解するために必要なコンテキストを提供してくれるものです。動的なストーリーがないと、特定の時点のスナップショットでレコードとスキーマを把握できるにすぎません。

図14：エンタープライズ・システムの信頼できるストーリーとしてのエンタープライズ・データ台帳

ビジネス・アプリケーションは、人間のユーザーのためのビジネス・ロジックとプライマリ・インタフェースを統合しているため、依然として信頼できる最終的なデータソースです。基盤となる永続データ・ストアは、一般的に、データが高可用性（HA）やディザスタ・リカバリ（DR）の際に使用される場合のデータの位置であるため、信頼できる最終的な状態です。しかしながら、この完全かつ包括的なコンテキストにおいてデータの意味を十分に理解するには、データが伝えているストーリーにアクセスする必要があり、そしてこのストーリーの起源はログのストーリーです。この信頼できるストーリーは、デバイス、アプリケーション、データ・ストアなどからまとめて収集されたイベントのシーケンスです。

慣例的に最適な台帳テクノロジー

エンタープライズ・データ台帳に最適な単一のテクノロジーはありません。前述のように、イベント台帳の概念では個々のユースケースは多種多様であり、マイクロサービスや分析用ログ収集のためのイベント・ソーシングや、汎用的なデータ・サービスとしてのイベント・ソーシングなどに重点を置いている場合もあります。したがって、特定のユースケースに対する具体的なテクノロジーの選択は、ビジネス・ニーズに合わせて行う必要があります。たとえば、イベント台帳は次のようなテクノロジーを基に作成することができます。

- マイクロサービス・イベント・ストア – CQRSなどの一般的なマイクロサービス・パターンと連携するネイティブ機能を搭載
- 時系列データベース – 一般的にIoTデバイスから行われる大量の書込み向けに最適化
- メッセージ・キューまたはサービス・バス – ビジネス・プロセス・トランザクションと構造化データ・ペイロード向けの立証済みキュー
- イベント・ストリーム・プラットフォーム – Apache Kafka、Pulsar、各種の専有クラウド・メッセージ・サービスなど
- データ・レプリケーション・ツール – 最大限のデータ整合性を実現できるよう、データベース・イベント・ログとネイティブに統合
- ブロックチェーン – 不変性と透過性が必須の複数パーティのイベント台帳で特に有用

一般用途のエンタープライズ規模のソリューションでは、複数のテクノロジーを組み合わせることで、最適なアプローチを提供できる場合があります。たとえば、（厳格なデータ整合性とデータ・リカバリ能力を実現する）Oracle GoldenGateなどのレプリケーション・ツールと、（水平スケール・アウトと豊富なポリグロット・データをサポートする）Apache Kafkaなどのオープンソース・フレームワークの組み合わせは、大企業のIT組織が採用する一般的なアプローチです。さまざまな台帳を使用する企業では、データ・カタログやスキーマ・レジストリを使用して、異なる台帳やデータ・ゾーンにまたがるデータ・ドメインを管理する場合があります。

エンタープライズ・データ台帳は魔法のアイデアではありませんが、従来の統合技術を使用して可能であったことをはるかに超えるような意義深い新しい可能性を実現します。まず、非常に細かいレベル（ナノ秒など）で時間を遡ることができる機能は、古いアーキテクチャを使用した場合は不可能でした。次に、ハイエンドのデータ・サイエンスの取組みに絶対不可欠なデータの忠実性が、これまでにないほど向上しました（ユーザー・イベントやイベントごとの表示や操作など）。最後に、広範に分散されたアーキテクチャでリアルタイムに（ミリ秒の待機時間で）ポリグロット・イベント（非構造化ログと構造化ログ）を相互に関連付けることができる機能は、極めて画期的です。

データ・モノリスの分解

複雑な考え方を取り入れ、コンポーネントのパーツに解体するプロセスとしての機能的な分解は、数学に起源があります[xxvi]。同様に、ソフトウェアのレルムでは、分解とは、複雑な実世界のドメインを取り入れ、その概念を、コンピュータ・システムでモデル化できる単純なパーツに分解するアクティビティです[xxvii]。新しいアプリケーションでは、分解は通常、プロジェクトの設計フェーズで行われます。オブジェクト指向の分解技法は、幅広く理解されており、最新のプログラミング言語の基盤です[xxviii]。

モノリスの分解のコンテキストでは、この考え方は、レガシー・システム（通常はモノリシックなソフトウェア・アーキテクチャ）をより小規模なモジュール型パーツにリファクタリングする方法として、マイクロサービス・プログラミング・コミュニティから発生しています。レガシーなモノリスをリファクタリングする方法については数多く文書化されていますが、このトピックは、いまだにサイエンスというよりもアートと見なされています。Chris Richardson氏はマイクロサービス・コミュニティのリーダーとして認識されており、2019年のOracle Codeイベントでの彼のスピーチ[xxix]は、モノリシックなアプリケーションの機能的分解方法についての規範的なリソースと考えられています。

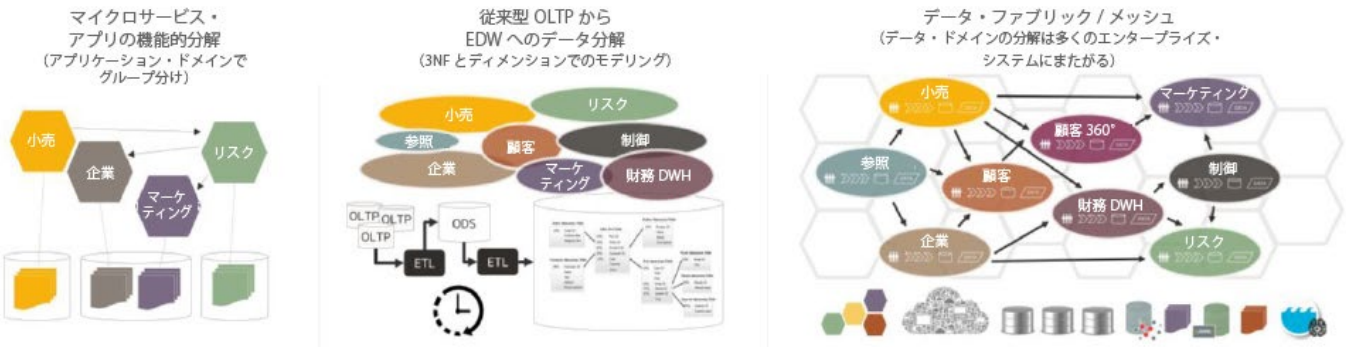


図15：さまざまな分解方式の比較

データ・メッシュまたはデータ・ファブリックの一部としての分解プロセスは、データ・ドメインにより重点を置いています。データ・ドメインの機能的分解には、非常に長い歴史があります。データを第3正規形（3NF）[xxx]にモデル化する1970年代の方法と技術は、現在も間違いなく、データベース内でのデータ・モデリングに対するもっとも一般的なアプローチです。ビジネス・ドメインを3NFモデルに分解する方法は数多く存在します。OLTPのユースケースに重点を置いている3NFでは、付随して生じるOLAPデータ・モデルは、通常は多次元（スター・スキーマ）モデルとしてモデル化されます。エンタープライズ・データウェアハウスが台頭した1980年代を振り返ると、データ・ドメインの機能的分解を読取り用に最適化された分析データに適用するアプローチが無数に存在します（Kimball、Data Vaultなど）。

データ・ファブリックとデータ・メッシュの分解は、範囲が根本的に異なります。データ・ファブリックとデータ・メッシュの本質は、さまざまなシステムによって生成され、さまざまなデータ・プロダクトとしてパッケージ化され、さまざまなコンシューマによって消費される分散化データだからです。ファブリックやメッシュの本質は、具体化されたエンティティとしてモデル化される単一のアプリケーションやデータ・ストアではありません。言い換えれば、メッシュやファブリックを介して流れるデータ・エンティティは、下層にある多数のシステムにまたがることができ、データを具体化するペイロードや永続ストアに応じて、複数の形態、構文、形状を取ることができます。

メッシュやファブリックでの機能的分解が異なるもう一つの領域は、モデリングの動作です。アプリケーション・コンポーネントでは、動作はAPIの背後に存在し、状態遷移ダイアグラムでモデル化できます。OLTPデータベースやデータウェアハウスでは、動作は時折、データベース内に常駐するストアド・プロシージャとしてモデル化されます。ファブリックまたはメッシュの動作は、通常はデータ・パイプラインによって具現化されます。データ・パイプラインは、さまざまなテクノロジーを使用して実装されますが、イベントを取得し、データを移動して変換し、その後データをイベント・ストアにロードするか、シンク（ターゲット・テクノロジー）に直接ロードするといった具体的なアクションに論理的に重点を置いています。

「原理2：エンタープライズ・データ台帳」で述べたように、企業はさまざまなイベント台帳を使用する可能性があります。データ・オブジェクトはさまざまなグローバル・システムにまたがることがあるため、関連のイベントとペイロードは、複数のトピック、キュー、またはレプリケーション・パスに存在する可能性があります。そのため、これらのイベント台帳（およびパーティション、レジストリ、トピック、キューといった台帳の要素）も、ペイロードや、データ・モデル、パイプラインとともに分解され、管理される必要があります。

実用的な視点から見ると、分解とリファクタリングが行われる程度はさまざまです。個々の状況によっては、論理的な分解（つまり、エンタープライズ・データ・カタログ・ツールを使用）のみに重点を置くのが理にかなっている場合もあります。論理的な分解により、既存のアーティファクトを適切に管理されるドメイン・タグ、分類構造、またはオントロジーの各カテゴリに整理することで得られる利点が提供されます。論理的な分解ではなく、より徹底的なリファクタリングを行い、アプリケーション、ETLツール、データウェアハウスといったレガシーなモノリスを解体して書き換えることもできます。このより徹底的なリファクタリングでは、ITの俊敏性の向上、データ・サービスの待機時間の短縮、データ・プロダクトとインサイトの適時性の向上をはじめとするアグレッシブなビジネス変革目標を達成できる可能性があります。

データ・ドメインとデータ・ゾーン

データ・ドメインは、明示的なビジネス問題の領域によって定義される論理的なカテゴリであり、これらのデータ・ドメイン・カテゴリは一般的に、一緒に属するデータ・アーティファクトをグループ化するために使用されます。一方、データ・ゾーンは通常、データの特定のライフサイクル段階に応じてデータが収集され、処理される物理的な場所を指します。

図16：データ・ドメインとデータ・ゾーン

データ・ドメインは、複数のデータ・ゾーンにまたがる場合があります。たとえば、小売業者には、在庫や購買といった複数のデータ・ドメインがあるかもしれませんが、これらのドメインのデータ・エンティティは、一時データ、生データ、準備済みデータ、およびマスター・データ用の複数の処理ゾーンにまたがっている場合があります。各処理ゾーンは、通常はITインフラストラクチャの異なる物理ロケーションに対応付けられますが、データ・ライフサイクルの異なるフェーズや品質レベルと関連付けられている場合があります。データ・ドメイン（在庫や購買など）は、同じ種類のデータを整理してグループ化するための論理的なグループであり、異なる方法（Java集計、JSON/Avroペイロード、Parquet、分析用のリレーショナル形式など）でデータの形式がシリアライズされる場合にも使用されます。データ・ゾーン全体で構文とシリアライズがすべて変更されても、データの意味は引き続きドメインに属します。

ドメイン駆動設計（DDD）の概念

マイクロサービス・コミュニティ内では、ドメイン駆動設計（DDD）に分類される話題はほとんどありません。DDDは、2003年にEric Evan氏の書籍、『Domain Driven Design』[xxxi]で、オブジェクト指向アプリケーション開発向けのパターンの集合として最初に考案されました。その後、2013年にVaughn Vernon氏の書籍、『Implementing Domain Driven Design』[xxxii]によって、DDDの概念は何年にもおよぶ実際の実装に基づき更新されました。現在もなお、マイクロサービスの思想的リーダーたちは、DDDのプラクティスについて定期的に議論しており、多数のDDDパターンを反復可能な規範的方法で実装することに、多くの開発者は戸惑っています。

DDDは、依然としてアプリケーション開発、オブジェクト指向、およびマイクロサービス開発者の各コミュニティでは明らかに注目されています。多くの有用なアイデアと方法がDDD手法で体系化されていますが、DDDに関連する固有のパターンと多くのワークショップの中心となるのは、アプリケーション開発です。一方、本書の範囲では、本質的にさまざまなエンタープライズ・アプリケーションやデータ・ストアにまたがるエンタープライズ・データ・ドメインを重視しています。そのため本書では、‘データ・ドメイン’という用語はDDDを指しません。

ビッグ・データ・コミュニティにおいて、データ・ゾーンの概念はありふれたものになりました。‘ビッグ・データ’以前は、Kimball、Inmon、およびData Vaultデータウェアハウス・アーキテクチャでも、データ・ゾーンの使用が定められていました。大量のデータを保持するあらゆるデータ・ストア内で、データセットを、データのライフサイクル・フェーズ、品質、またはセキュリティ制御によって区別されるさまざまな物理ゾーンにパーティション化することは非常に有益です。データ・レイクやデータウェアハウスと異なり、動的なデータ・ファブリックまたは信頼性のあるデータ・メッシュでは、データは分散化されます。そのため、データ・ゾーンの概念は、異なるストレージ・エンジン、処理テクノロジー、または物理クラウド・インフラストラクチャに関連付けられる場合があります。

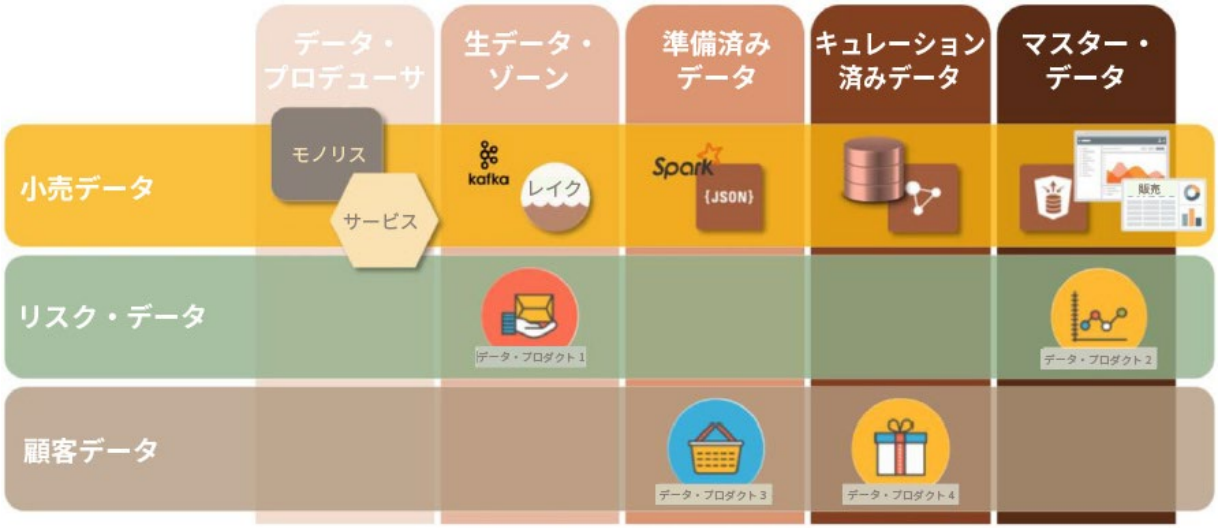


図17：あらゆるゾーンに存在し得るデータ・ドメイン、データ・ゾーン、テクノロジー、データ・プロダクトの例

個々の企業やプロジェクトには、必要とされる種類のゾーンに対して固有の要件があります。一部の組織はセキュリティ制御を最優先事項とし、一方、他の組織はデータ・プロダクトを迅速に開発し、繰り返し改善できることを優先させるかもしれません。

データ・ゾーン	説明	データ・プロダクトの例	テクノロジーの例
一時ゾーン	ステージングされ、取込みの準備が整ったデータは、取込み後に削除される場合があります	非該当、未管理	ファイル・システム、プロデューサ提供のキュー
生データ・ゾーン	アプリケーションやデータ・ストア（プロデューサなど）から直接出力されたイベントおよびデータの収集ポイント	生データのフィードは通常、データ・プロダクトのガバナンス制御を生のフィードで直接行うデータ・サイエンティストが非常に関心を持っています	イベント向けのApache KafkaやJMS、ファイル/大量データ向けのオブジェクト・ストレージやHDFS
準備済みデータ	このゾーンでは、基本品質フィルタがフィルタに適用され、適合する構文/構造体にデータがフォーマットされます	ITデータ・パイプライン（ETL、ストリーム統合など）のデータ・プロダクト（ソース・カスタマー表の補完的な全変更データなど）	セルフサービスCDC、レプリケーション・ツール、ストリーム統合
キュレーション済みデータ	キュレーションとは、通常は、データ・スチュワードやデータ・アナリストなどによる人的アクティビティによって作成されたモデルを指します	通常は業務系データ・ストア（ODS）、あるいはイベント駆動型コンテキストでは、管理トピック一式とAvro/JSONペイロード	データ・カタログ、スキーマ・レジストリ、および変換用パイプライン
マスター・データ	従来のMDMシステムと同様、マスター・データは重要なデータ・レコード（顧客レコード、製品レコードなど）の‘ゴールド・コピー’です	マスター・データ・レコード自体が、完全にパッケージ化され、独自のライフサイクルを持つ個別エンティティとして管理されるデータ・プロダクトです	リレーショナル・データベース、グラフ・エンジン、データ・パイプライン、ビジネス・ルール
セキュア・ゾーン	このゾーンには下位区分があります。ゾーンには、DMZ、信頼できる領域、または制限された領域が含まれる場合があります	データ・プロダクトには、非暗号化財務トランザクション、預金残高、ハッシュ化されたパスワードなどが含まれる可能性があります	Kafka、Hadoop、DBなどの該当ツールのセキュリティ・フレームワーク

各組織は、ニーズに応じて異なる方法でゾーンを管理することを選択する場合がありますが、あらゆるデータ・ファブリックまたはデータ・メッシュの一般的なプラクティスでは、データ・プロダクトはゾーンで適切に管理されるエンティティと定義されます。データ・プロダクトはどのようなゾーンにも存在できます。データ・コンシューマは、ライフサイクルのさまざまな段階で、データの消費に関心がある可能性があります。たとえば、ユーザー動作のリグレッション分析を実行しているデータ・サイエンティストは、データ・プロデューサから直接生イベントを確認したいでしょう。一方、経営幹部用のダッシュボードとレポートは、通常は信頼性が極めて高いキュレーション済みデータ/マスター・データを基に構築されています。

つまり、データ・アーキテクトは、動的なデータ・ファブリックまたは信頼性のあるデータ・メッシュ内でデータ・ドメインとデータ・ゾーンを定義する作業に精通している必要があります。ただし、分散化されたデータ資産でこの分解を検討し、実行する範囲は著しく変化します。ビジネス・ドメインでは、直感的なソリューションで、データ・カタログを使用して1つまたは複数のドメイン内のデータ資産を管理します。分散アーキテクチャのデータ・ゾーンでは、データ・プロデューサやデータ・コンシューマ、必要とされるKPIやSLAに応じて、さまざまなツールの選択肢が存在する場合があります。

何がメッシュをメッシュにするか

動的なデータ・ファブリックとは、データ・ファブリックを動的にする待機時間の短いストリームと言い表されます。本書で信頼性のあるデータ・メッシュの考え方を紹介した際に、ACIDデータを正しく処理できることが、データ・メッシュの信頼性を作り上げると説明しました。それでは、何がデータ・メッシュをメッシュにするのでしょうか。

現代の日常では、‘メッシュ’は次のように多くの領域で、ありきたりな言葉として使用されています。

- **WiFiメッシュ** – 家庭およびビジネス用のインターネット・テクノロジーでは、単一のルーターから、分散化されたWiFiアクセス・ポイント群に移行することで、インターネットの速度と信頼性が向上します
- **スマート・ホーム・メッシュ** – Z-Wave、Zigbeeなどのコネクテッド・スマート・ホーム・プロトコルでは、家庭で使用するIoTデバイスは、信号を再現するために相互に接続することも、メインのハブ/ルーターに接続することもできます
- **5Gメッシュ** – 通信事業者が世界中で5Gネットワークをデプロイするなか、安定した接続と高速通信を維持するための局所的なスモール・セルのデプロイメントが重要な属性となります

- **サービス・メッシュ** – ソフトウェア・コンポーネントの分散メッシュの一部として、コンテナ、Kubernetes、OpenShiftなどで実行できるアプリケーションの設計は、ソフトウェア・アプリケーション開発者の間で主流になりました

メッシュ・コントローラ/コントロール・プレーン

メッシュをメッシュにする決定的な特性はいくつかあります。1つ目はネットワーク概念です。メッシュの中心的な考え方は、何といっても接続されたモノのネットワークです。2つ目は、メッシュが分散化されていることです。あらゆるメッシュ・ネットワークと非メッシュ・ネットワークの第一の相違点は、メッシュが集中型ハブを経由した接続に依存していない点です。メッシュ・ネットワークでは、ノードとデバイスを相互に直接接続でき、通信を毎回ハブを経由させることなくルーティングできます。3つ目は、すべてのメッシュが一種のコントロール・プレーンを持つことです。すべてのメッシュ・ネットワークでは、ネットワークのどこかにコントローラまたはハブがありますが、機能的な通信が各ネットワーク・ホップでコントローラやハブを経由しなければいけないというわけではありません。名称がハブか、コントローラやコントロール・プレーンかにかかわらず、すべてのメッシュには、メッシュ・ノードの管理に役立つこの高度な監視プロセスがあります。つまり、管理されていないメッシュはそもそもメッシュではありません。

サイドカー・プロキシ・パターン

1994年に初版が発行された“Gang of Four”の書籍、『Design Patterns:Elements of Reusable Object-Oriented Software』[xxxiii]により、プロキシ・パターンなどのオブジェクト指向パターンが普及しました。プロキシを使用した基本設計原理では、通常はプロキシ・オブジェクト内で適用される追加ロジックとともに、インタフェースを他のコード・オブジェクトに提供します。分散システムでプロキシを使用した初期の事例では、プロキシは“リモート・プロキシ”として、通信の複雑性をリモート・オブジェクトの背後に隠すローカル・インタフェースとともに使用されました。

2016～2017年頃、サイドカー・プロキシという用語（サイドキック・パターンと呼ばれることもあります）が、サービス・メッシュの動向（Istio、Linkeredなど）と決定的に結び付き、その後、Kubernetes、OpenShift、Kongといった現代のサービス・メッシュの中心的な設計機能になりました。これは、すべての主要プロバイダのあらゆるパブリック・クラウド・アーキテクチャで使用される決定的なパターンでもあります。

サイドカーは、分散アプリケーションと結合されたインタフェース・プロキシ・オブジェクトです。可観測性/テレメトリ、ロギング、構成、ネットワーク・ルーティングといった一般的な機能群（これらの機能は実装によって異なる場合があります）は、サイドカーによって管理されるため、基盤となるアプリケーション・ソフトウェアで管理する必要はありません。サイドカーは、関連するすべてのサイドカーで操作を実行できる共有コントローラ（コントロール・プレーン）によって管理されるため、膨大な数の‘データ・プレーン・サービス’をサイドカー・プロキシによって効果的に制御できます。

サイドカー・プロキシは、最新のパブリック・クラウド・インフラストラクチャが機能する仕組みや、IT組織が自社のオンプレミス・インフラストラクチャ内でアプリケーションを“サービスとして”運用する方法の基盤を提供する基本パターンです。

データ・メッシュには常にコントローラがありますが、データ・ファブリックにはない場合があります。単一のメッシュは1つの共有コントローラに接続され、そのコントローラはメッシュ・ノードの運用ライフサイクルを管理します。さまざまなメッシュが相互にやり取りできますが、メッシュのインスタンスが管理する範囲は、特定のコントローラによって管理されるノード/ポッド/サービス一式によって定義されます。たとえば、WiFiメッシュとスマート・ホーム・メッシュは、同一家屋内で実行でき、相互にやり取りできますが、各メッシュは結局のところ独自の物理コントローラを持ち、他のメッシュとは別に管理されます。一方、統一されたデータ・ファブリックの概念の中には、考え方が概して論理的なものもあります。すべてのデータ・ファブリックが必ずしも共有のシステム・コントローラを持つわけではありません。たとえば、データ・ファブリックの中には、堅牢なデータ・ガバナンス方式（プロセス・コントロール）および共有系統のメタデータ・カタログの使用と、さまざまなデータ・ドメインのビューによってのみ統一されるものもあります。本書の目的におけるデータ・ファブリックの完全な定義は、「企業のニーズを満たす動的なデータ・ファブリック」のセクションに記載されています。

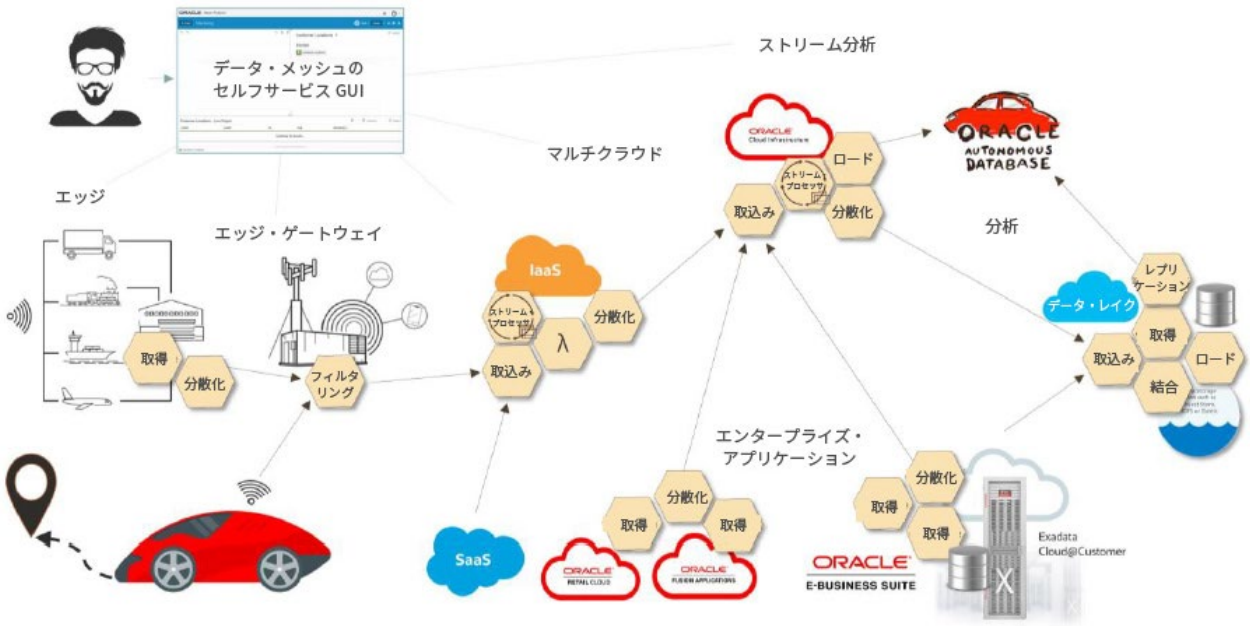


図18：データ・メッシュは、共有コントローラを持つが、分散されたさまざまな物理ロケーションで実行可能

データ統合の従来のハブアンドスポーク方式（ETL、データ・フェデレーションなど）と比較すると、データ・メッシュのトポロジは独自です。あらゆるメッシュと同様、データ・メッシュは（1）1つのネットワーク概念であり、（2）分散化されていて、（3）中心的なサービスでコントロール・プレーン、つまりサイドカー・タイプ・パターンを使用します。接続されたあらゆるインフラストラクチャに物理的に常駐するメッシュ・ノード全体で、リアルタイム・データの取得やルーティングを行うことができます。単一ハブや共有サービス・バスは不要です。大企業では、さまざまなデータ・メッシュのデプロイメントが存在し、それぞれが、独自のコントロール・プレーン・サービスを持つ可能性があります。実際、Oracle GoldenGateを使用すると、データ・メッシュは、Oracle Cloudでホストされるサーバーレス・インフラストラクチャやオンプレミスのインフラストラクチャから制御でき、Oracle Cloud以外のクラウド内で制御することさえも可能です。データ・メッシュは、GoldenGateコントロール・プレーンがどこで実行されているかにかかわらず、イベントを配信したり、他の任意のデプロイメントからイベントを受信したりすることができます。これは、WANに接続されたあらゆるデバイスにエンタープライズ・データ・イベントをシームレスにルーティングできるという、メッシュ・タイプのデプロイメント固有の価値提案です。

おもなユーザー、セルフサービス・ロール

ソフトウェア開発のあらゆる領域と同じく、ローコードとセルフサービス機能への移行は、非テクニカル・ユーザーによるデータ・ファブリックの利用を高める上で不可欠です。大半の業務ユーザーに要求される基本的な職務から、多くのプロセスを必要とするITプロジェクトを排除することは、業務ユーザーとIT部門の双方にとってメリットがあります。データ・ファブリックとデータ・メッシュは、セルフサービス・ユーザー・インタフェースが未来であるという意味では何ら違いはありません。非テクニカル・ユーザーのデータ・アクセスを可能にし、ITスタッフとITプロセスへの包括的な依存を排除することが重要です。

セルフサービス・データの準備は、ソフトウェアにおいて十分に確立されたカテゴリです。主要なあらゆる分析プラットフォームの一部であり、データ統合ツールにおける一般的な機能領域です。しかしながら、セルフサービス・ツールを使用して実現できていることは限られています。高度な機械学習（ML）や、自然言語処理（NLP）、グラフベースのナレッジ・ベース（KB）を組み込んだとしても、セルフサービス・テクノロジーには、データのキュレーションを行い、データの正確な意味（セマンティック）を確定するループで依然として人間の関与が必要です。ML、NLP、およびグラフの幅広い普及は、データ・キュレーションの自動化と向上に役立っていますが、まだ人間に取って代わるものではありません。そのため、動的なデータ・ファブリックと信頼性のあるデータ・メッシュには、依然としてファブリック/メッシュを運用し、管理する人材が必要です。

おもなユーザー	
データ運用/DBA	• 業務系データ・プラットフォーム（OLTP DBエンジン、ドキュメント・ストアなど）の専門家 • アップタイム、可用性、リカバリ可能性についての主要なKPIおよびSLAを担当 • セルフサービス・ツールを使用して、変更されたデータをデータ・エンジニアに提供
データ・エンジニア	• データ・パイプライン（データの移動と変換）の開発者 • セルフサービス・ツール、またはコーディングが必要な低レベル・パイプライン・テクノロジーを使用する場合もある • データ、および本番パイプラインのアップデートとパッチ適用においてDevOpsを一元的に実行
データ・プロダクト・マネージャー	• 業務と（直接または間接的に）連携するデータ・アナリストの一種 • データ・プロダクトの所有者（「データ・プロダクト思考」のセクションを参照） • IT/エンジニアリング部門、業務部門、およびデータの技術専門家の間の調整を図る
インフラストラクチャ管理者	• パブリック・クラウドにおけるクラウド運用、ネットワーク、およびセキュリティ • POD管理とCICDライフサイクルのためのコンテナ・レジストリとサービス・メッシュ • ファブリック/メッシュ・コンポーネントを実行しているオンプレミス・チーム用のハードウェアとネットワーク

セルフサービス・データ・プロダクト

セルフサービスへの移行フェーズでは、非テクニカル・ユーザーが管理できるデータ・プロダクトもありますが、それ以外の種類のデータ・プロダクトは、当然ながらIT組織が引き続き生成とキュレーションを行う必要があります。セルフサービス式のデータ・アクセスは、通常はデータ・カタログで始まります。データ・カタログでは、非テクニカル・ユーザーが、データ資産を仮想の“買い物かご”に入れることによって“データを買う”ことができます。レジでは、セルフサービス・プロセスが、非テクニカル・ユーザーのコンシューマに、データをどのように提供して欲しいかを確認します。たとえば、ユーザーはさまざまな形式（Excel、JSON、Avro、XMLなど）やさまざまなロケーション（データ・マート、データ・レイクなど）を選択できます。セルフサービス・プロセスでは、データはドメインごと、データ・ゾーンごとに参照できます。

リアルタイムのストリーム・データ・メッシュには、多様なデータ・プロダクトが存在する可能性があります。完全にセルフサービスのデータ・プロダクトもあれば、IT部門がキュレーションと準備を行うデータ・プロダクトもあります。たとえば、Oracle GoldenGateでは、分散サービスの概念に、変更されたデータ・イベントを消費するマイクロサービス・ベースのAPIが含まれます。GoldenGateのデータ・フォーマッタは、変更されたデータをXML、JSON、Avroなどの一般的な形式で提供できます。そのため、下流のサブスクライバが常に最新のデータ（またはスキーマ）変更を取り込むことができるAPIが、ストリーム・データ・プロダクトである可能性があります。その他の種類のセルフサービス・データ・プロダクトとして、ビッグ・データ開発者ではなく、非テクニカル・ユーザーによって可視化が作成される、データ・ストリームにおけるデータの可視化/分析が挙げられます。

最後に、セルフサービス・ツールは、データ・ファブリックとデータ・メッシュがフレームワークの外部に存在する場合でも、その利点を享受できることも指摘しておく価値があります。たとえば、セルフサービス・ツールを使用してレポート作成用のローカル・データ・マートを作成した場合、動的なデータ・ファブリックを統合して、IT部門がキュレーションを行った更新データを継続的に送り込むことができます。この方法により、個人デバイスのセルフサービス・ツールが、より広範なエンタープライズ・データ・ファブリックまたは信頼性のあるデータ・メッシュと共存して、その利点を享受できます。

連続変換およびロード（CTL）のデータ・パイプライン

過去50年以上のコンピューティングの歴史において、データ変換のタスクは、スケジューラによって駆動されるバッチ・プロセスとして実行されるか、メッセージごとにイベントとして実行されるかのいずれかでした。バッチ処理は高速ですが、頻繁に実行されません（1日に1回など）。一方、イベント処理は非常に頻繁に実行されますが、通常はバッチ処理よりもはるかに低速です（集計処理において）。2015年頃から、3つ目の方法として新しいアプローチが現れました。この新しいアプローチは、ストリームMPP（大規模パラレル・プロセス）プラットフォームに容易にアクセスできる環境が広がり、（クラウド/サーバーレスとオンプレミス・ハードウェアの双方に関する）相対的なコンピューティング・コストが大幅に削減されたことにより実現しました。イベントごとのリアルタイム・データ変換を専用で行うスーパーコンピュータ規模のクラスタを、初めて比較的経済的に実行できるようになりました。

ETLおよびE-LTのアプローチでは、スケジューラでバッチが実行されるとデータ変換が行われます。一方、CTLのアプローチでは、データ・イベントが発生するとデータが処理されます。コンピュータ的に注意を要するのは、ステートフルなデータの処理です。ETLおよびE-LTのアプローチでは、自然に発生する時間枠（バッチ・ジョブの頻度）があります。ETLジョブが24時間ごとに実行される場合、定義上は一度に24時間分のデータが処理されます。データのマッピングと結合はすべて、スケジューラ頻度という自然に発生する時間枠によってバインドされます。さらに、バッチ・ジョブを実行するエンジンは、通常はバッチで消費できる最大データ量に従ってサイジングされます。

データ変換の種類		頻度
ETL	• 抽出、変換、ロード（ETL） • スケジューラ/cronよって実行され、続いてタスクを実行するETLパイプライン • データは、データ・ストアとは別に中央エンジンで処理されます	バッチ/マクロバッチ
E-LT	• 抽出 – ロード、変換（E-LT） • スケジューラ/cronよって実行され、続いてタスクを実行するETLパイプライン • データはターゲット・データ・ストアにステージングされ、データ・ストア内で変換が行われます	バッチ/マクロバッチ
CTL	• 連続変換およびロード（CTL） • ストリーム・パイプラインは、イベントが到着する24時間365日利用できます • データはストリーム・エンジン内で処理されます • 一部のテクノロジーでは、CTLプロセスを単独でバッチ処理（Apache Flinkなど）およびデータ・サイエンス/SQL処理（Apache Sparkなど）と一緒に配置することを選択できます	連続/イベントごと

一方、CTLパイプラインでは、定義された時間枠内のステートフルなデータを管理する必要があります（CEP（複合イベント処理）エンジンのウィンドウイング機能の概念に類似しています）。このような時間枠は、ビジネス・ロジックの一部として使用される場合のあるデータ結合とデータ集計の変換ロジックを促進するために使用されます。処理中にDAG（有向非巡回グラフ）のインメモリに保持される必要のあるデータ量を計画することで、これらの環境のサイジングを行うことができます。Oracle GoldenGateといったストリーム分析向けのテクノロジーには、データ・キャッシュを管理するため、およびクラスタにまたがるインメモリ・データ・グリッドを活用するためのユーザーが構成可能な設定が含まれます。最新のCTLのアプローチと過去のイベント・プロセッサを分離することが、ステートフルなストリーム・データの効果的な管理につながります。

信頼性のあるCTLデータ処理

本書の冒頭で述べたように、データ統合に（他の統合方法とは異なり）独自のエンタープライズ・ソフトウェア市場セグメントが存在するおもな理由として、整合性のある適切なデータを処理する際の信頼性が保証されていることが挙げられます。データ統合ツール（ETL、レプリケーション、データ・フェデレーションなど）は、リレーショナル・データベース・トランザクションのACID整合性の範囲に従い、その範囲で作業するように一から構築されています。一方、他の統合方法、多くのビッグ・データ・フレームワーク、およびApache Kafkaなどのメッセージ・システムは、そのように構築されていません。

連続変換およびロード（CTL）データ処理は、本質的に、さまざまなペイロード形式（XML、JSON、Avroなど）を処理し、リレーショナル・データベースと分析形式（Parquet、ORCなど）のセマンティックの整合性と正確性を維持するポリグロット・ストリームのアプローチでなければなりません。データの正確性を維持する責任は、パイプライン開発者やデータ・コンシューマの手に全面的に委ねるべきではありません。CTLシステム自体が、信頼性のポリシーを実施しながら、ビジネス・データを処理する必要があります。

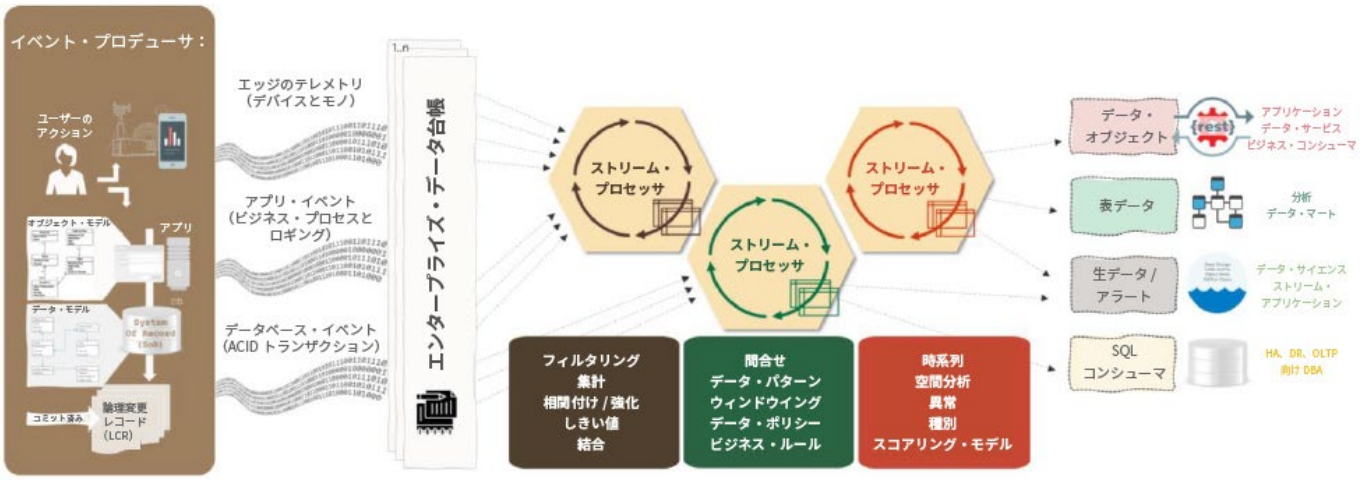


図19：CTLプロセス・コンポーネントが台帳からイベントを受信し、データをリアルタイムで処理し、信頼性のあるデータをコンシューマに出力

企業組織は、古いバッチ処理の環境から最新のストリーム・データの環境へと徐々に移行しているため、データ・プロダクトを繰り返し迅速に改善できることに伴う多大な利点を享受するでしょう。このCTLの分散化されたデータ処理モデルへの移行は、ストリーム・プロセッサを任意の主要パブリック・クラウドまたはオンプレミスで実行しながら、データ・イベントが短い待機時間でその間を継続的に移動できるマルチクラウド・アーキテクチャとも緊密に連携しています。私たちをストリーム・データ・プロダクト・ファクトリの産業化に最終的に導いてくれるのが、より反復可能なこのエンタープライズ規模のバージョンです。

反復可能なデータ・プロダクト・ファクトリ

> *「本当の問題、本当の難題、そして最大の可能性を秘めているのは、マシンを作るマシンを構築することだと気付きました。つまり、工場を建設することです。私は工場を製品のようなものと考えています。」*
> *-Elon Musk氏*

私生活の製品は、毎日さまざまな作成者によって作られます。熟練の職人が一つ一つ手作業で作り上げる製品もあれば、場合によってはロボットが稼働するハイテク工場で大量生産される製品もあります。最新のデータ・ファブリックまたは信頼性のあるデータ・メッシュにより、産業化の手段が提供され、データ・プロダクトの生産手段が自動化されます。データ職人が製品をオーダーメイドで作成して管理することに依存するのではなく、工場内を自動化することで、工場は本当の魔法が起きる場所になります。

以前もHenry Ford氏が注目したように、Elon Musk氏は製品を製造する工場に注目しています。Musk氏は2016年の株主総会で次のように述べました。「*本当の問題、本当の難題、そして最大の可能性を秘めているのは、マシンを作るマシンを構築することだと気付きました。つまり、工場を建設することです。私は工場を製品のようなものと考えています*[xxiv]。」工場は製品です。Elon Musk氏は、この思考をテスラとSpaceXに適用していますが、同じ思考をデータからプロダクトを作成するファクトリに適用する必要があります。

小規模な企業や戦術的なプロジェクトでは、生成されるデータ・プロダクトは、確かにデータ職人（高度な技術を持つ開発者、データ・サイエンティスト、データ・アナリストなど）によってオーダーメイドで作成される可能性があります。しかしながら、複数の大陸で数千人の労働者をサポートする大企業のIT組織では、自動化されたデータ・ファクトリの概念への投資が、現代のデジタル時代における成功と失敗の明暗を分ける可能性があります。

開発者が手作業で作成したシンプルなオーダーメイドのデータ・プロダクトを忘れ去るには、データ・パイプラインを、データ・プロダクトを作成するための生産と自動化の手段と考えることが不可欠です。より具体的に言うと、データとして生まれたデータ・プロダクトは、ソース・プロデューサから取得され、他のデータと結合され、クレンジングされて準備され、最終的にある種のデジタル・ペイロードに書き込まれます。CTL（連続変換およびロード）データ・パイプラインは、Apache FlinkやApache Sparkなどのフレームワークで作業するデータ職人が手作業で作成できますが、そのようなパイプラインは、昔の靴屋が木槌で靴を作るのと同様にスケーラブルではありません。

CTLパイプラインを自動化するツールには以下の属性が必要です。

- **セルフサービス** – データ・アナリストにとって使い勝手が良いノーコードのグラフィカルなユーザー・エクスペリエンスを活用
- **メタデータ駆動型** – 仕様と、コード、ランナー、またはエンジン間の間接レイヤー
- **自律型** – 自己最適化（パフォーマンス/規模）と自己修復（データ・ドリフトなど）が可能
- **サーバーレス** – パブリック・クラウド内、またはIT部門によって管理されるサービス・メッシュ・デプロイメントとして
- **信頼性のあるポリグロット** – ACID整合性の存続とスキーマレス・ペイロードの処理が可能

極めて高度なデータ・プロダクト・ファクトリを望むのは、ビジネス・データ・プロダクトの生産において整合性と反復性を実現するためです。デジタル経済によって変革する業界が増加するにつれて、ビジネスがデータを生産的に使用する機会も毎年増加しています。すべての製品とサービスは、データとデータ・プロダクトの可用性によって根本的に変わります。高度なデータ・プロダクト・ファクトリこそが、ビジネス/IT部門がデータ資産を使用して確実に繰り返し良好な成果を上げることができるようにするものです。

第4次産業革命とスマート・ファクトリ

> 「新しい世界では、大きい魚が小さい魚を食べるのではなく、素早い魚が動きの遅い魚を食べるのです。」
> – Klaus Schwab氏

2015年に世界経済フォーラムの創始者であるKlaus Schwab氏は、‘第四次産業革命’[xxxv]という新語とともに、私たちは大規模な産業化の新たなフェーズに突入しているという中心的考え方を生み出しました。この考え方の基本は、産業化にはこれまで3つの波（1.蒸気動力、2.電化、3.デジタル化）があり、現在は第4フェーズの「サイバーフィジカル」に突入しているというものです。サイバーフィジカル・フェーズの特徴は、デジタル・システム（AI、ML、ニューラル・ネットワークなど）の徹底した自動化の飛躍的前進と、物理的な世界（5Gメッシュ、ロボット、ナノテクノロジー、バイオテクノロジー、モノのインターネットなど）に対する接続性の向上です。インダストリ4.0思想の中心的かつ反復的なテーマにおいて重要なのは、スマート・ファクトリと、俊敏なサプライ・チェーンおよびロジスティクスです。

物理的製品の生産機能がまさに大変革を経験しているなかで、データ・プロダクトを生産する手段も、このインダストリ4.0の時代に突入する必要があります。つまり、データ・プロダクトのスマート・ファクトリを実現する要素として、自動化やロボット工学を取り入れ、AI、ML、グラフ・テクノロジーを使用する必要があります。

グローバル経済のリーダーたちは、世界経済フォーラムのダボス会議[xxxvi]などのグローバル・イベントで、繰り返しスマート・ファクトリの議論を深めていますが、それはなぜでしょうか。簡単に言えば、デジタル・ビジネスの変革がグローバル経済を再構築することが広く認識されているためです。高い俊敏性によって迅速に行動できるビジネスが未来の経済の勝者になるということが、重要なインサイトです。Klaus Schwab氏が述べたように、「*新しい世界では、大きい魚が小さい魚を食べるのではなく、素早い魚が動きの遅い魚を食べるのです。*」これはハイテクの世界では、破壊的な新しいテクノロジーを使いこなす新興企業が定評のある既存企業をたびたび出し抜くイノベーションのジレンマ[xxxvii]として知られています。

俊敏性は、もはや単なる流行語ではありません。多くの業界で、俊敏性は企業の死活問題です。データに依存して収益や戦略を生み出し、データを競争上の優位性として活用するビジネスが生き残るためには、スマート・データ・ファクトリを実装することが近いうちに必須になる可能性があります。多くのグローバル業界では、これまでぜいたくと言われてきたもの（世界クラスのデータ・プロダクト・チームへの投資など）が、近いうちに勝負に参加するための必須条件となるでしょう。

DevOps、CI/CD、DataOpsを使用した生産の俊敏性

ITでは俊敏性は人によって意味が異なります。オンライン・アプリケーションを破壊することなく、ドライブをホットスワップできることや、停止時間を作らずに、本番データ・パイプラインにコードをプロモートすることを意味することもあれば、アプリケーション全体を分散化されたクラウド・コンピューティングに移行することを意味することもあります。実際には、最新のデータ・プロダクト管理では、俊敏性の概念はITの複数レイヤーで適用される必要があります。データ・プロダクトのサプライ・チェーンの俊敏性は、その中でもっとも俊敏性が低いプロセスによって決まります。

これはソフトウェア開発において主流であるため、‘DevOps’という用語が最初に現れたのが2008年だったことが信じがたいほどです[xxxviii]。DevOpsが普及する以前は、（開発環境から本番環境への）コードのプロモーションで、ITの作業が数日かかることや、数週間かかることさえも全く普通でした。編成構造から、インフラストラクチャの操作に使用されるツールに至るまでのあらゆるものに、ユーザーの介入と複雑なコミュニケーション・プロセスが必要でした。2020年頃のクラウド駆動型のソフトウェア開発エコシステムでは、このような古い非効率性は改められ、CI/CD（継続的インテグレーション/継続的デリバリ）に対応するように設計されたDevOps自動化ツールチェーンに置き換えられました。

ソフトウェア開発の領域では、DevOpsはマイクロサービス設計パターンの普及と深く結び付いてきました。マイクロサービスは、オンライン操作を中断させることなく、容易にアプリケーションを絶えず進化させることができるように設計されています。データ管理の領域では、DevOpsプラクティスは根付くのにより時間がかっています。DevOpsに関する調査では、データ管理と速いデリバリ・スケジュールの足並みをそろえることの課題が繰り返し指摘されています[xxxix]。ほとんどのデータ管理システムがいまだにモノリスとして管理されていることがその一因であり、データとスキーマが定義上、アプリケーション・ロジックに密結合されていることにも原因があります。データがデータベースで3NFとしてモデル化されているか、KafkaでJSON/Avroドキュメントとしてモデル化されているかにかかわらず、アプリケーション・コードには、使用される永続ストアが何であろうと、常にある程度のインピーダンス・ミスマッチ[xl]が存在します。

図20：DataOpsとDevOpsは、データ・ファクトリとデータ・メッシュに対する俊敏なアプローチにおいて別個の不可欠な部分である

動的なデータ・ファブリックに対する俊敏なCI/CDのアプローチを作成するには、データ管理で効果的なDevOpsを実施するという課題を乗り越えることが求められます。DevOpsのエンジニアは、継続的な運用のために、データ・ファブリック/データ・メッシュを常にオンラインの状態に保つことを気に掛ける必要があるでしょう。高可用性（HA）、ディザスタ・リカバリ（DR）、パッチ適用、移行（ソフトウェアまたはインフラストラクチャ）、垂直スケーラビリティや水平スケーラビリティなどの運用上の懸念事項にはすべて、責任をもって対処する必要があります。
このようなDevOpsの懸念事項は、DataOpsの制御フローに適用される俊敏なプロセスを支援しますが、両者は互いに独立した関係です。

‘ProdOps’について

2018年後半から、‘ProdOps’という用語がCI/CDについての議論でより頻繁に使用されるようになりました。本書では、この用語はデータ・ファブリックまたはデータ・メッシュの説明には含めていません。ProdOpsの定義がまだ明確でないためです。一部のコミュニティでは、ProdOpsは‘本番運用’を表し、CI/CDとDevOpsに対応した一連のタスクを実行する本番ユニットにおけるIT組織の役割と解釈されます。別のコミュニティでは、ProdOpsは‘製品運用’を表し、販売、マーケティング、製品管理、エンジニアリング、品質保証、サポート・エンジニアを含む製品のライフサイクル全体と関連しています。どちらの定義も、データ・プロダクト思考とデータ・メッシュ機能のコンテキストにおいて興味深いものですが、現在もなお統一した定義が存在しないため、本書の主要トピックからは割愛しています。

DataOpsは、主に、スマート・データ・ファクトリ内のアクティビティの管理と制御に関連します。データ・プロデューサへの接続、データ・パイプラインのライフサイクルの作成とバージョニング、共通のビジネス・ルールとMLアルゴリズムの効果的な再利用の確立、多数のデータ・プロダクト間の従属関係の管理などがその例です。一般的に、DataOpsはツールに依存しませんが、動的なデータ・ファブリックまたは信頼性のあるデータ・メッシュ向けのツールはすべて、特にデータ・ガバナンスにおいて、極めて自動化されたDataOps制御をサポートする必要があります。

データ・メッシュにおけるセキュリティとガバナンス

データ・セキュリティは、CIOにとって永遠に“トップ3”の懸念事項ですが[xli]、AIと分散化されたマルチクラウド・インフラストラクチャの幅広い採用により、あらゆる人にとって常に重要な問題になりました。GDPR、サーベンス・オクスリー法（SOX）、バーゼルIおよびII、HIPAAなど、政治的なデータ規制の厳格化の影響により、データの保護と管理を徹底する方法を制御することが必須になっています。このような制御は、データが物理的に保管される場所や、運営会社に適用される司法権に応じて、各国にさまざまな影響を与える可能性があります。別のアプリケーションや運用ロケーションから取得したデータを1つにまとめたいと考える多国籍組織は、Webやガバナンスにおける複雑なデータ要件にすでに直面しています。このような規制遵守の義務は小さな問題ではありません。組織は非遵守によって、重要相当な金額の罰金を課せられる可能性があります[xlii]。

セキュリティとデータ・ガバナンスは、物理的な永続層における最優先課題です。本書の冒頭で述べたように、動的なデータ・ファブリックまたは信頼性のあるデータ・メッシュは、データ・ストアの代わりになるものではありません。業務系データベース、データ・マート、データウェアハウス、およびデータ・レイクは、エンタープライズ・データ資産内で引き続きデータの永続性を実現します。したがって、これらのデータ・ストアは、今後も機密のデータ・リソースへの制御されたセキュアなアクセスを管理するためのもっとも重要なアクセス・ポイントです。同様に、（特定のデータを参照できるユーザーを制限する）区分化されたデータ・アクセス・ポリシーは、基本的にデータ・ストア層の一部として扱われる必要があります。保存中のデータの暗号制御と不明瞭化はデータ・ストアの機能であり、ファブリックやメッシュのどのような概念によっても置き換えられません。

分散化されたマルチクラウド・アーキテクチャでは、データ・ストアは、サード・パーティ・ベンダーの管理の下で、（場合によっては、政治上/規制上異なる境界内の）さまざまなクラウドに物理的に常駐する場合があります。フルマネージド型のOLTPデータベース・エンジン、データウェアハウス・データベース、またはオブジェクトストレージ・ベースのクラウド・データ・レイクには、データの所有者ではなく、クラウド・ベンダーによって所有および管理される物理的なストレージと他のインフラストラクチャがあります。保存中のデータと使用中のデータを不正アクセスから保護するために、各クラウド・ベンダーやデータ・ストア・テクノロジーによって、さまざまな契約ポリシーや技術的防御措置が導入されます。マルチクラウド・サイバーセキュリティへの包括的なアプローチとして、手始めに2018年の国際標準化機構（ISO）のレポート、『*ISO/IEC 27103:2018 — Information technology, Security techniques, Cybersecurity and ISO and IEC Standards*』[xliii]を使用すると良いでしょう。このISOのレポートでは、セキュリティに対する複数レイヤーから成る“多層防御”[xliv]アプローチについて説明しており、セキュリティの5つの一般的な側面（特定、保護、検出、対応、リカバリ）における推奨事項を提示しています。

データ・メッシュとデータ・ファブリックの概念は、いくつかの点において、データ・セキュリティ環境を複雑化します。たとえば、組織が自社の全データを、単一のアクセス制御リストと単一のポリシー実行ポイント（PEP）を使用して、単一の政治的規制管轄区域内の単一の物理ロケーションにある単一のリポジトリに単純にまとめることができれば、データ運用上のセキュリティとデータ・ガバナンスは大幅に簡素化されるでしょう。しかしながら、メッシュとファブリックで重要なのは、分散データ・アーキテクチャの実現です。他の法人が所有するインフラストラクチャ内でデータ・サービスを運用する、あるいは他国でビジネスを運営するためのビジネス決定には、固有のトレードオフが伴います。非常に大規模な企業では、（データを分散化するために）このようなビジネス決定をすでに下しており、現在は、広範に分散化されたデータをいかに効果的に責任を持って管理するかということが問題になっています。

データ・メッシュとデータ・ファブリックに固有のセキュリティとガバナンスの側面は、主として、データ・ストア間やクラウド間の移動を余儀なくされる移動中のデータに対して効果的なセキュリティとガバナンスを実現することを中心に展開されています。セキュリティの観点から見ると、分散化されたデータがメッシュで極めてセキュアに移動するためには、ネットワーク上のアプローチが重要です。ガバナンスの観点から見ると、分散化されたデータは、データ系統、説明可能性、データの整合性と妥当性確認におけるさらなる課題を生み出します。

分散化された認証および認可

データ・メッシュおよびデータ・ファブリック・サービスが多くのネットワークをまたいでデプロイされている場合、サービスのセキュアな相互運用は、通常、開放型システム間相互接続（OSI）の階層ネットワーク・アーキテクチャのネットワーク・レイヤー6および7の機能です[xlv]。たとえば、WebSocket（WSS）は、一般的にレイヤー7サービスと見なされており、双方向暗号化認証向けのmTLS（mutual Transport Layer Security）と一緒に使用される場合、サービス間通信で非常に高いレベルの信頼性を実現できます。（信頼できる証明書を使用した）相互認証スキームは、次のような攻撃に対する防御に有効です[xlvi]。

- **中間者攻撃** – 第三者が通信を盗聴しようとする行為
- **反射攻撃** – 古いメッセージを再実行してサーバーをだまそうとする行為
- **スプーフィング攻撃** – 悪意のあるアクターが、（信頼できるソース・ドメインから取得した）不正なデータを使用して信頼できる別のユーザーとして振る舞う行為
- **なりすまし** – 悪意のあるアクターが、類似のデータを使用して他のユーザーの信頼を得ようとする行為

ほとんどのサービス・メッシュ・フレームワーク（動的なデータ・ファブリックおよび信頼性のあるデータ・メッシュの基盤レイヤー）では、mTLSベースの認証を使用できます。同様に、パブリック・クラウドのサービスを経由して運用する場合は、通常は認証にmTLSの基盤が提供されます。

さらにアプリケーション・レイヤーでは、ユーザー認証と、特定のリソース、API、およびデータへのアクセス認可を委任する方法を提供することが重要です。OAuth 2.0[xlvii]やOpenID Connect[xlviii]などの標準では、トークンベースのアクセス要求を使用して、セキュアに認可する方法の規範的原型が提供されます。ただし、OAuth 2.0およびXACML[xlix]（認証のためのポリシー記述言語）は、通常は集中型ゲートウェイまたはハブを経由して実行されますが、個別のクラウド・プロバイダが独自のポリシー記述言語を持っていることが頻繁にあります。その場合、分散化されたデータ・メッシュまたはデータ・ファブリックを実行することは相当困難になります。IDのプロバイダとアクセス・ポリシーがネットワークごとに異なるためです。

分散化された認可は、最新のメッシュ型ソリューションにスケール・アウトできなければなりません。これを行う方法の1つは、メッシュ・コントローラが、さまざまなクラウドやオンプレミス・ツールの認可フレームワークとネイティブに相互運用されるようにすることです。別の方法は、メッシュの一部として実行できる分散化されたポリシーベースのコントローラを実装し、サポートすることです。Open Policy Agent（OPA）プロジェクトlは、そのような選択肢の1つです。

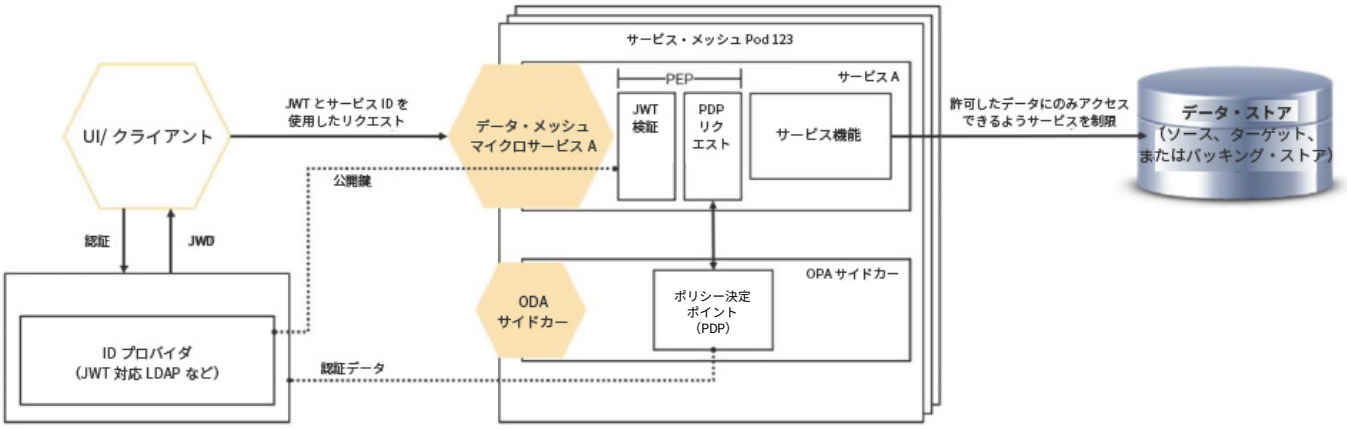


図21：マイクロサービス内の分散化されたPEPおよびPDP向けのOpen Policy Agentサイドカー

OPAアーキテクチャでは、各マイクロサービス（データ・メッシュ・マイクロサービスなど）は独自のポリシー実行ポイント（PEP）です。ただし、意思決定はOPAサイドカー・プロキシ経由で処理されます。OPAサイドカーによって評価されたポリシーは、各サイドカーに固有であり、OPAクライアントによって操作できます（すべての通信はREST APIです）。加えて、（‘Rego’と呼ばれるJSON表記法で記述された）ポリシー自体は、LDAPストアに保存され、認証時にJSON Web Token（JWT）を介してリフレッシュされるか、定期的にレプリケートされます。

マルチクラウド・メッシュ・アーキテクチャでは、PIP（ポリシー情報ポイント）の集中化を存続させながら、データ・メッシュの個々のマイクロサービス全体にPEP（ポリシー実行ポイント）とPDP（ポリシー決定ポイント）を分散させる手段として、ODAサイドカー・プロキシ・パターンを活用できます。この方法では、個別の意思決定に単一のハブアンドスポークは必要ありません。大抵はクラウド・プロバイダに固有の無数の異なるIAM（Identity Access Manager）PDPにポリシーを変換する必要もありません。分散化されたメッシュおよびファブリック・サービスは、多様なネットワークでポリシーを共有できます。

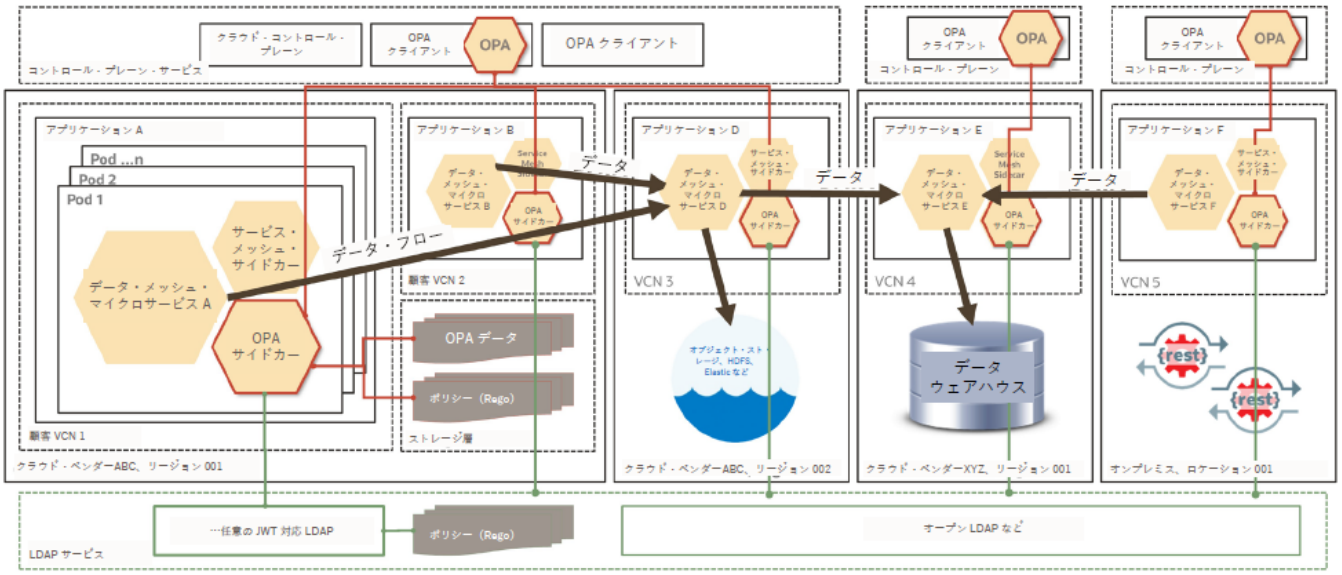


図22：共通のポリシー記述言語を使用した、分散化されたマルチクラウド・ポリシーの意思決定向けのOpen Policy Agentサイドカー

WSS、mTLS、OAuth 2.0、およびOpen Policy Agentを組み合わせることで、動的なデータ・ファブリックまたはデータ・メッシュとして実行される極めてセキュアな信頼性のあるサービスを実現できます。極めて分散化されたデータ・サービスは、TCPやHTTPなどの低層プロトコル上で相互運用できます。一方、分散化された認可については、分散化されたポリシー決定のOPAに類似したパターンによって処理されます。OPAポリシー記述言語（Rego）は、データ層のアクセス・ポリシーで使用されることもある柔軟性の高いJSON表記法であり、複数サービスのRESTエンドポイントのAPIレイヤーでデータセット（ペイロード）をフィルタリングするときに特に有用です。しかしながら、永続層（データ・ストアなど）やメッセージ・サブシステム（Kafka、JMSなど）内では、引き続ききめ細かいデータ・アクセス制御が必要です。

きめ細かいデータ・アクセス制御

いろいろな意味で、データ・アクセス制御に対するOPAのアプローチは管理がもっとも簡単です。統一されたポリシーと分散化できるPEPおよびPDPを使用して、API主導のアプローチを提供できるためです。しかしながら、大半のエンタープライズ・ドメインには、自社の大多数のレガシー・データ・ツールで最新のエージェントベースのフレームワークを使用する余裕はありません。データベース、メッセージング・フレームワーク、ビッグ・データ・レイクといった大部分のデータ中心ツールには、通常は専有のPEPとPDPが組み込まれています。最新のデータ・ストアとデータ統合ツールは、LDAPと適度に統合されますが、多くの場合、相互運用性は、認証やシングル・サインオン（SSO）などのより簡単な要件に限定されます。きめ細かいデータ・アクセス制御においてもっとも困難な要件は、データ・ストア自体に委ねられるのが通例です。

その他のアクセス制御として以下が挙げられます。

- **ロールベース・アクセス制御（RBAC）** – データ・メッシュ/データ・ファクトリ・ツールは、さまざまなユーザー・ロールに対して独自のRBACモデルを持つことで、プラットフォームの機能に対するさまざまなアクセス・レベルをサポートします
- **APIアクセス制御** – ポリシー・エージェントのアプローチが取られなかった場合、APIゲートウェイを、データ・メッシュ/データ・ファブリック・クライアントが使用するREST APIへの外部アクセス制御として使用できます。APIゲートウェイは通常、“East-West”トラフィックよりも“North-South”トラフィックを重視した中核的なサービス・メッシュ・セキュリティを補完します
- **データベース、データ・レイク、およびメッセージングのアクセス** – 個々のデータ・リソースには独自の組込みRBAC制御があり、データ・リソースから見ると、データ・メッシュ/データ・ファブリック・ツールは通常は単なる普通のクライアントです。特定の列、プロパティ、スキーマといった各要素に対するきめ細かいアクセスは、データ・リソースに固有のポリシーによって管理される場合があります

- **カタログおよび種別** – 適切に管理されたデータ・メッシュ/データ・ファブリック・フレームワークには、カタログから管理される資産を種別できるローカルなデータ・カタログがあります。プラットフォームのRBAC制御は、これらのカタログの種別とメタデータに基づき、アクセスを制限できなければなりません
- **クラウド・コンパートメント** – パブリック・クラウドには、通常は何らかのコンパートメントがあります。これらのコンパートメントは、特定のクラウド・テナントに関連付けられたリソースにRBAC制御の追加レイヤーを提供するために、IDサブシステムによって使用されます

分散化されたあらゆるソフトウェア環境と同じく、アクセス制御に対する“多層防御”のアプローチには、複数レイヤーから成るモデルが必要です。

セキュリティに関するその他の考慮事項

分散化されたデータ・ファブリック/データ・メッシュのデプロイメントにとって重要なセキュリティに関する考慮事項は、他にも数多く存在します。本書はセキュリティを深く掘り下げることを目的としていませんが、以下に触れておくことは重要です。

- **台帳の暗号化** – 台帳の種類を問わず（イベント・ソーシング、イベント・ストリーム、データ・レプリケーション、ブロックチェーンなど）、基盤となるストレージ層は、ディスク上で完全に暗号化できなければなりません
- **台帳の不変性** – ブロックチェーンでは、台帳の種類は、コミットのクォーラムと、コミットされたトランザクションの不変性が求められる保護と制御を備えている必要があります
- **ウォレット、キー・ストア、シークレット・ストア** – マルチクラウド・データ・ファブリック/データ・メッシュは、デプロイされるいずれのインフラストラクチャでも、多様なウォレット、シークレット・ストア、およびキー・ストアAPIを使用できなければなりません
- **プリンシパルを使用した資格証明管理** – 分散化されたメッシュでは、あるユーザーの“代わりに”、他のアクターにアクセスを付与することが必要な場合が頻繁にあります。呼び出しツールがユーザーの資格証明についての機密情報を検出できなくても、資格証明が外部のデータ・ストアに提供されるようにするために、一部のツールは、プリンシパルの参照、受け渡し、および受け取りが可能でなければなりません
- **ネットワークの暗号化** – 暗号化は、さまざまなプロトコルを使用して、複数のネットワーク・レイヤーに適用できます。データ・メッシュ/データ・ファブリックの典型的なデプロイメントは、暗号化されたプライベートな専用回線で実行される場合もあれば、通信にIPSec VPNクライアントが使用される場合もあります。ペイロード自体は、HTTPSまたはSSL/mTLS経由のWebsocketで実行される場合があります
- **リバース・プロキシ** – 通常はポートのルーティングを簡素化する上で重要な役割を果たすセキュリティ・リバース・プロキシを、悪意のあるリクエストに対するクライアント呼出しを調査するために導入することもできます
- **ハイサイド・ガード/ローサイド・ガード** – 国家のセキュリティ・ドメインでしばしば使用されるセキュリティの分離された優れたネットワークでは、通常は‘ガード’・ソフトウェアを使用して、ネットワーク間を移動するペイロードを調査します
- **ターゲット開始型パス –** ファイアウォールや非武装地帯（DMZ）を横断するトランザクション台帳の分散レプリケーションでは、外部からインバウンド接続を開始できない場合に、受信者サービスがアウトバウンド接続を開始できます（信頼性の高いロケーションが信頼性の低いロケーションへのリンクを作成）

セキュリティは複雑なトピックになり得るため、分散データ・アーキテクチャを全面的に採用する場合は適切に設定することが不可欠です。動的なデータ・ファブリックまたは信頼性のあるデータ・メッシュ用のエンタープライズ・ツール・キットによって、セキュリティ制御と高度なデータ・ガバナンスのための高性能で検証可能な機能が提供される必要があります。

データ・ガバナンス

動的なデータ・ファブリックまたは信頼性のあるデータ・メッシュでのガバナンス制御は、ほぼあらゆる点において、モノリシックなデータ管理アーキテクチャの一部としてこれまで使用されてきたガバナンス制御を反映していなければなりません。基盤となるテクノロジーが変わったからといって、規制とコンプライアンスに対するビジネス要求が変わるわけではありません。IT部門がどのようなツールを使用していようと、データ・プロダクト・ファクトリがいかにスマートであろうと、ガバナンスに対するトップレベルの要件は引き続き存在します。

データ・ガバナンスのプラクティスが動的なデータ・ファブリックと信頼性のあるデータ・メッシュのプラクティスとどの程度異なるかは、主にこれらの新しいアプローチ固有の特性に依存します。基本的に、データ・プロダクト思考の性質によって、データ・プロダクト管理にKPIとSLAによる高水準のガバナンスがもたらされます。データ・サービスの分散化されたモジュール式メッシュの第一原理として、データ・ガバナンス技法は、同じ場所に配置されずに根本的に分散化されたデータ資産と本質的に連携する必要があります。イベント台帳の使用の第二原理として、移動中のデータ・イベントに適用されるガバナンス・ポリシーと、静的データに適用されるより従来型のガバナンスが必要です。信頼性のあるポリグロット・データの第三原理として、データ・フローが（データ・セマンティックとともに）失われないようにするために、ガバナンス・ポリシーが、トランザクション・セマンティックの検証可能性を追跡および記録する必要があります。

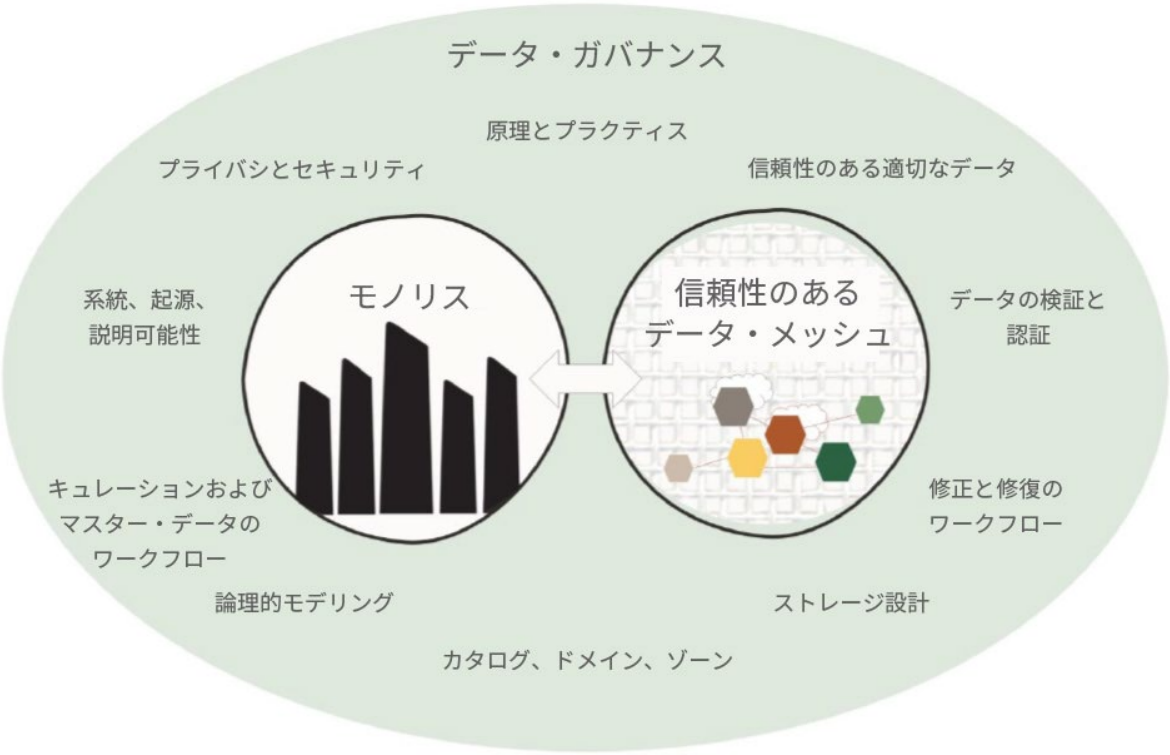


図23：信頼性のあるデータ・メッシュのデータ・ガバナンスは、管理されたデータ・ファブリックおよびレガシー・データ・モノリスと連携する必要がある

大半の大企業では、レガシー・データ管理のモノリスは、動的なデータ・ファブリックと信頼性のあるデータ・メッシュのための新しいテクノロジーと共存することになるため、ガバナンスのフレームワークは、新しいデータ・アーキテクチャと古いデータ・アーキテクチャの両方にまたがる必要があります。データ・ガバナンス自体は、単なるツールの集合ではなく、プラクティスです。人材やプロセス制御がなければ、データ・ガバナンス用のどのようなソフトウェア・ツールも確実に失敗に終わります。動的なデータ・ファブリックと信頼性のあるデータ・メッシュに適用されるデータ・ガバナンスのもっとも重要な領域のいくつかを以下に示します。

データ・ガバナンスの懸念領域		固有の考慮事項
原理とプラクティス	組織のレポート階層、事業部門間の調整、企業の権限、経営幹部の支援などの人的要素、およびデータ・スチュワードを支える文化に吹き込まれるベスト・プラクティス。	リアルタイムのデータ・アーキテクチャへの移行は、ガバナンスの失敗の影響が表面化するまでの時間が短縮されることを意味します。迅速なレスポンス・チームがDevOps/DataOpsと連携する必要があります。
データ・カタログ、ドメイン、ゾーン	大半の大企業には、ビジネス・システムや分析ツールからメタデータをスキャンして取り込むさまざまなデータ・カタログがあります。これらのカタログは、必然的にさまざまなビジネス・データ・ドメインとさまざまな物理データ処理ゾーンにアクセスします。	データ・プロダクト思考を採用することで、組織がカタログをデータ・プロダクトとデータ・ドメインに連携することが促進されるはずです。ドメインを横断する'カタログのカタログ'を検討してください。
信頼性のある適切なデータ	さまざまな処理セマンティックを持ち、さまざまなエンタープライズ・ツールにまたがるデータ・パイプラインは、データ整合性の喪失、見捨てられたレコード、重複データや不正確なデータの原因になる可能性があります。	スマート・データ・ファクトリの概念によって、確実にACID/ポリグロット・データセットが適切に処理されるようにする制御が組み込まれる必要があります。
データの検証と認証	どのような分散システムでも、時間とともにデータ・ドリフトが発生する可能性があります。データ・ドリフトには、人的要因や機械的要因によるスキーマ定義の変更やレコードの紛失が挙げられます。検証と認証の制御によって、データのコピーが信頼できる情報源と一致することを確認できなければなりません。	分散データ・アーキテクチャでは、データの認証は、レコードの数を数えたり、レコードを比較したりすることだけではありません。データタイプの変換が発生する可能性もあるため、値とスキーマを比較することも不可欠です。
系統、起源、説明可能性	データはネットワークやデータ・パイプラインを横断して移動するため、複数回変換され、他のシステムを起源とするデータ値と結合される可能性があります。系統により、データ・フローの正確なトレースが提供され、起源により、データの信頼できる最終的な情報源が示されます。また、説明可能性により、抽出されたデータが作成された方法についての詳細な説明（推論されたデータ値、自動化された結合など）が提供されます。	リアルタイムのイベント駆動型システムでは、通常、APIへの外部イベントなど、レコードレベルのトレーサビリティが要求されます（データの'セット'やシステム間のインタフェースを検査する静的系統や従来のバッチとは異なります）。
キュレーションおよびマスター・データのワークフロー	管理データ・レコードの概念は、MDM（マスター・データ管理）システムを起源とし、参照データもその範囲に含まれます。キュレーションされ、厳格に管理されたこのデータの例として、顧客レコード、製品レコード、市民データのほか、法典、国コード、分析レポートのディメンションといった参照データが挙げられます。	データ・プロダクト思考は、この分野に自然に拡張されており、データ・ドメインおよびパッケージ化/管理されたデータ・プロダクトの管理は、マスター・データが従来から管理されてきた方法と概念上類似しています。
データのプライバシとセキュリティ	国際的な規制（Basel/BCBS 239、忘れられる権利など）と密接に関係するデータ・プライバシの側面には、管理レコードが適切に削除される保証といったあまりにも単純な懸念事項から、抽出データを生成する機械学習アルゴリズムでのプライベート・データの使用といったより複雑なトピックに至るまでが含まれます。データ・セキュリティはありきたりなテーマですが、データのアクセス制御、マスキング、および暗号化プロトコルと同様に重要です。	分散データ・アーキテクチャでは、動的なデータ・マスキングの側面がより重要になります（ディスクのデータを除去）。ポリグロット・データでは、データ・パイプライン内にデータが入り混じっているため、アクセス用のポリシー制御は'多層'になる場合があります。
データ修正のための修復ワークフロー	すべてのコンピュータ・システムにおいて、データは必然的に破損のリスクにさらされています。人的要因や機械的要因が破損を引き起こす可能性があります。 修復のためのワークフローにより、データ・スチュワードとデータ・プロダクト・マネージャーのチームに対して、ソースやパイプラインにあるデータやデータ・プロダクトの出力データを修復するための多層に及ぶ制御が提供されます。	データ・プロダクト思考は修復のアプローチを変える場合があります。たとえば、データ・プロダクト・マネージャーがワークフローのSLAを直接管理できるようにします（従来はDBAまたはデータ・エンジニアによって管理されていました）。

データ・ガバナンスの倫理上の懸念事項

人工知能（AI）が出現した当初から、よりスマートになり、自律性が増すテクノロジーを使用することについての倫理的な問題の調査に専心する有力なサブカルチャーの存在がありました。1990年代に台頭したエンタープライズ・データのデータ・ガバナンスは、企業の規制遵守を取り巻く現実的な懸念事項に主に重点を置いてきました。しかしながら、ビジネス変革、“インダストリ4.0”、およびサイバーフィジカルなシステムにおいてデータの重要性が増すのに伴い、データ・ガバナンスの倫理的な懸念事項をより重視することを求める声が多くの専門家から上がるようになりました[ii]。倫理的な懸念事項は、コンプライアンスと密接に関連するデータ・ガバナンスの側面とは異なり、組織がデータを使用して何を行うか（または何を行わないか）、データがどのように収集され、どのように使用されるかといった疑問から生じる可能性があります。これらの倫理的な懸念事項は、情報に基づく同意、匿名性、プライバシ、透過性についての判断に影響を与える可能性があります。データ・ガバナンスに対する最新のアプローチでは、このような倫理的な懸念事項を取り込み、データ・スチュワードのライフサイクルに伝達するツールが可能になるかもしれません。

データ管理アーキテクチャの相対的な成熟性にかかわらず、データ・ガバナンスのポリシーは、システムに欠かせない部分として適用される必要があります。極めて成熟した動的なデータ・ファブリックおよび信頼性のあるデータ・メッシュのアーキテクチャでは、データ・ガバナンスの役割は、後で追加されるのではなく、第一原則でなければなりません。単一のITツールがデータ・ガバナンスに関するすべてのニーズを満たすことはできませんが、ファブリック/メッシュでは、ツールは依然としてデータ・ガバナンスに不可欠な基盤です。これは、Oracle GoldenGateプラットフォームが、信頼性のあるデータのユースケースにおいて業界のリーダーである理由の1つです。

Oracle GoldenGate：データ・メッシュとの信頼できる橋渡し役

Oracle GoldenGateテクノロジーは、業界唯一のテクノロジーです。最新の分散化されたマイクロサービス、ポリグロット・ペイロード機能、CTL（連続変換およびロード）や時系列分析向けのストリーム処理とともに、信頼性のあるデータ・レプリケーション機能を長きにわたり提供していることがその理由です。

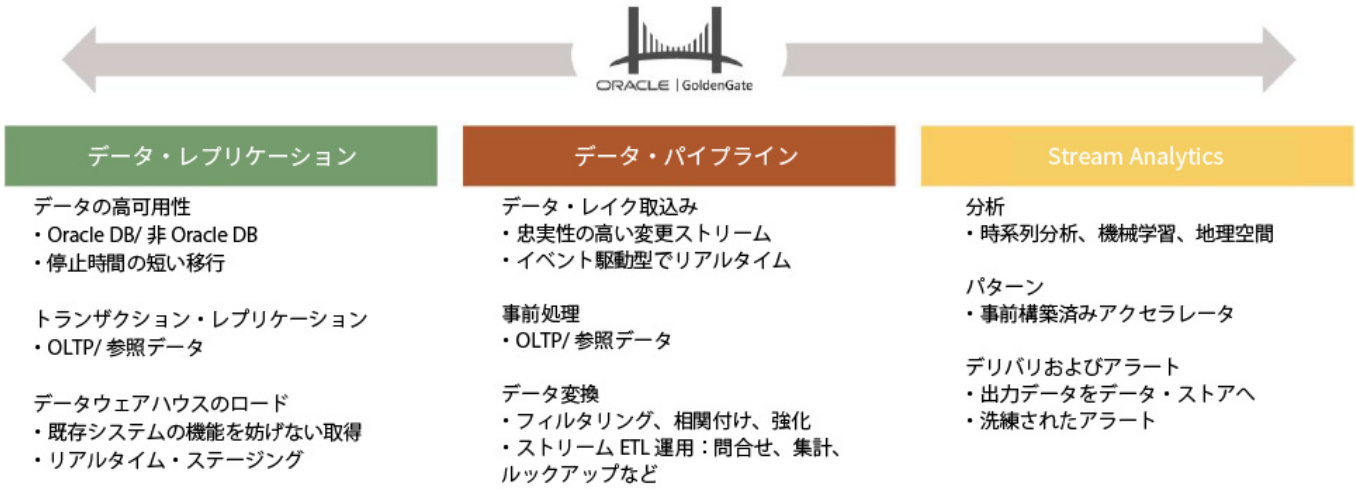


図24：GoldenGateプラットフォームの機能

信頼性のあるリアルタイム・データを優先させる組織にとって、Oracle GoldenGateは理想的なソリューションです。20年以上の間、GoldenGateは信頼性のあるデータ・ファブリック・アーキテクチャの定評あるリーダーであり、GoldenGateの最新バージョンは、データ・プロダクト思考、分散アーキテクチャ、イベント駆動型パイプライン、およびポリグロット・データ・ペイロードを中心に構築される信頼性のあるデータ・メッシュ機能の基礎を築きます。GoldenGateプラットフォームでは、チェンジ・データ・キャプチャ（CDC）、信頼性のあるリアルタイムのデータ・レプリケーション、データ取込み、連続変換およびロード（CTL）用のデータ・パイプラインのほか、ストリーム・データでの多様な分析が可能です。

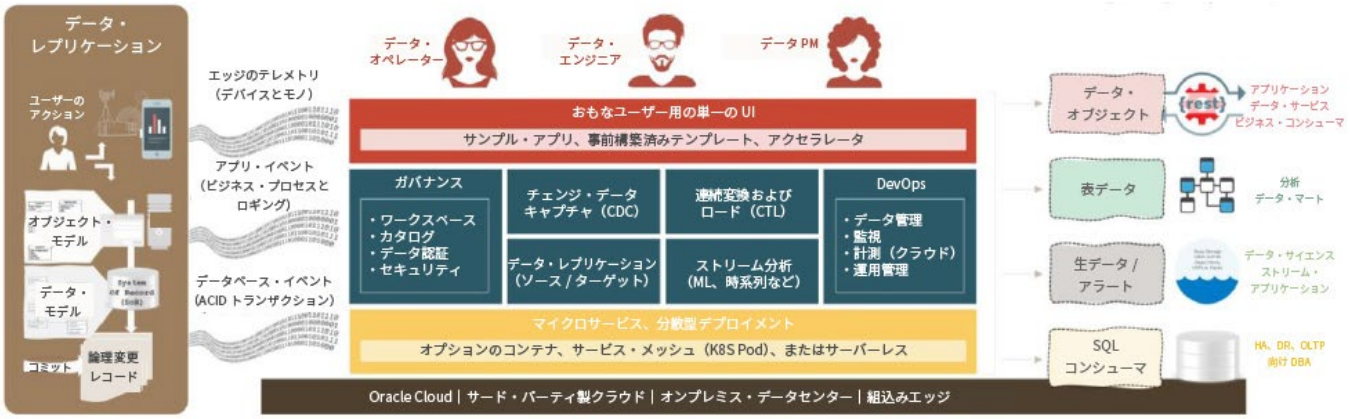


図25：GoldenGateプラットフォーム、論理的アーキテクチャ、および主要コンポーネント

動的なデータ・ファブリックと信頼性のあるデータ・メッシュのコンテキストでは、GoldenGateプラットフォームは以下の主要機能を提供するように連携されています。

- **高価値のデータ・プロダクト** – GoldenGateプラットフォームによって提供
- **外部データ・プロダクトとデータ・メッシュ・パターンの実現** – 実現化テクノロジーとしてGoldenGateを使用
- **リアルタイム・データ、マイクロサービス、信頼性のあるトランザクション台帳** – 将来に対応したアーキテクチャ
- **セルフサービス、ローコード、厳格なガバナンス** – 非テクニカル・ユーザーも使用できるように設計
- **ワールドクラスのストリーム処理、オープン・コア** – オープンソース・テクノロジーを再パッケージ化しただけではない独自の価値

これらの機能の組み合わせにより、組織がモノリシックな古いデータ・アーキテクチャから、動的なデータ・ファブリックと信頼性のあるデータ・メッシュ設計の次世代アーキテクチャへと移行できるよう支援する強力な架け橋が提供されます。

GoldenGateによって提供されるデータ・プロダクト

GoldenGateプラットフォームでは、データ・プロダクトそのものを提供できるほか、実現化テクノロジーとして他のデータ・プロダクトもサポートされます。本書の「データ・プロダクト思考」のセクションで説明したように、データ・プロダクトの形態や構造はさまざまです。すべてのデータ・プロダクトは、（i）データの使用を通してビジネス成果を直接後押しし、（ii）正式なKPIとSLAを使用して管理されます。Oracle GoldenGateストリーム・データ・プラットフォームから直接提供されるデータ・プロダクトとして、以下が挙げられます。

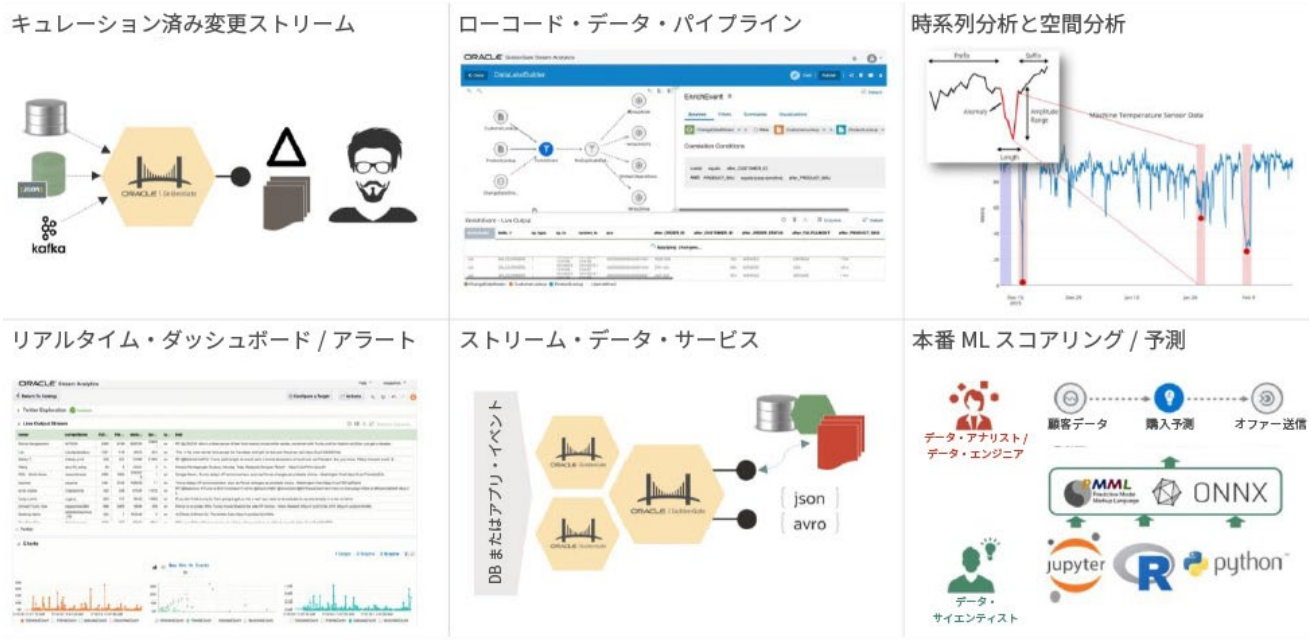


図26：Oracle GoldenGateストリーム・データ・プラットフォームから直接提供されるデータ・プロダクト

キュレーション済みの変更ストリーム

本書の「エンタープライズ・データ台帳」のセクションで提示したチェスの例のように、データの変更ストリームは、チェスの対局中の移動履歴にアクセスできるようなものです。信頼できる完全なストーリーを伝える変更ストリームは、データのセマンティックを理解する上で不可欠です。GoldenGateは、多種多様なリレーショナル・データ・ストア、非リレーショナル・データ・ストア、およびメッセージ・システムで、変更されたデータを検出できます。GoldenGateには、イベントのローカル・レコードを保管する信頼性のあるイベント台帳が内蔵されています。壊滅的な障害が発生した際は、このイベント台帳をデータの信頼できるリカバリ・ポイントとして使用できます。GoldenGateプラットフォームにあるマイクロサービスの1つによって、変更ストリームに‘分散パス’が提供されます。この分散パスは、キュレーションされたデータ・オブジェクトの集まりです。分散パスの登録コンシューマは、パス内のあらゆるオブジェクトに対する変更イベントの継続的なフローを受信します。

データ・パイプライン

データ・パイプラインは‘データ・プロダクト・ファクトリ’の基盤であり、ETLまたはCTLデータ・フローを実現する重要な要素です。リアルタイム・ストリームでは、データ・パイプラインは、ウィンドウイング機能、データ変換、および分析関数から構成されます。すべてのデータ・パイプラインが本格的なデータ・プロダクトとして管理されるわけではありませんが、多くはそのように管理される可能性があります。たとえば、あるデータ・パイプラインが、リアルタイム分析ビューや、多くのパイプラインで集計しなければならないマスター・データ・オブジェクトの変更ストリームを作成するなどの理由で、外部ユーザーによって消費されている場合、そのデータ・パイプラインは、確立されたKPIとSLAを使用して厳格に管理されている可能性があります。

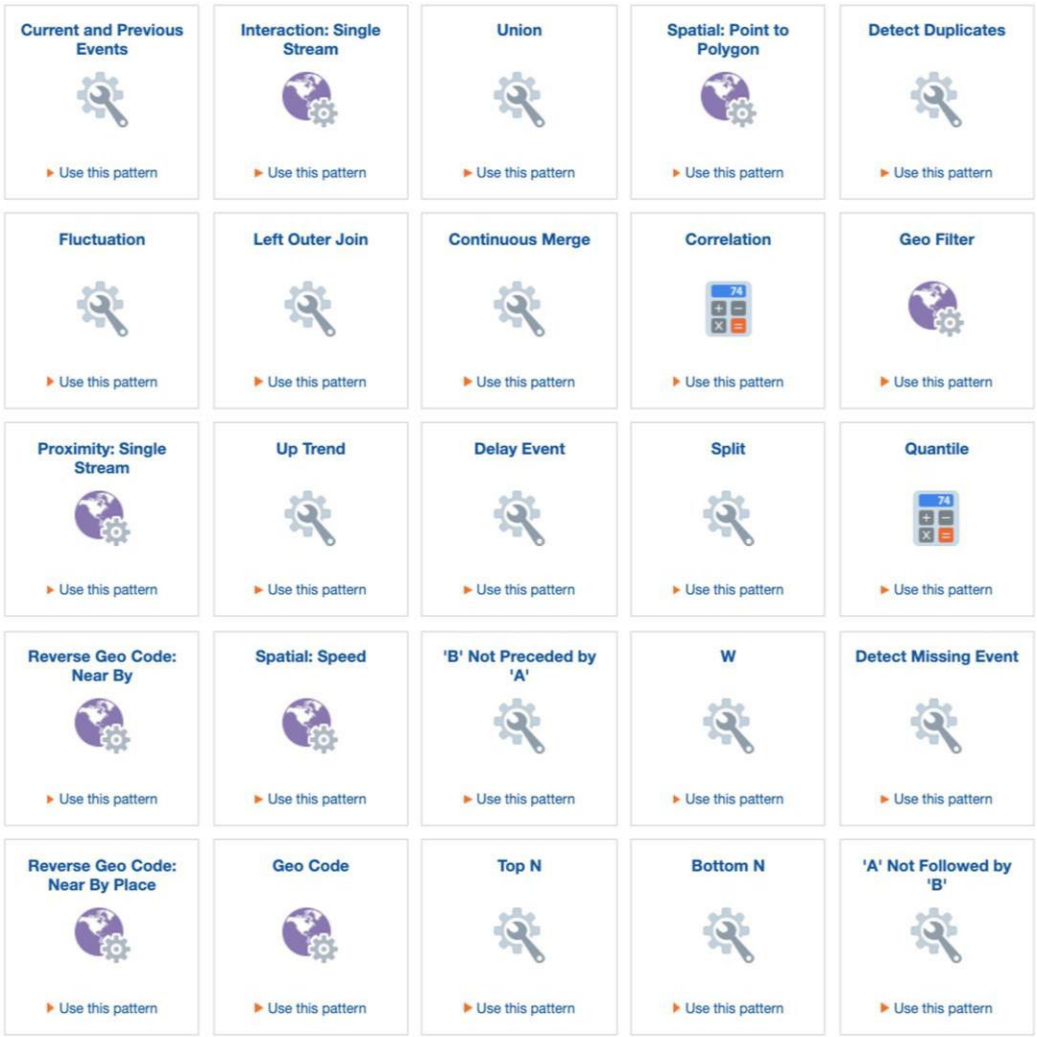
GoldenGateデータ・パイプラインでは、以下の機能がサポートされます。

ウィンドウイング機能	データ変換操作	分析パターン
• グローバル・ウィンドウ • 固定ウィンドウ • スライディング/タンブリング・ウィンドウ • セッション・ウィンドウ • カスタム・ウィンドウ • カスタム・マージ・ウィンドウ • グループ・タイムスタンプ・ウィンドウ	• フィルタリング • 集計 • 結合 • 式 • 相関付け • 問合せ/関数 • カスタム関数	• 時系列 • 地理空間 • 種別 • 機械学習（PMML、ONNX） • Apache Spark（ネイティブ） • ルール・エンジン

時系列分析

Oracle GoldenGate Stream Analyticsでは、ナノ秒スケールまでの順序付けされたイベント処理がサポートされます。組込みのパターンでは、セルフサービスのData Product Managerによってデータ・パイプラインに迅速に取り込むことができるノーコード・ライブラリが提供されます。事前に構築されているこれらのパターンの多くでは、時系列関数と分析関数が提供されます。大抵の場合、時系列分析は、パイプライン自体で直接製品化されますが（アラート用など）、時系列データの出力は、ビジネス・データ・コンシューマが直接消費するのに適したリアルタイム可視化機能にも送信することができます。

図27：GoldenGate Stream Analytics内の時系列分析パターン・ブラウザ



リアルタイムのダッシュボードとアラート

パイプラインのデータ・プロダクトには、ビジネス成果を直接後押しし、特定のKPIまたはSLAの下で管理される任意のデータ・アーティファクトが含まれます。この定義には、電子メール、SMS/携帯メール、Kafkaトピック、またはイベントのアラートに使用される可能性のあるその他のゲートウェイ経由でビジネス・ユーザーに通知を送信できるアラート・パイプラインが含まれます。同様に、ライブ・ダッシュボードの直接的なサポートも含まれます。GoldenGate Stream Analyticsでは、データ・キューブ用のネイティブなアーティファクト・タイプと緊密に統合されたデータ・カタログと、10種類以上のデータ可視化がサポートされます。

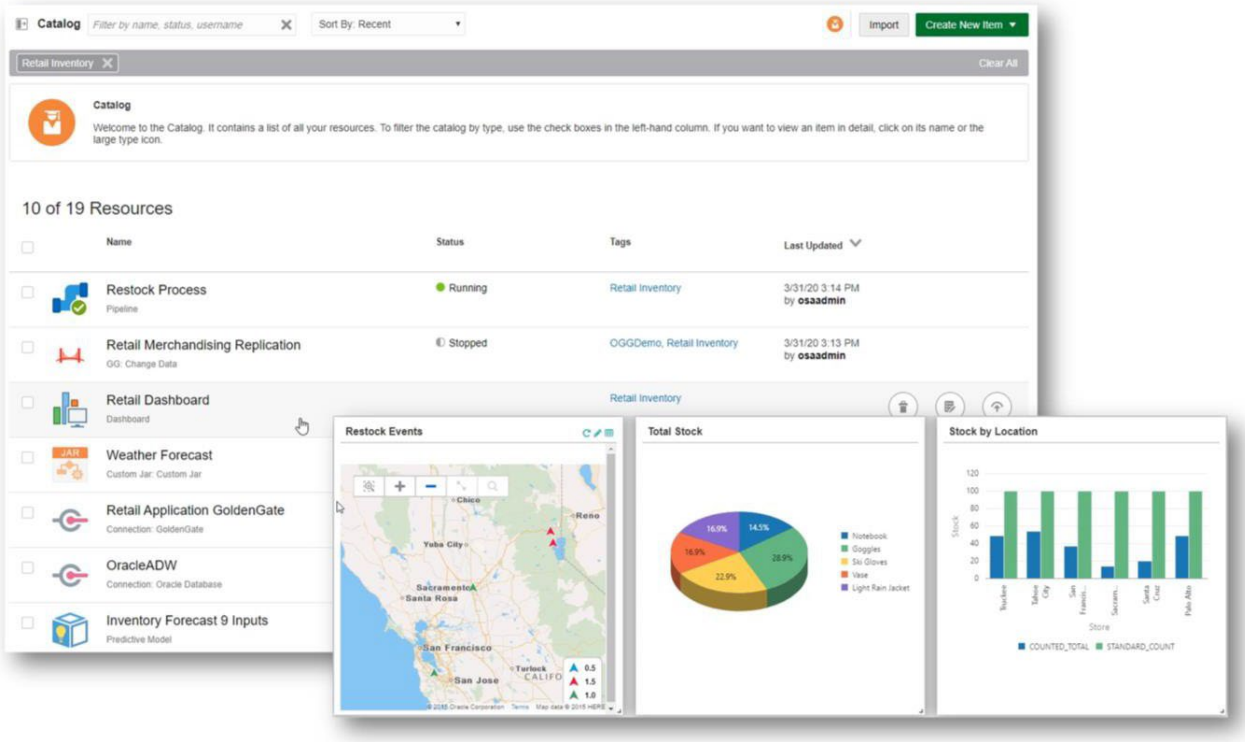


図28：データ・プロダクトには、統合型カタログから直接管理されるダッシュボード、アラート、キューブが含まれる

ストリーム・データ・サービス

多くのITデータ・レイクの実装では、‘生データ・ゾーン’は、下流のETLと分析にとって重要なステージング領域です。これは、物理的製品のサプライ・チェーンで、製品が小売店舗に配送される前や、消費者に直接配達される前の段階として展開される流通センター（フルフィルメント・センター）のようなものと考えることができます。物理的な世界では、多くの生産者の製品を扱い、多くの小売店舗が共用できるサービスを提供するサード・パーティのロジスティクス・プロバイダが存在します。デジタル・データ・プロダクトの領域では、Oracle GoldenGateは、データ・プロダクトの移動、ルーティング、ステージングというロジスティクスを実質的に処理する製品化されたサービスです。GoldenGateプラットフォームでは、リアルタイム・ストリーム、マイクロバッチ、または（時間枠を使用した）大規模なバッチ処理の一環として、完全なデータまたは変更データを継続的にストリームできます。これらのサービスはデータ・プロダクトであり、通常は厳格なKPIおよびSLA一式によって管理されます。

GoldenGateのソース	GoldenGateのターゲット
• データベース：Amazon Aurora PostgreSQL、Amazon Aurora MySQL、Amazon RDS for MariaDB、Amazon RDS for MySQL、Amazon RDS for Oracle、Amazon RDS for PostgreSQL、Amazon RDS for SQL Server、Azure Database for MySQL、HPE NonStop、IBM DB2 for i、IBM DB2 for z/OS、IBM DB2 LUW、MariaDB、Microsoft SQL Server、MySQL Database Server、MySQL Database Server、Oracle Autonomous Database、Oracle Database、PostgreSQL、SAP Sybase ASE	• サポートされるソースはすべて、GoldenGateのターゲットになることもできます • 固有のターゲット：Amazon Kinesis、Amazon Redshift、Amazon S3、Apache Cassandra、Apache Hadoop、Azure Data Lake Storage、Azure Event Hub、Azure Synapse Analytics、Cloudera Data Platform（CDP）、Cloudera Hadoop（CDH）、Confluent Platform、Datastax Enterprise Cassandra、Elasticsearch、Google BigQuery、Greenplum、Hortonworks Hadoop（HDP）Java Message Service

GoldenGateのソース	GoldenGateのターゲット
• 非リレーショナル：Cassandra、MongoDB、Kafka Connect、Java Message Service（JMS） • アプリケーション：Oracle Fusion ERP、Oracle EBS、Oracle Retail Cloud、Oracle Argus Cloud、Oracle Transportation Management • Oracle Integration Cloud および Kafka Connect 経由：Salesforce、Marketo、Workday、MQTT、Jira、Oracle NetSuite、Oracle Commerce Cloud、Oracle HCM、Oracle Responsys、Oracle Engagement Cloud、LinkedIn、SAP Success Factors、Twillio、Twitter、その他120以上	（JMS）、Java DB Connector（JDBC）、MapR Hadoop、MongoDB、Netezza、OCI Autonomous Data Warehouse、OCI Object Storage、OCI Streaming Service、Oracle Cloud Object Storage、Oracle NoSQL、Oracle TimesTen、SAP Hana（JDBC）、Snowflake、Teradata、Vertica（ファイル/JDBC） • Apache Kafka経由：すべて • Oracle Integration Cloud経由：すべて • Oracle IoT Cloud経由：すべて

注：サポートされるテクノロジーの最新情報については、必ずオラクルの担当者に確認してください

GoldenGateは、リレーショナル・トランザクションで信頼できるACID整合性を維持する信頼性のあるデータ・プロバイダであり、データ・プロダクトをさまざまな構文にフォーマットできるポリグロット・ストリーム・データ・サービスでもあります。具体的には、ターゲットのネイティブ形式、HDFSシーケンス・ファイル、デリミタ付きテキスト（行および操作モードの両方）、JSON（行および操作モードの両方）、Avro（行および操作モードの両方）、XML、Parquet、ORC（Optimized Row Columnar）などの構文にフォーマットできます。

本番環境のMLスコアリング/予測

Oracle GoldenGateは、機械学習（ML）モデルを本番環境に導入するための最適なプラットフォームです。主に履歴データでリグレッション分析をサポートするMLモデルもあれば、リアルタイム・データ・トランザクションでスコアリングとリコメンデーション機能をサポートするように設計されたMLモデルもあります。GoldenGateプラットフォームでは、MLモデルをリアルタイム・データ・ストリームに直接統合できるため、個別のイベントを、事前定義済みのモデルに従ってスコアリングできます。たとえば、ストリーム・パイプラインのMLモデルには、顧客に対する次善のアクションとなるリコメンデーション、顧客のアフィニティを特定するスコアリング・アルゴリズム、潜在的な不正行為を検出するための関連MLが含まれる可能性があります。

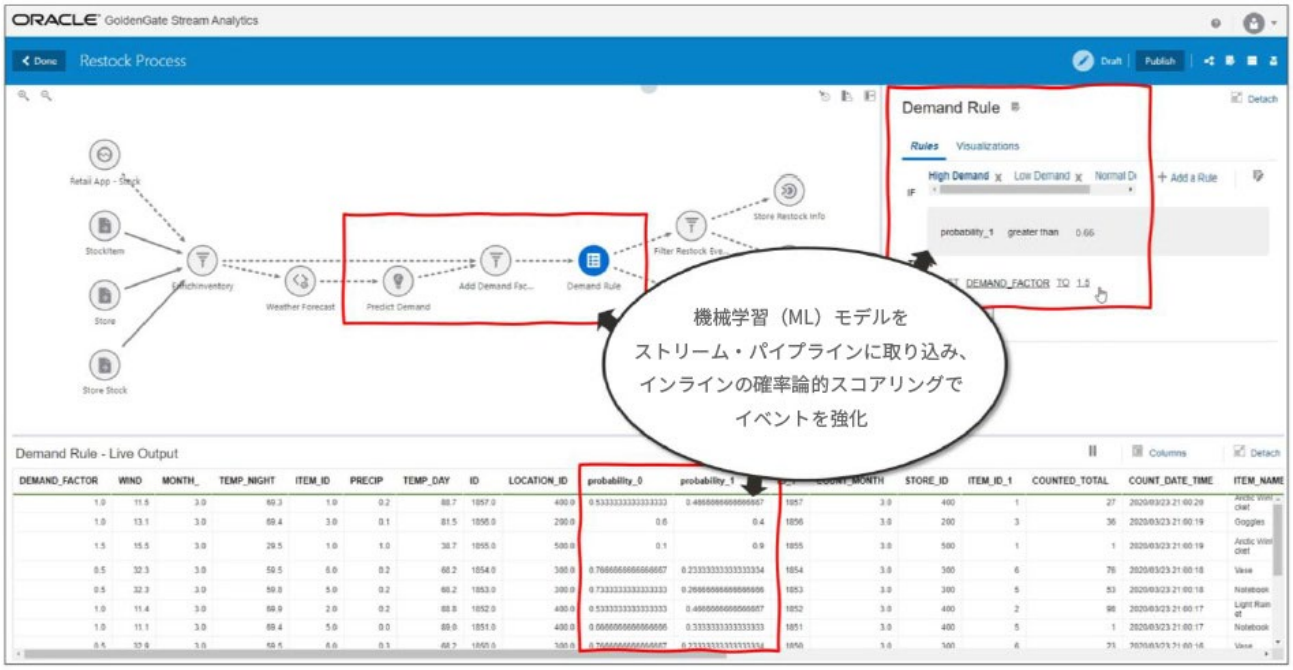


図29：データ・プロダクトは、リアルタイムの予測スコアリングの促進に使用される管理MLモデルになることが可能

MLモデルを実行するデータ・パイプラインは、エッジまたはIoTゲートウェイでの高度なフィルタリングに使用できます（モノのインターネット（IoT）デバイスで生成される不要なイベントを減らすなどの目的で）。イベントを生成するイベント台帳は、マイクロサービス・データ・メッシュにデプロイできます（GoldenGateの分散パス、Apache Kafkaトピック、あるいはJava Message Serviceのメッセージとして）。より集中型のデプロイメントは、任意のパブリック・クラウドで簡単に実行できます。

マイクロサービス、クラウドネイティブなアーキテクチャ

Oracle GoldenGateは、マイクロサービス・アーキテクチャへの移行を実現した最初のデータ・レプリケーション・テクノロジーです。2017年にサンフランシスコで開催されたOracle Open Worldイベントで、オラクルは次世代アーキテクチャの内覧を開始しました。
REST APIを単に既存のモノリシックなツール上に配置する他のベンダーとは異なり、GoldenGateアーキテクチャは、モノリスへの依存をなくし、完全にカプセル化されたマイクロサービス・メッシュ形式のアーキテクチャを採用するように全面的にリファクタリングされていました。

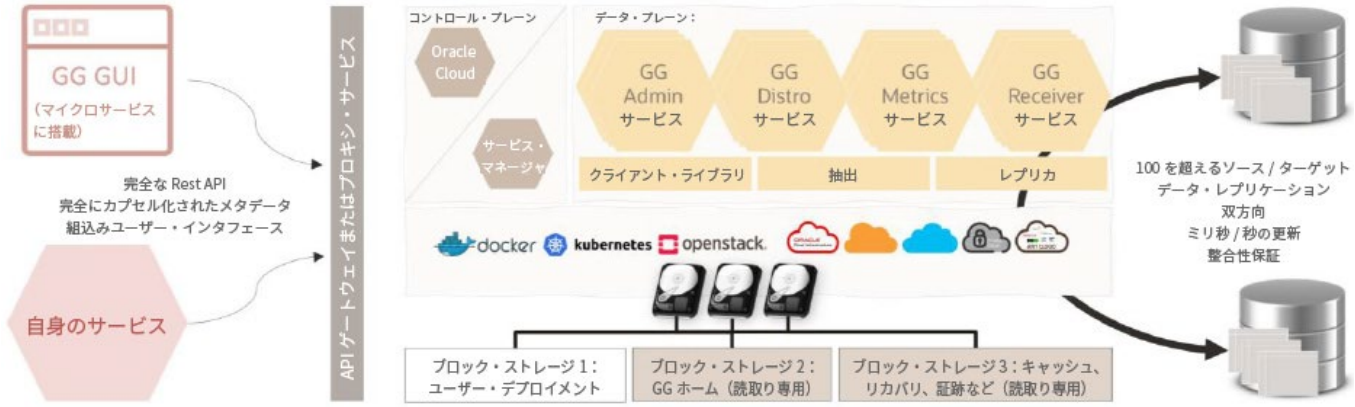


図30：GoldenGateのマイクロサービス・コンポーネント

GoldenGateのオンプレミス・デプロイメント内では、GoldenGate Service Managerが、自身と関連付けることができる他のGoldenGateマイクロサービス・デプロイメントのコントロール・プレーンとして機能します。GoldenGate Administrationマイクロサービスは、データ運用スタッフ/DBAが抽出やレプリカを管理するためのサービスです。Distributionマイクロサービスは、変更データ・パスを公開して、WAN/LAN全域でセキュアな分散を行うことができるサービスです。Receiverマイクロサービスはこれらの分散パスを受信し、メトリック・サービスによって、あらゆるテレメトリとメトリックへのセルフサービスのアクセスが提供されます。

Oracle Cloud内でネイティブなフルマネージド・サービスとして実行されている場合、コントロール・プレーンはOracle Cloudです。このフルマネージドのGoldenGateクラウド・サービスでは、従量課金、ワークロードに合わせた自動スケーリング、自動バックアップとリカバリ、自動化されたパッチ適用など、多くの利点が提供されます。

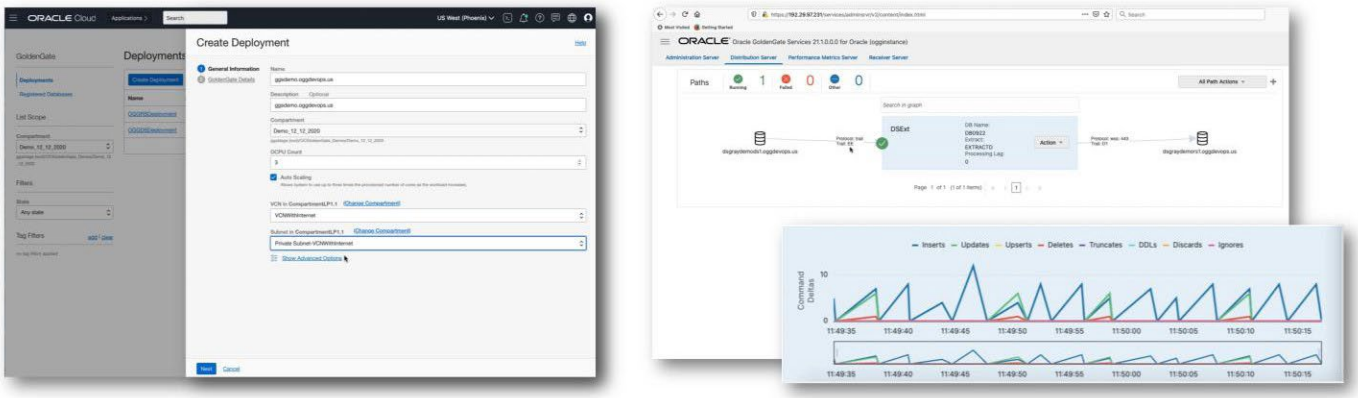


図31：Oracle Cloudから完全に管理されるGoldenGateと、マルチクラウドまたはオンプレミスからのグラフィカルなユーザー・エクスペリエンス

リアルタイムのイベント駆動型データ移動

GoldenGateはソース・システムから直接イベントを取得できます。GoldenGateがソース・システムと同じネットワーク上の同じホストにインストールされているか、あるいは全く別のデータセンター内の環境にインストールされているかは問いません。これらのイベントは、ソース・システムから送信されるとすぐに検出されます（つまり、ポーリングベースのソリューションではなく、リアルタイム・ソリューションです）。これを受けてGoldenGateでは、イベントを任意の登録ターゲット・システムに即座に送信します（マイクロバッチは開発者が指定しない限り使用されません）。さらに、DistributionおよびReceiverマイクロサービスを利用すれば、GoldenGateをWAN全域でマイクロサービスのメッシュとしてデプロイできます。これらのマイクロサービスは、ソース/ターゲット・クライアント・フレームワークから分離されたネットワークとプロトコルを提供するため、極めて有用です。一部のソース/ターゲットは、待機時間の影響を受けやすいため、ストリーム・イベントが長距離を移動しなければならない場合や、信頼性のないネットワーク経由で移動しなければならない場合、異常が発生する可能性があります。DistributionおよびReceiverサービスでは、セキュアなソケットと証明書ベースのmTLSを使用した、耐性と柔軟性が高いセキュアなルーティングが提供されます。

証跡プロトコルとしてのGoldenGateのトランザクション台帳

Apache Kafkaと同じく、Oracle GoldenGateは内蔵されたトランザクション台帳によって駆動されます。GoldenGateでは、これを‘証跡’と呼びます。ディスク上にある場合は（リカバリ目的で使用される場合など）、証跡ファイルと呼ばれ、メモリにあり、ネットワーク経由で転送される場合は、機能上、証跡プロトコルです。GoldenGate証跡は規範形式です。どのソースでデータ・イベントが消費されるかにかかわらず、GoldenGateが処理している間はデータ・イベントは証跡形式に再フォーマットされるためです。この仕組みにより、GoldenGateは極めて高いレベルの信頼性を実現できます。証跡形式のセマンティックは、GoldenGateに緊密に統合されており、20年以上の間、さまざまなデータベースやデータタイプで高可用性（HA）およびディザスタ・リカバリ（DR）の形式として使用されてきました。

GoldenGateは、ソース・データベース自体のコミット・ログと緊密に結合されているため、データベース・ペイロードの参照整合性を維持します。各データベースのコミット戦略のセマンティックはわずかに異なる可能性があります。たとえば、MongoDBでACIDトランザクションがサポートされる方法は、Oracle DatabaseやPostgreSQLとは異なります。
GoldenGateの統合は、最高レベルの信頼性を実現するように設計されているため、データ・コンシューマとデータ・プロダクトは、適切で信頼性があります。GoldenGate証跡には、すべてのトランザクションに一意の変更番号（SCN/CSN）を持たせる機能が内蔵されています。この一意の番号は、トランザクションが適切にグループ化され、正しい順序で適用されるようにするために使用されます。

GoldenGateの下流では、GoldenGate証跡を多種多様な非リレーショナル・ターゲット（クラウド・オブジェクト・ストレージ、Apache Kafka、ElasticSearchなど）と統合でき、開発者は、GoldenGate証跡のメタデータを利用して、カスタムのデータ・パイプラインを強化できます。GoldenGateデータ・パイプライン・テクノロジーは、Distributionマイクロサービスおよび証跡メタデータとネイティブに統合されるため、CTLパイプライン処理では、整合性のある適切なデータが使用され、信頼性も確保されます。

GoldenGateによって実現されるデータ・ファブリックおよびデータ・メッシュ・パターン

最新のエンタープライズ・データ・アーキテクチャでは、多種多様なデータに対する多数の要件が定められており、さまざまなベンダー・テクノロジーを活用して、エンド・コンシューマのデータ・プロダクトが提供されます。Oracle GoldenGateプラットフォームを使用すると、あらゆる種類のデータ・プロダクトを提供するソリューションを、さまざまなベンダーやクラウド・プロバイダと連携させることができます。

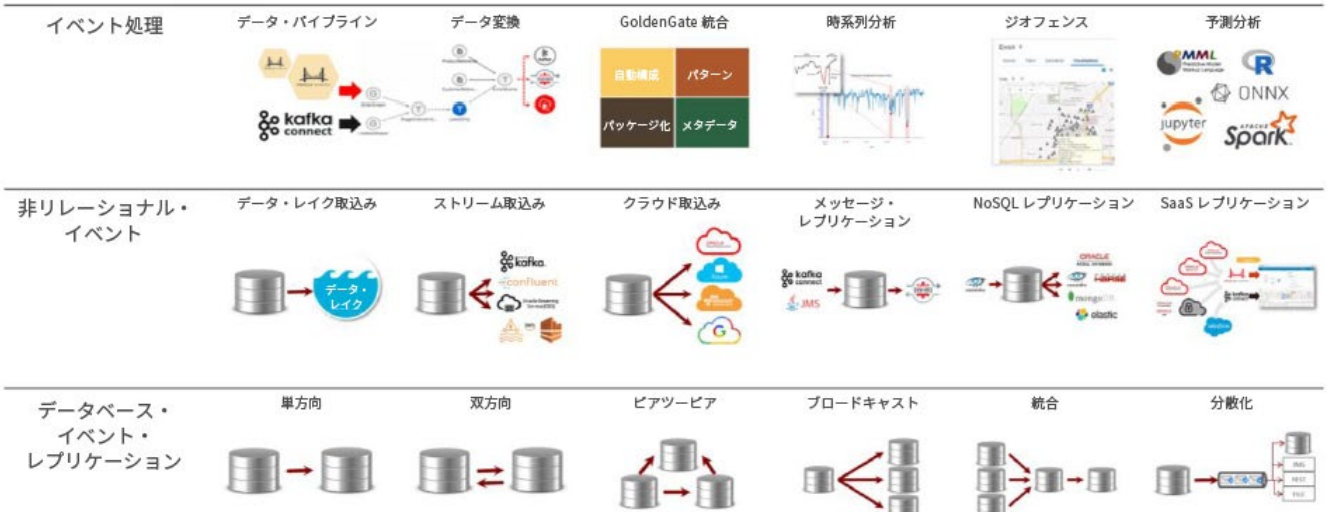


図32：Oracle GoldenGateプラットフォームを使用した、さまざまなデータ・プロダクト・ソリューションの実現

DNデータベース・イベント・レプリケーションは、長い歴史を持つGoldenGateの機能であり、数千もの顧客がこのような種類のデータ・ファブリックでGoldenGateを使用しています。データベースの同期、マルチアクティブな高可用性（HA）、およびデータウェアハウスへのリアルタイム取込みにおける一般的なユースケースは、GoldenGateデータ・ファブリックの基本のユースケースです。GoldenGateマイクロサービス・エディションにすでに移行しており、データベース・イベント・レプリケーションのテクノロジーを使用している顧客は、信頼性のあるデータ・メッシュに移行する準備がすでに整っています。

非リレーショナル・データのレプリケーションでは、ポリグロット・ストリーム・サービスのバックボーンが提供されるため、データ・ペイロードをXMLや、JSON、Avroのように取得および移動できます。この領域の中心的なユースケースでは、たとえばデータ・イベントをデータ・レイク、クラウド・ストレージ、Apache Kafkaなどのメッセージング・プラットフォームに簡単に取り込むことができます。他のユースケースはより複雑であり、幅広いプラットフォームを活用して、NoSQLストアやKafkaから、または動作保証されたSaaSクラウド・アプリケーションから直接イベントを取得します。信頼性のあるポリグロット・イベント・ストリームは、GoldenGateが実現できる中心的な機能です。

GoldenGate独自の組込みイベント処理エンジンは強力で高速ですが、顧客が他の機能にすでに投資している場合や、他の機能を選択する場合があります。GoldenGateの中核的なレプリケーション・フレームワークは、オープンソース、他のベンダー、またはパブリック・クラウド・プロバイダの非オラクル製イベント処理ツールや分析ツールと直接統合できます。

マイクロサービス・トランザクション送信ボックス、CQRS、イベント・ソーシング

マイクロサービス・アプリケーション開発者が絶えず苦悩していることの1つに、データの扱いがあります[liii]。あるマイクロサービスの定則では、開発者はすべてのデータ・ロジックをマイクロサービスのコンポーネント層に配置すべきであると定めています。この一連の考え方は、下層のデータベースとの結合を最小限に抑えることを目的としています[liii]。このアプローチが原因で、多くの熟練の設計は、データ・トランザクションを調整して整合性を確保するタスクを、実質的に直接アプリケーション開発者の手に委ねています。無数の設計パターンがこのアプローチをサポートしており、そのうちの大部分では、アプリケーションが‘結果整合性’のデータ・ポリシーに逆戻りすることが求められます。共有のデータ・ストアが存在せず、整合性/ロールバックは開発者自身によって処理されるためです。

トランザクション送信ボックスは、通信を簡素化し、より優れたデータ整合性を実現するために新たに出現した設計パターンです。元の形態では[liv]、開発者はアプリケーションと、場合によっては完全に独立した‘メッセージ中継’サービスにおいて、2つのアップデートを記述する必要があります。しかしながら、このパターンと、チェンジ・データ・キャプチャ（CDC）や、Oracle GoldenGateなどのレプリケーション・ツールを容易に組み合わせることで、パターンが極めて簡素化されることに多くの人が気付きました。CDCを活用することで、元表とフィルタリングされた任意の列は自然に‘送信ボックス’になり、メッセージ中継は最適なレプリケーション・フレームワークとなります。この方法では、マイクロサービス・イベント（データ・ストアのコミット・イベントなど）を、ブローカやデータ・レイクに、または他の任意のマイクロサービスに直接伝播できます。

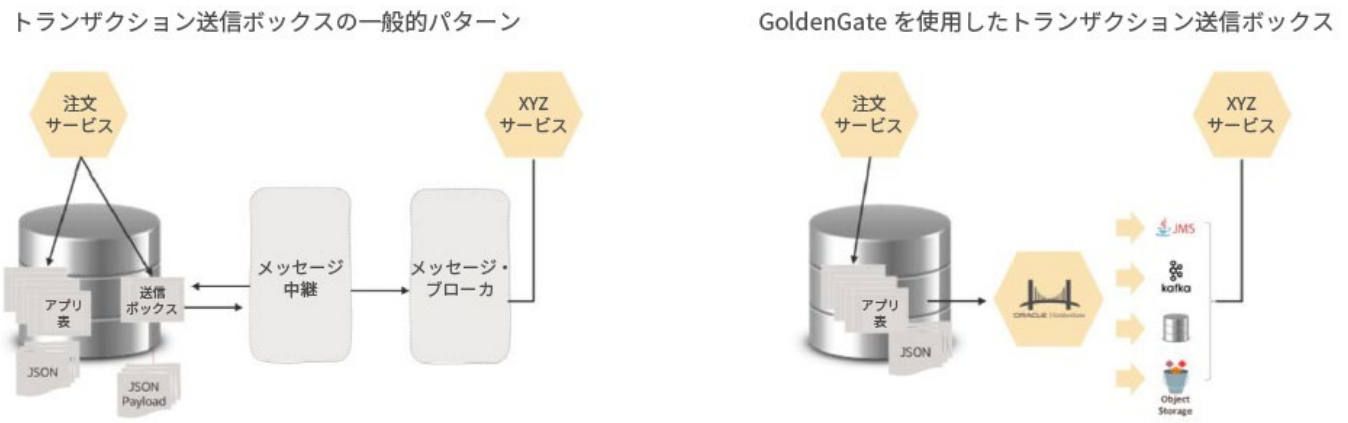


図33：Oracle GoldenGateを使用したトランザクション送信ボックス・パターンの実装

CDCとデータ・レプリケーションをマイクロサービスとともに使用すると、複雑性が大幅に低減されます。これは、分散アーキテクチャで信頼できる整合性とデータの正確性を実現する極めて素晴らしい方法です。変更検出ツールを使用しない場合は、従来のように何らかのポーリング・メカニズムをマイクロサービス・アプリケーションに実装する必要があります。ポーリングには、データ・ストアが変更を探す問合せによって常に飽和状態になるなど、多くの欠点があります。ポーリングのもう1つの大きな欠点は、リアルタイムではなく、むしろ、数分ごとに（または設定されたポーリング頻度で）変更を収集するマイクロバッチ・ソリューションであるという点です。一方、CDCツールはコミット・イベントによって駆動され、通常はデータ・ストアでのオーバーヘッドは非常に少なくてすみます（実際、データベース・ストアで追加のオーバーヘッドを排除する技術が一般的に存在します）。

CDCツールを使用したトランザクション送信ボックスのアプローチに否定的な人は、CDCツールとデータベースの緊密な結合を快く思わないかもしれませんが、これは、完全に整合性のとれたデータが取得およびレプリケートされることを保証できるというもっとも重要な利点であり、CDCレイヤーでの緊密な結合は十分に価値のあるトレードオフです。

トランザクション送信ボックス・パターンを、CQRS（コマンド・クエリー責務分離）[lv]といったその他の一般的なマイクロサービス・データ層パターンやイベント・ソーシング[lvi]に拡張すると、開発者の複雑性が低減され、優れたデータ整合性が保証されるなど、同様の利点がもたらされます。

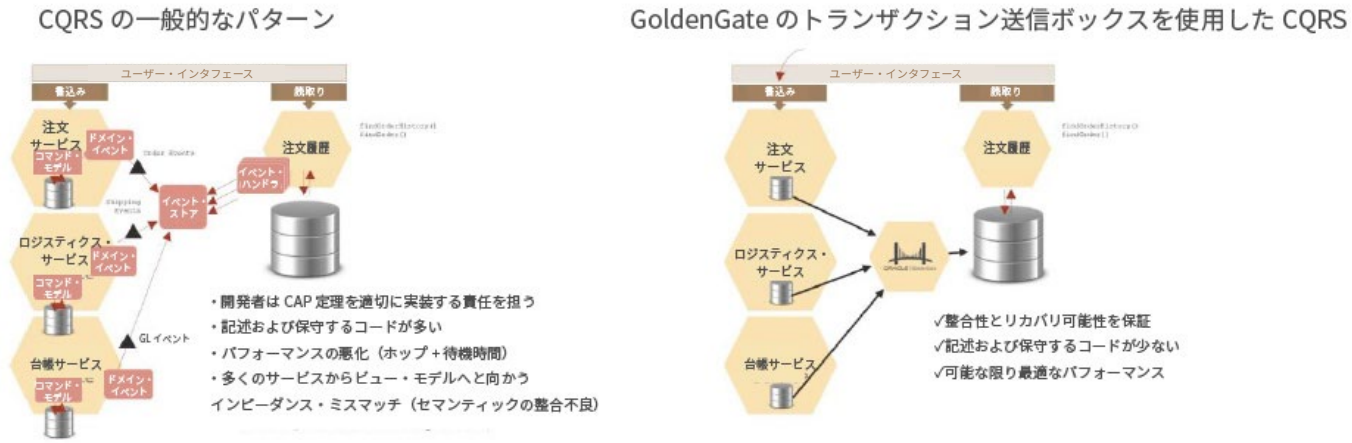


図34：Oracle GoldenGateトランザクション送信ボックスを使用したCQRSパターンの実装

イベント・ソーシング

イベント・ソーシングのテーマは複雑です。全体的なパターンについて記述された書籍は数多く存在しますが、イベント・ソーシングがマイクロサービス設計で使用されるのは、一般的に次の4つの方法です。

図35：イベント・ソーシング・パターンがマイクロサービス・アーキテクチャで使用される異なる方法

この図から分かるように、マイクロサービス開発者がイベント・ソーシングを使用する方法はさまざまです。しかしながら、少し離れて広いアーキテクチャ上の視点からイベント・ソーシングを見ると、本書の前半の「エンタープライズ・データ台帳」のセクションで説明した同種のパターンを反映しています。そのコンテキストでは、マイクロサービスは、さまざまなエンタープライズ・システムの中の、専門化された一種のイベント・プロデューサです。

エンタープライズ・データ・ファブリックの範囲には、定義上、あらゆるエンタープライズ・データ・プロデューサが含まれます。動的なデータ・ファブリックでは、マイクロサービス・アプリケーション層は完全にサポートされる必要があり、ファイブリック内でサポートされる最重要イベント・プロデューサでなければなりません。本書では、単一のイベント台帳がすべての用途を満たす必要はなく、恐らくそのようなイベント台帳は存在しないことを複数のセクションで述べています。したがって、マイクロサービス・アプリケーション向けに最適化された特別なイベント・ストアを必要とする特定のマイクロサービス・アプリケーション（そのうちの多くは、CQRSパターンなどをネイティブに実装しています）に固有の要件が存在することが時折あるかもしれません。

しかしながら、汎用性のある動的なデータ・ファブリックまたは信頼性のあるデータ・メッシュは、さまざまな台帳テクノロジーに対応して、信頼性と整合性のある適切なデータをサポートするパターンの使用を促進する必要があります。Oracle GoldenGateとともにトランザクション送信ボックス・パターンを使用することが、マイクロサービスにとって効果を発揮する選択肢であるのはそのためです。そうすることで、マイクロサービス・テクノロジー・スタックとエンタープライズ・データ・ファブリックを連携し、中核となるマイクロサービス・アプリケーションの開発を簡素化し、分散化されたデータ・トランザクションのシステム・レベルのデータ整合性を大幅に向上できます。Oracle GoldenGate自体がひとまとまりのマイクロサービスであるため、同じDevOpsおよびCI/CDのベスト・プラクティスをGoldenGateコンポーネントに適用できます。これにより、データ整合性と高い俊敏性という双方の長所がもたらされます。

ワールドクラスのストリーム処理

ストリーム処理は、エンタープライズ・ソフトウェアの一種として、数十年前から存在します。しかしながら、ようやく2015年以降、オープンソース・ビッグ・データ・プラットフォームの広範な可用性によって、イベント・ストリーム処理は、主要なIT組織において実用的なものになりました。ストリーム処理のユースケースは広範で多様です。

図36：イベント・ストリーム処理の一般的なユースケース

ただし、イベント・ストリーム処理の効果は、これらのユースケースごとの例と比較して、より普遍的と感じられる傾向にあります。さらに広い範囲で見ると、ストリームは、クラウド、ビッグ・データ、データ管理をすべて含めた数十億ドル規模の市場状況を永遠に変えるでしょう。サービス・メッシュとサーバーレス・クラウド製品をすでに形作っているのは、実現化テクノロジーの基盤であるストリームです。同様に、ビッグ・データとデータ管理をすべて網羅するプラクティスは、開発者がデータ・ストリームとイベント・ログを直接操作できるようにする方向へと急速に転換しています。わずか5年先を考えても、イベント・ストリームがその中心にないクラウド、ビッグ・データ、およびデータ管理のソリューションを想像することは困難です。

Continuous Query Language（CQL）を使用したスタンフォード大学とのコラボレーション

イベント処理に関するオラクルの道のりは、1977年にまで遡ります。当時、もっとも初期のバージョンのOracle Databaseさえも、‘オンラインREDOログ’と呼ばれるトランザクション・イベントによって一元的に定義されていました。しかし、データベースは通常、イベント・ログをエンジンの内部アーティファクトとして扱います。エンドユーザーが直接操作できるようには設計されていません。2000年代半ばには、オラクルはスタンフォード大学とのコラボレーションを開始し、イベント・ストリームへの問合せを行うためのSQLベースの提案（Continuous Query Language（CQL））を最終的に作成しました。この記事[lvi]は、宣言型プログラミング・モデルの基本原理と、ストリーム・データとさまざまな種類のウィンドウイング機能の操作に必要な多数の計算セマンティックについて述べています。

Oracle Complex Event Processing（Oracle CEP）

スタンフォード大学とのコラボレーションの重要な成果の1つは、最初のOracle Complex Event Processing（Oracle CEP）エンジンが作成されたことです。Oracle CEPエンジンは、CQL宣言型言語によって定義された概念を実装し、その時代のオラクルのミドルウェア・テクノロジー・スタック（BEA WebLogicアプリケーション・サーバーおよびOracle Coherence for Javaのデータ・グリッド）にエンジンを体現しました。このテクノロジー・スタックを使用して、Oracle CEPエンジンは、2010年代の単一Exalogicノードで、1秒あたり100万を超えるイベントをスケーリングできました。さらに印象的なのは、コア・エンジンとCQLセマンティックが絶えず進化し、時系列、地理空間イベント、および複雑な複数ストリームの相関関係向けの、非常に機能が豊富なデータ処理セマンティック一式をサポートするようになったことです。

オープンソース・プラットフォームへの移行

2015年までに、オラクルはビッグ・データのイベント・ストリーム処理で起きた変化に気付き、Oracle CEPエンジンをApacheオープンソース・フレームワークと、そして最終的にはOracle GoldenGateと連携させることを決めました。

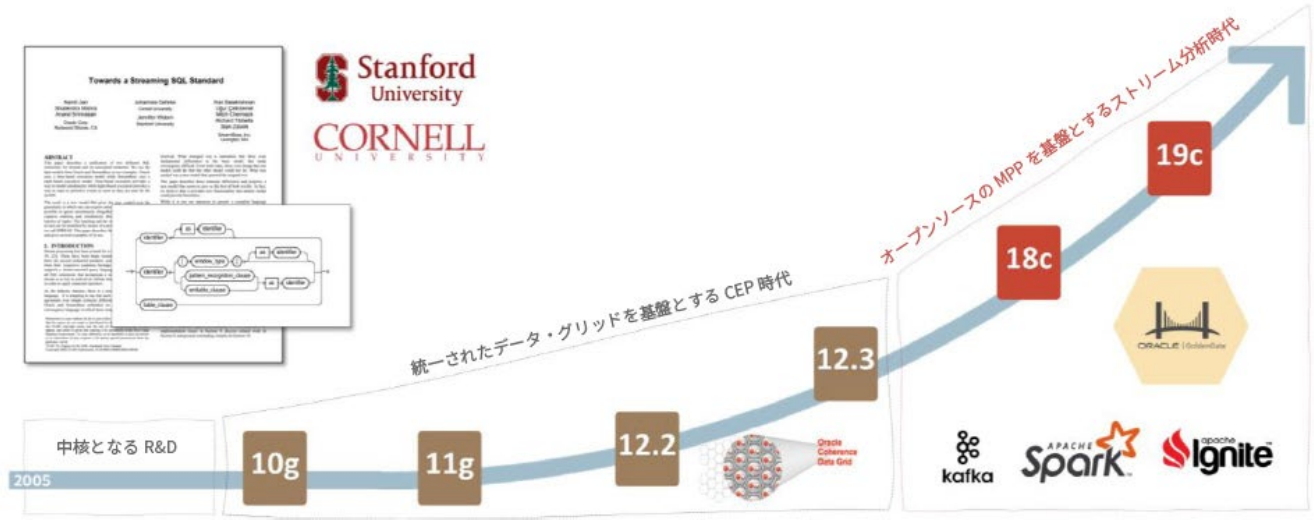


図37：オラクルのストリーム処理コア・エンジンの開発における3つのおもな時代

Oracle CEPエンジンをリファクタリングするための初期の調査では、さまざまなApacheテクノロジーが検討されました。Apache Stormは、強力で定評のある（Twitter規模）オプションですが、2015/16年にはすでに古いと捉えられていました。この時期、Apache Flinkはドイツの大学組織に起源を持つ無名のプロジェクトでした。Apache Sparkはすでに大成功を収めており、Sparkのストリーム・フレームワークは当時は（現在も依然として）脆弱であったものの、Oracle CEPエンジンがストリーム・セマンティックのパワーを実際に提供することになっていたため、問題ではありませんでした。最終的に、オラクルは（スタンフォード大学とともに初めて探究された基盤となるCQLセマンティックを実装した）Oracle CEPエンジンのホストとして、SparkのMPP基盤を利用することを決定しました。現在、GoldenGate Stream Analyticsパイプラインは、どのようなApache Sparkフレームワークでも実行でき、コア・エンジンは、ストリーム処理向けの独自の強力なセマンティックを実装しています。

		オープンソースの一般的な代替品		
[最適] ●●●⊙○ [不適]	GoldenGate Streaming	Spark Streaming	Apache Flink / Beam	Confluent KSQL
ユーザー・エクスペリエンス				
ローコード開発（組込みパターン / アクセラレータ）	●	○	⊙	⊙
インタラクティブ / ライブ編集（ブラウザ・ベース、即座のビュー変更）	●	○	●	⊙
組込みのライブ・ダッシュボード（イベント駆動型チャート / グラフ）	●	⊙	⊙	○
コア・ストリーム・セマンティック				
計算対象（変換、結合、フラット化、ステートフルなど）	●	◎	◎	◎
時間枠（グローバル、固定、スライディング、タンブリング、カスタムなど）	●	◎	●	●
処理時間中（トリガーイベント、時間、数、タイマーなど）	●	⊙	◎	⊙
改良を結び付ける方法（破棄、蓄積、取消し）	◎	◎	◎	⊙
分析				
堅牢な CEP 機能（複合イベントの相関付け、ネイティブなタイム・レコーダ）	●	○	⊙	⊙
ジオフェンスおよび地理空間（緯度 / 経度、組込みマップ、カスタム・マップ・タイルなど）	●	○	○	○
機械学習（ネイティブなスケーラビリティ、ONNX、PMML、python のサポートなど）	◎	◎	◎	⊙
時系列分析（組込みの間隔パターン、しきい値処理など）	◎	⊙	⊙	⊙
その他の機能				
バックプレッシャー（パイプラインごとの動的な取込み）	自動	カスタム	カスタム	カスタム
状態管理（ストリームとネイティブ・キャッシュ全体の自動化）	データ・グリッド	カスタム	RocksDB	RocksDB
データ整合性（OLTP 変更イベント、挿入 / 更新 / 削除）	自動	カスタム	カスタム	カスタム
GoldenGate ストリーム・タイプ（SCN/CSN、トランザクション、順序などを認識）	ネイティブ	カスタム	カスタム	カスタム

図38：GoldenGate Stream Analyticsと一部の一般的なオープンソース・ストリーム・プラットフォームの大まかな比較

Oracle CEPエンジンがApache Spark内で実行されるようにリファクタリングされたのと同じ時期に、ITの世界では別のマクロトレンドとしてローコード開発が普及していました。そのため、開発者を下層のプログラミング（Java/ScalaおよびCQL）からさらに排除し、ストリーム処理パイプラインおよび分析のポイントアンドクリック開発に対応した（Webブラウザで実行される）グラフィカルなユーザー・インタフェースを提供する決定が下されました。この時期に、オラクルは開発者の体験が活発化されるように設計された豊富なパターン・アクセラレーター式もサポートし始めました。現在は、Oracle GoldenGate Stream Analyticsは紛れもなく、もっとも簡単に使用できるストリーム・ツールの1つです。事前構築済みのサンプル・アプリケーションが組み込まれており、Oracle Cloudから5分未満で起動できます。

オラクルは、使いやすく設計され、エンタープライズ・データを扱うための信頼性を備え、複雑なストリーム・データを大規模に処理する非常に強力な基盤となるコア・エンジンを搭載したワールドクラスのストリーム・プラットフォームに尽力しています。Oracle Cloudのお客様は、ノーコードGUIのみを使用してストリーム・アプリケーションを開発できるため、可能な限りシームレスでサーバーレスのエンドユーザー・エクスペリエンスが実現します。オンプレミスのお客様には、高度なモジュール式のGoldenGateマイクロサービスとともに、オープンソースのApacheフレームワーク内にストリームをデプロイするオプションが提供されます。

Oracle Cloudを使用した動的なデータ・ファブリックと信頼性のあるデータ・メッシュ

本書の最初のセクションで説明したように、広義のデータ・ファブリックには、オラクルの他のテクノロジーやサービスも含まれます。オラクルは、広範なデータ・ファブリック・エコシステム全体で、唯一のリーダーと認識されており[lviii]、次の完全なポートフォリオを提供しています。

クラウドネイティブな一般的なプラットフォーム・データ・ファブリック	マルチクラウドとオンプレミス向けの最善のデータ・ファブリック
• 分析と自律型DB向けのセルフサービスETL • OCI Data Catalog、OCI Data Integration、OCI Data Flow • OCI GoldenGateとOCI向けOracle Stream Analytics • Oracle Integration CloudとOracle Cloud SQL	• Oracle Data Integrator（ETL、品質、メッセージングを提供） • Oracle GoldenGateとStream Analytics • Oracle Big Data SQL（データ連携） • Oracle Data Visualization（データ準備）

本書の主要目的は、リアルタイム、イベント駆動、およびストリーム・データの各ユースケースに主に重点を置いた'動的な'データ・ファブリックを中心としています。すでにデータ・メッシュの定義において重点がイベント・パイプラインに絞られているため、オラクルがさらに重きを置く領域は、'信頼性のあるデータ'です。これには、データ・ガバナンスのほか、ACIDレベルのデータ整合性に対するトランザクション保証といった複数の側面が含まれます。

本書でこれまで説明したテクノロジーはすべて、顧客が提供するインフラストラクチャ、Oracle Cloud以外のパブリック・クラウド、およびDockerやKubernetesなどのサービス・メッシュ・テクノロジーで幅広く実行できますが、ようやくここで、Oracle Cloud Infrastructure（OCI）固有のサービスと機能に目を向けることができます。世界の25箇所以上のOCIデータセンターのいずれかで、完全な規模のデータ・ファブリックまたはデータ・メッシュを運用するにはどうしたらいいでしょうか。

Oracle Cloud Infrastructure（OCI）のデータ・ファブリック製品、サービス、機能	
データの準備	Oracle Analyticsに付属する、自然言語処理（NLP）を活用するAIベースの革新的なエンジンと、機械援助型データ収集のためのグラフ・ナレッジベース（リンク付きオープン・データ）
自律型データ・ツール	緊密に統合され、Autonomous Data Warehouseとバンドルされた機能一式。受賞歴のある、バッチ・データ変換用のOracle Data Integrator E-LTツールも含まれます
OCI Data Integration	ローコード・ユーザー・インタフェースと高度なワークロード・オプティマイザを備えたサーバーレスETL機能
OCI Data Flow	OCI Data Integrationと緊密に統合された、サーバーレスApache Spark実行
OCI Streaming Service	ネイティブなイベント・メッセージング・フレームワーク。Oracle Cloud全域でKafkaに似た機能を提供します
OCI GoldenGate	Oracle Cloudのネイティブなフルマネージド型GoldenGate。OCIのネイティブ・サービスに固有の自動スケーリング機能とその他多数の高度な自動化機能を提供します
GoldenGate Stream Analytics	GoldenGate Stream Analyticsの機能はすべて、時間あたりのOCIクラウド・クレジットで利用できます。必要なすべてのエンジン・コンポーネントとともに事前にパッケージ化されており、どのようなSparkノードでもワークロードを実行できるだけの十分な柔軟性を備えています
Integration Cloud	業界をリードするアプリケーション統合とビジネス・プロセス自動化機能。あらゆる種類のSaaSおよびソーシャル・メディア・データ・イベントと統合するための、データ・ファブリックの理想的なコンポーネントです
IoT Cloud	スマートな製造、予測メンテナンス、およびロジスティクス向けの産業用IoTソリューション。エッジ・デバイスやIoTゲートウェイと統合し、'デジタル・ツイン'を管理するための、データ・ファブリックの理想的なコンポーネントです
OCI Data Catalog	ガバナンス全般とビジネス・データ・ドメインの管理に不可欠なOCIサービス

上述の各サービスでは、同じOCIコントロール・プレーンと、Oracle Cloudを包括的なプラットフォームとして機能させることができる数多くのプラットフォーム・サービス機能が共有されます。

大まかな青写真

Oracle Cloudで運用する場合、特定のユースケースで適切なサービスを選択できるよう顧客を導くための青写真となるリファレンス・アーキテクチャを利用できます。これらのアーキテクチャでは、役立つベスト・プラクティスも提供されます。OCI内でソリューションを実現できるよう顧客を支援するTerraformの自動化が提供される場合もあります。データ・ファブリックを図で説明するために、大まかな青写真をここで示します。最新のリファレンス・アーキテクチャをすべて参照するには、OCIアーキテクチャ・センター（https://www.oracle.com/jp/cloud/architecture-center/）にアクセスしてください。

業務系データ向けの動的なデータ・ファブリックと信頼性のあるデータ・メッシュ

本書の「運用データ・ストアと分析データ・ストアの連携」セクションで先ほど説明したように、データ・ファブリックまたは信頼性のあるデータ・メッシュにおける業務系データのユースケースには以下が含まれます。

- **オンライン・データベース移行** – オンライン・アプリケーションの中断を回避
- **データベースの高可用性（HA）とディザスタ・リカバリ（DR）** – すべてのリージョンで稼働時間を最大化
- **参照データ/マスター・データの同期** – ルックアップ表とゴールデン・レコードの同期を完全に維持
- **ポリグロット・データ統合** – 異なるOLTPデータベースやマイクロサービス（CQRSなど）でデータを同期

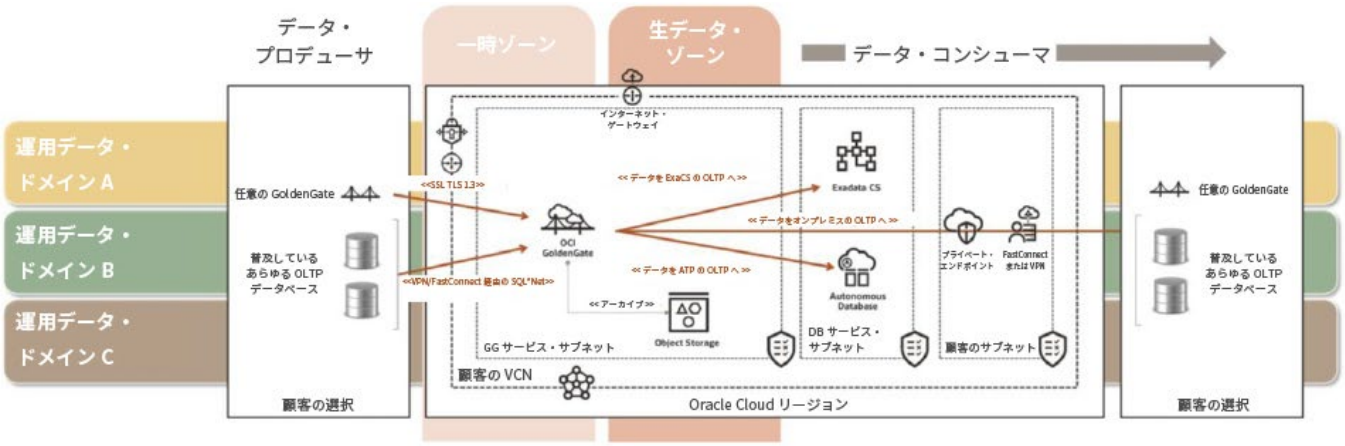


図39：オンプレミスのGoldenGateマイクロサービスと業務系データ・ストアを使用したメッシュのOCI GoldenGate

上記の青写真には、複数の異なるソースおよびターゲットと統合された（独自の専用サービス・サブネット内の）OCI GoldenGateサービスが示されています。OCI GoldenGateは、（オンプレミスまたは他のクラウドで実行される）別の任意のGoldenGateデプロイメントや、数百ものさまざまなソース・テクノロジーを組み合わせたものと直接統合できます。この統合により、リアルタイムのデータベース・トランザクションを、OCI GoldenGateデプロイメント経由で、下流のデータ・コンシューマ（Exadata Cloud Service、Autonomous Database、またはサポートされる任意の数の外部データ・ストア、ビッグ・データ、NoSQLエンジン、メッセージング・プラットフォームなど）に継続的に送信することができます。この図のユースケースは実際に運用可能であり、一時ゾーンに配置された永続ストレージには、恐らくわずか数時間/数日分のログ台帳が記録されます。生データ・ゾーンでは、OCI GoldenGateサービスは、ガバナンス、監査、またはコンプライアンス用の長期ストレージを必要とする顧客のために証跡データをアーカイブできます。GoldenGateは、任意の業務系データ・ドメインからデータ・トランザクションをレプリケートでき、‘データ・プロダクト思考’のガイドラインで定義されたベスト・プラクティスに応じて、特定のGoldenGateデプロイメントを名前付きデータ・ドメインに割り当てることができます。これにより、サービスがドメインごとにパーティションに機能分割されます（Oracle Cloudでは、これを行う1つの方法として、OCIコンパートメントが使用されます[lix]）。

分析データ向けの動的なデータ・ファブリックと信頼性のあるデータ・メッシュ

本書の大部分では、データ・ファブリックまたは信頼性のあるデータ・メッシュにおける分析ユースケースのサポートに重点を置いています。以下は、主要なユースケースの要約です。

- **ストリーム分析/アプリケーション** – 通常は意思決定支援および自動化向け（次善のアクションなど）
- **データウェアハウスおよびデータ・マート** – ストリーム・ユースケースではデータ取込みとCTLサービスを提供
- **データ・レイクおよびデータ・サイエンス** – 忠実性の高いシステム・イベントは、ほとんどのデータ・サイエンティストにとって極めて貴重
- **CTLおよびデータ統合** – あらゆるデータ管理ユースケース向けの連続変換およびロード

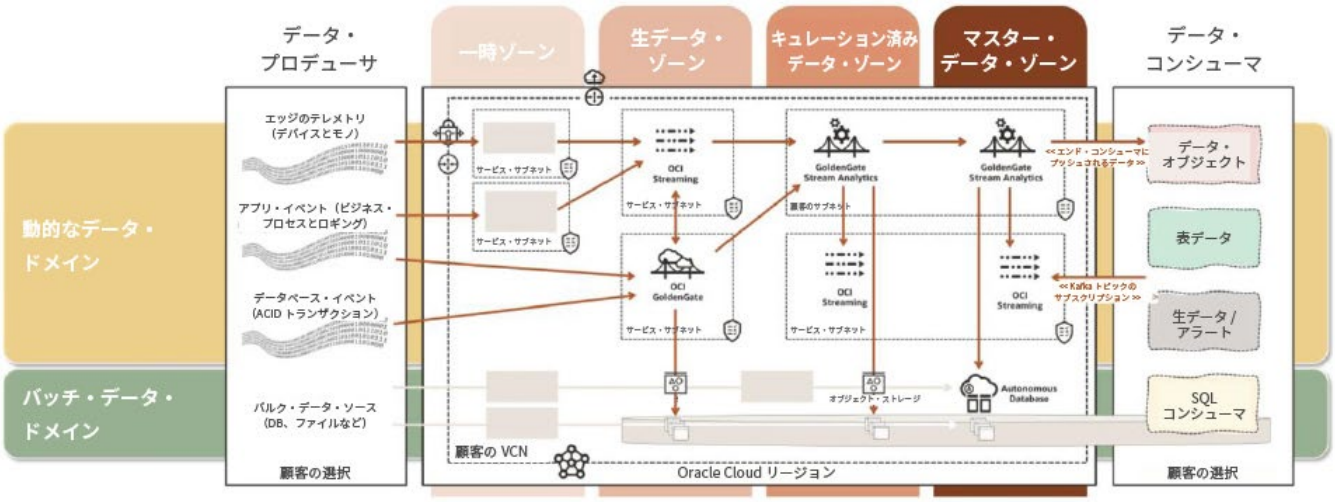


図40：ストリームとリアルタイム・サービスを重視した信頼性のあるデータ・メッシュのOCIコンポーネント（バッチ・サービスは参照用に表示）

上記の青写真には、リアルタイムでイベントを生成できるさまざまなデータ・プロデューサが（左側に）示されています。Oracle Cloudには、IoTイベント向けのOracle IoT Cloud、SaaSアプリケーションおよびソーシャル・メディア・イベント向けのOracle Integration Cloudなど、一時ゾーン（取込み）のオプションが複数存在します。生データ・ゾーンは、数日/数か月分のデータを保管できるように構成できます（エンタープライズ・データ台帳として機能）。この図には、Kafkaに似たメッセージング向けのOCI Streamingと、信頼性のあるデータ・トランザクション向けのOCI GoldenGateが表示されています。OCI StreamingとOCI GoldenGateはどちらも、GoldenGate Stream Analyticsとの統合により、データを準備してキュレーション済みデータ・ゾーンとマスター・データ・ゾーンに変換できるようになります。いずれのゾーンのデータも、Oracle Database、OCI Object Storage、またはOCI Streaming内で永続化でき、どれを使用するかの選択はユースケースに基づきます。ストリーミング・ユースケースでは、規範的なイベント（キュレーション済み、またはマスタリングされたJSONペイロードなど）は、通常はOCI Streamingの管理されたトピックに置かれます。下流のデータ・コンシューマは、OCI Streamingトピックを選択してサブスクライブできます。あるいは、GoldenGateが、データをさまざまなターゲット・システムに直接プッシュすることもできます。これにより、イベント・チェーン全体がプッシュ型になります。

上図では、動的なリアルタイム・ストリーム・コンポーネントに明確に重点が置かれていますが、完全な図を示すために、バッチ・ドメイン処理がいくつか表示されています。OCI Data Integrationは、多種多様なソース・データ・ストアのデータをバッチ処理することで、データを取得し、さまざまなバッチ・データ変換を適用できます。オプションとして、OCI Data Integrationは、OCI Data Flowサービスを使用して、Apache Sparkジョブをバッチで実行できます。データベースを中心に据えたユースケースでは、Autonomous Database（Oracle ADB）でOracle Data Integratorの使用が直接サポートされます。Oracle Data Integratorを使用すると、データを直接Oracle ADBでバッチ処理し、データベース内で一連のプッシュ・ダウンやE-LTデータ変換を適用できます。

1つの図では表示できないほど、さまざまな相互作用のモデルが数多く存在します。エンタープライズ・データ・ファブリックまたはデータ・メッシュのユースケースでは、バッチ処理フローに一部のストリーム・イベントが送信される可能性があります。
同様に一部のバッチ処理（データベースのロードなど）によって、下流のリアルタイム・データ・パイプラインがトリガーされる可能性があります。すべてのOCIツールとOCIサービスは、同じコントロール・プレーン、セキュリティ・ポリシー、およびネットワーク・インフラストラクチャから実行されるように設計されているため、これらのデータ・ファブリック/メッシュの組み合わせはすべて、実現可能です。

Oracle Cloudデータ・カタログ

データ・ファブリック、またはデータ・メッシュをOracle Cloudから実行する場合、ガバナンスを簡素化するためにデータ・ドメインをまとめるのはOCI Data Catalogです。チームは自身の'データ・プロダクト思考'の経験則に応じて、単一のデータ・カタログを使用してすべてのデータ・ドメインを管理することを目指すことができます。もしくは、ドメインごとに異なるデータ・カタログを連携したり、さまざまなベンダーやクラウド・プロバイダの多種多様なデータ・カタログをサポートしたりしなければならない可能性も高いでしょう。Oracle Cloudで実行される、またはOracle Cloudと統合されるすべてのデータ・ドメインにとって、OCI Data Catalogはビジネス・ドメインのガバナンスに理想的なツールです。

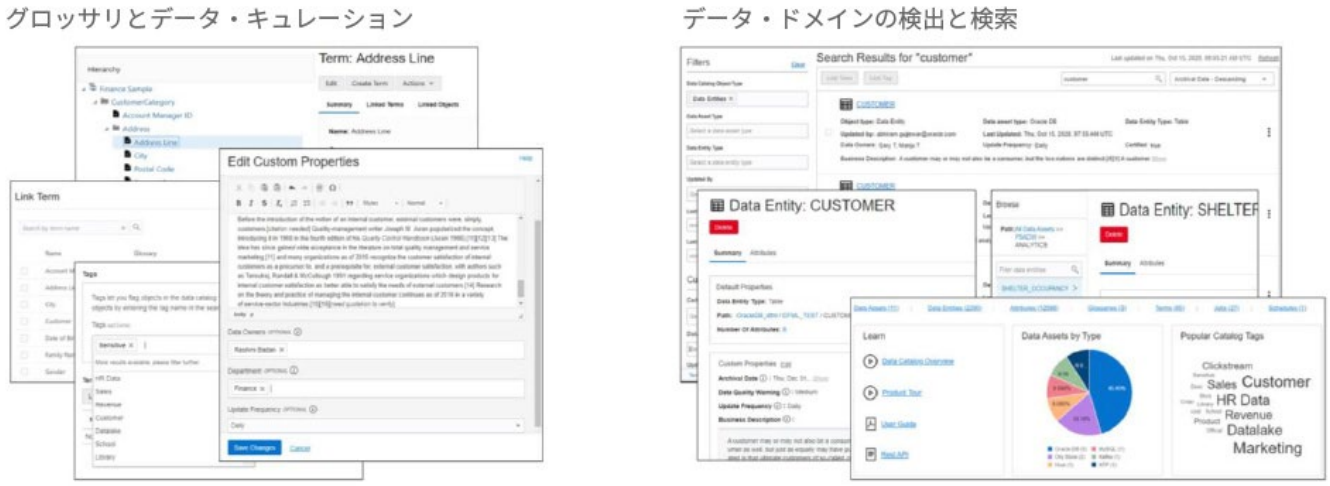


図41：OCI Data Catalogを使用した、ビジネス・データ・ドメイン、分類法、データ・ゾーンの連携

総じてOracle Cloudでは、動的なデータ・ファブリックと信頼性のあるデータ・メッシュを構築する過程にあるあらゆる組織を支援できる豊富なサービスと機能が提供されます。Oracle Cloud固有の各サービスは、運用上優れた複数の方法であらかじめ統合されており、共通のコントロール・プレーンを共有します。すべてのOCIデータ・ファブリック・サービスに適用される総合的なアプローチにより、使用した分だけ支払うだけですむ（アイドル・リソースのない）サーバーレス・インフラストラクチャの利点が多数提供されるほか、ローコードまたはノーコード機能が実現します。

結論 – 変化に備える

データ管理の世界では、あまりにも長い間、OLTPを分析ワークロードから人為的に分離してきました。古いバッチ処理方法と脆弱なウォーターフォール型のデータ管理方法は、体系的に破綻してもおかしくない状況にあります。それ以前のアプリケーション開発ソフトウェア・エコシステムと同様、データ管理ソフトウェア・エコシステムは、ビジネス主導の新たな思考と俊敏な運用に適応し、それらを採用することで、今後数十年間の成功を勝ち取る必要があります。

私たちは、サービス・メッシュ、SDN（Software Defined Network）、CTL向けのポリグロット・ストリーム、汎用スケール・アウト・プラットフォームなど、成功を実現する新たなテクノロジーを使用すべき転換点に来ています。これらの基盤を改善することにより、エンタープライズ・データ向けの新しいタイプの動的なデータ・ファブリックが可能になります。もっとも困難な部分は、相変わらず、何が可能であるかについての気持ち、考え、文化を変えることです。古い前提を改めてデータ・プロダクト思考を適用し、OLTPおよび分析データ・ドメインを統合し、バッチ優先思考を否定し、分散アーキテクチャ・パターンへの移行を採用する必要があります。

本書では初めから終わりまで、動的なデータ・ファブリックと信頼性のあるデータ・メッシュについて説明しました。これらの機能の交わりが、エンタープライズITデータ・アーキテクチャの新しいパラダイムです。この新しい概念で明確に言及した過去との意義ある違いは、以下のとおりです。

- **基本要素：データ・プロダクト思考** – 組織的に統制された、データ資産の管理方法
- **原理1：分散化されたモジュール式メッシュ** – 過去のデータ統合モノリスを否定
- **原理2：エンタープライズ・データ台帳** – 静的データではなく、データ・ログからの統合を促進
- **原理3：信頼性のあるポリグロット・データ** – あらゆるペイロード・タイプをサポートし、ストリームのACID整合性を存続

総体的に見て、これらの基本原理が交わることで、このアプローチとこれまでのアプローチが差別化されます。たとえば、データ・ファブリックの市場領域は非常に大規模（数十億ドル規模）で、データ統合とデータ管理の技法やアプローチは多種多様ですが、そのうちのほとんどは、分散化されたイベント駆動型のデータ・アーキテクチャには適用されません。同様に、考案された当初のデータ・メッシュ・テクノロジーは、マイクロサービス・イベント・ソーシング・パターンと極めて緊密に連携されており、データ・パイプラインや、信頼性のあるACIDレベルのデータ整合性を優先していません。

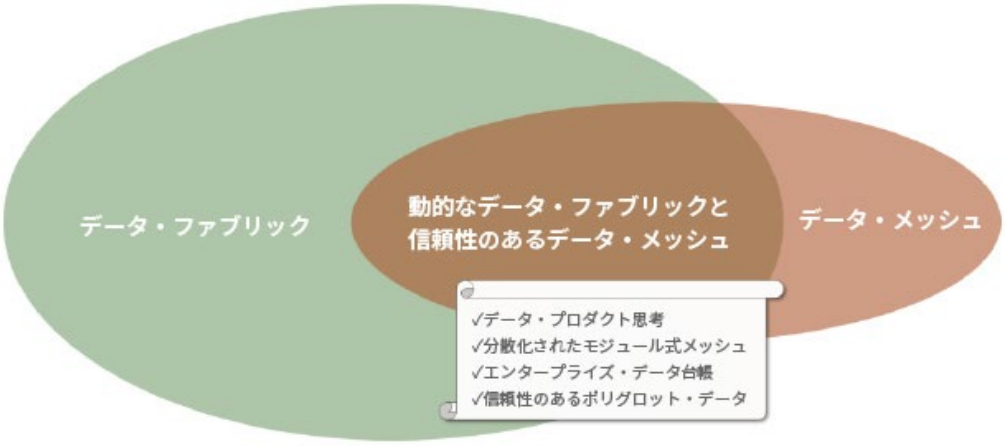


図42：動的なデータ・ファブリックと信頼性のあるデータ・メッシュ

本書では、2つの概念の融合、すなわち、（イベント駆動型で分散化された）メッシュのアーキテクチャ属性を持つ（信頼性のあるエンタープライズレベルの）データ・ファブリックについて説明しています。データ・プロダクト思考は、長い間メッシュとファブリックの双方と関連付けられてきた特性を持ちますが（データの収益化、データ・スチュワードシップ/ガバナンス、DataOpsといった別の名称ではありますが）、ここでのデータ・プロダクト思考は、アプローチ全体の中心思想へと押し上げられ、データのビジネス価値がデータ管理哲学の中心に置かれています。

オラクルは、長きにわたりリーダーとしてデータ管理（データベース・ベンダー第1位[lx]、およびGartnerのクラウド・データベースのリーダー[lxi]）とデータ統合（12年間、Gartner Magic Quadrantの‘リーダー’[lxii]）を先導しており、これら次世代の動的なデータ・ファブリックと信頼性のあるデータ・メッシュのカテゴリでも他社に先行しています。GoldenGateプラットフォームなど、オラクルのデータ管理とデータ統合テクノロジーに投資するお客様には、確実に未来へと向かう道筋が示されています。実際、動的なデータ・ファブリックと信頼性のあるデータ・メッシュの幅広い視野こそが、ストリーム・データ向けのOracle GoldenGateプラットフォーム全体への継続的な投資を導いています。

Oracle GoldenGateチームは、顧客がデータから価値を生成できるよう支援するツールとテクノロジーの提供に尽力しており、これは、“*人々が新たな方法でデータを参照し、インサイトを発見し、無限の可能性を解き放つことができるよう支援する*”というオラクルの総合的な使命と一致します。現在、Oracle GoldenGateプラットフォームにより、顧客は次のような革新的なデータ・プロダクトを生成できます。

- **キュレーション済み変更ストリーム** – データ・プロダクトの所有者は、変更されたデータのAPIを外部のコンシューマに公開可能
- **ローコード・データ・パイプライン** – 機能が豊富なデータ・フローと変換のための視覚的な管理エディタ
- **時系列分析と空間分析** – 移動中のあらゆるデータを分析するための使いやすいパターン
- **リアルタイムのダッシュボードとアラート** – 継続的に更新されるグラフと可視化を迅速に作成
- **ストリーム・データ・サービス** – 変更されたデータをデータベース、NoSQL、または任意のデータ・レイク・プラットフォームにリアルタイムにプッシュ
- **本番環境のMLスコアリングとアラート** – 精密なスコアリング・モデルをリアルタイム・ストリームに取り込む

GoldenGateプラットフォームは、次のような他のテクノロジー経由で消費されるデータ主導のソリューションとデータ・プロダクトを実現するためにも広く使用されています。

- **同期式OLTPデータベース** – 停止時間の短い移行、高可用性（HA）およびリカバリ向け
- **マルチアクティブ・データベース** – 地理データのシャーディングとリージョン間のデータ・レプリケーション向け
- **従来型データウェアハウス** – ‘トリクル・フィード’またはCTL（連続変換およびロード）を実現する手段として
- **データ・レイク取込み** – クラウドやビッグ・データ・テクノロジーへの、既存システムの機能を妨げることのないリアルタイムのデータ・ステージング
- **マイクロサービス送信ボックス** – アプリケーション・イベントをレプリケートしながらデータ整合性を維持するパターン
- **ストリーム・プロセッサ** – エッジ、ゲートウェイ、ストリーム・アプリケーション向けの組込みストリーム・ソリューションとして

高価値のデータ・プロダクトに置かれたオラクルの主眼と、GoldenGateから導かれるビジネス成果は、未来を定義付ける最新の分散データ・アーキテクチャ向けの真のマルチクラウド・ソリューションとして業界をリードしています。

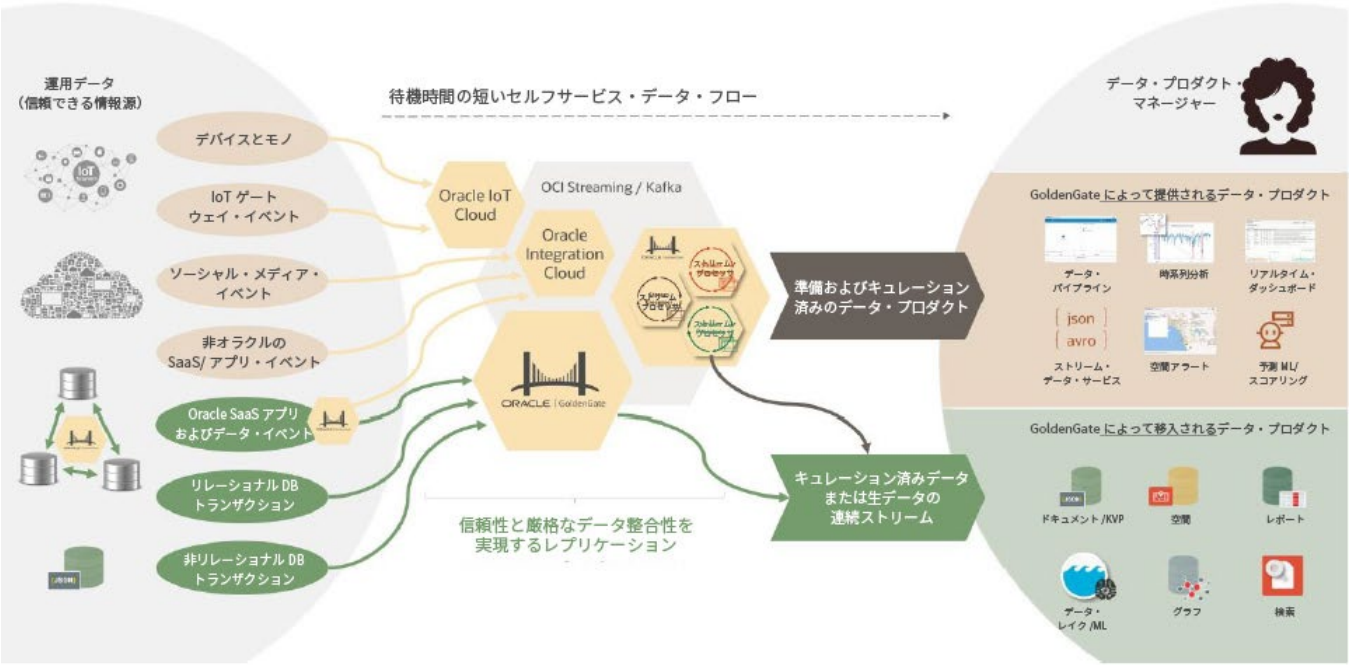


図43：Oracle IoT CloudとOracle Integration Cloudを使用したメッシュにおける、GoldenGateを使用した最新の動的なデータ・ファブリックおよび信頼性のあるデータ・メッシュ

信頼性のあるリアルタイム・データを中心とするGoldenGateの既存機能とその最新のマイクロサービス/クラウド・アーキテクチャは、次世代の動的なデータ・ファブリックと信頼性のあるデータ・メッシュの理想的な基盤となっています。データ・プロダクト思考によって刺激され、お使いのデータ・アーキテクチャを（i）分散化されたモジュール式メッシュにし、（ii）信頼性のあるポリグロット・データ・ストリームで、（iii）統合のためのイベント台帳を活用することを目指すお客様は、Oracle GoldenGateプラットフォームとオラクルのデータ・ファブリック・ポートフォリオの全サービスをご検討ください。

参考資料

[i] Forresterブログ、『Information Fabric Delivers Next Generation of Data Virtualization』（Noel Yuhanna著、2013年8月）
https://go.forrester.com/blogs/13-08-15- information_fabric_30_delivers_the_next_generation_of_data_virtualization/

[ii] Gartner、『Emerging Technologies:Data Fabric Is the Future of Data Management』（Sharat Menon、Ehtisham Zaidi、Mark Beyer共著、2020年12月4日）https://www.gartner.com/en/documents/3994025

[iii] Forrester、『The Forrester Wave™:Enterprise Data Fabric, Q2 2020』（Noel Yuhanna著、2020年6月）
https://www.forrester.com/report/The-Forrester-Wave-Enterprise-Data-Fabric-Q2-2020/RES157288

[iv] Forrester、『The Forrester Wave™:Enterprise Data Fabric, Q2 2020』（Noel Yuhanna著、2020年6月）
https://www.forrester.com/report/The+Forrester+Wave+Enterprise+Data+Fabric+Q2+2020/-/E-RES157288

[v] https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2818.1994.tb03441.x

[vi] https://gcn.com/articles/2000/06/07/heres-all-you-need-to-know-on-tcpip.aspx

[vii] https://cacm.acm.org/magazines/2009/12/52840-a-smart-cyberinfrastructure-for-research/fulltext

[viii] https://communitywiki.org/wiki/DataMesh

[ix] Gartner、『Maverick* Research:Revolutionizing Data Management and Integration With Data Mesh Networks』（Mark Beyer、Guido De Simoni、Ehtisham Zaidi共著、2016年11月4日）
https://www.gartner.com/en/documents/3500835/maverick-research-revolutionizing-data-management-and-in

[x] https://martinfowler.com/articles/data-monolith-to-mesh.html

[xi] https://ja.wikipedia.org/wiki/パラダイムシフト

[xii] https://en.wikipedia.org/wiki/Cruft

[xiii] https://www.gartner.com/smarterwithgartner/use-a-hybrid-integration-approach-to-empower-digital-transformation/

[xiv] *本書で説明する方法とツールを採用した顧客との実際の会話から抽出したデータ*

[xv] https://ja.wikipedia.org/wiki/デザイン思考

[xvi] https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying

[xvii] https://queue.acm.org/detail.cfm?id=3321612

[xviii] Gartner、『Market Share Analysis:Data Integration Tools, Worldwide, 2019』（Sharat Menon著、2020年7月13日）
https://www.gartner.com/en/documents/3987502/market-share-analysis-data-integration-tools-worldwide-2

[xix] https://ja.wikipedia.org/wiki/デザイン思考

[xx] https://hbr.org/2018/10/how-to-build-great-data-products

[xxi] https://uxplanet.org/product-thinking-101-1d71a0784f60

[xxii] https://hbr.org/2016/09/know-your-customers-jobs-to-be-done

[xxiii] https://hbswk.hbs.edu/item/clay-christensen-the-theory-of-jobs-to-be-done

[xxiv] https://medium.com/rapido-labs/the-data-pm-and-his-experiments-8f38bcf4ae9b

[xxv] https://www.oracle.com/jp/a/tech/docs/maximum-availability-wp-19c-ja.pdf

[xxvi] https://en.wikipedia.org/wiki/Functional_decomposition

[xxvii] https://en.wikipedia.org/wiki/Decomposition_(computer_science)

[xxviii] https://ja.wikipedia.org/wiki/オブジェクト指向プログラミング

[xxix] https://chrisrichardson.net/post/refactoring/2019/10/09/refactoring-to-microservices.html

[xxx] https://en.wikipedia.org/wiki/Third_normal_form

[xxxi] https://www.oreilly.com/library/view/domain-driven-design-tackling/0321125215/

[xxxii] https://www.informit.com/articles/article.aspx?p=2023702

[xxxiii] https://en.wikipedia.org/wiki/Design_Patterns

[xxxiv] https://medium.com/@mark.tesla2/elon-musk-and-first-principles-thinking-e0035730d7

[xxxv] https://www.foreignaffairs.com/articles/2015-12-12/fourth-industrial-revolution

[xxxvi] https://www.businessinsider.com/what-is-davos-world-economic-forum-conference-2020-1

[xxxvii] https://ja.wikipedia.org/wiki/イノベーションのジレンマ

[xxxviii] https://ja.wikipedia.org/wiki/DevOps

[xxxix] https://www.devopsdigest.com/the-state-of-database-deployments-in-application-delivery-2019

[xl] https://ja.wikipedia.org/wiki/インピーダンスミスマッチ

[xli] https://www.ieee.org/about/news/2020/survey-chief-information-officers-and-chief-technology-officers.html

[xlii] https://dataprivacymanager.net/5-biggest-gdpr-fines-so-far-2020/

[xliii] https://www.iso.org/standard/72437.html

[xliv] https://ja.wikipedia.org/wiki/多層防御_(セキュリティ)

[xlv] https://ja.wikipedia.org/wiki/OSI参照モデル

[xlvi] https://en.wikipedia.org/wiki/Mutual_authentication

[xlvii] https://oauth.net/2/

[xlviii] https://openid.net/connect/

[xlix] https://en.wikipedia.org/wiki/XACML

[l] https://www.openpolicyagent.org/

[li] https://www.datanami.com/2021/01/21/governance-privacy-and-ethics-at-the-forefront-of-data-in-2021/

[lii] https://blog.christianposta.com/microservices/the-hardest-part-about-microservices-data/

[liii] https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/data-sovereignty-per-microservice

[liv] https://microservices.io/patterns/data/transactional-outbox.html

[lv] https://microservices.io/patterns/data/cqrs.html

[lvi] https://microservices.io/patterns/data/event-sourcing.html

[lvii] https://dl.acm.org/doi/10.14778/1454159.1454179

[lviii] https://www.forrester.com/report/The+Forrester+Wave+Enterprise+Data+Fabric+Q2+2020/-/E-RES157288

[lix] https://docs.oracle.com/en-us/iaas/Content/Identity/Tasks/managingcompartments.htm

[lx] https://db-engines.com/en/ranking_trend

[lxi] https://www.oracle.com/news/announcement/oracle-recognized-in-gartner-cloud-database-market-reports-121420.html

[lxii] https://blogs.oracle.com/dataintegration/gartner-mqdataintegration2020

Connect with us

+1.800.ORACLE1までご連絡いただくか、oracle.comをご覧ください。

北米以外の地域では、oracle.com/contactで最寄りの営業所をご確認いただけます。

blogs.oracle.com facebook.com/oracle twitter.com/oracle