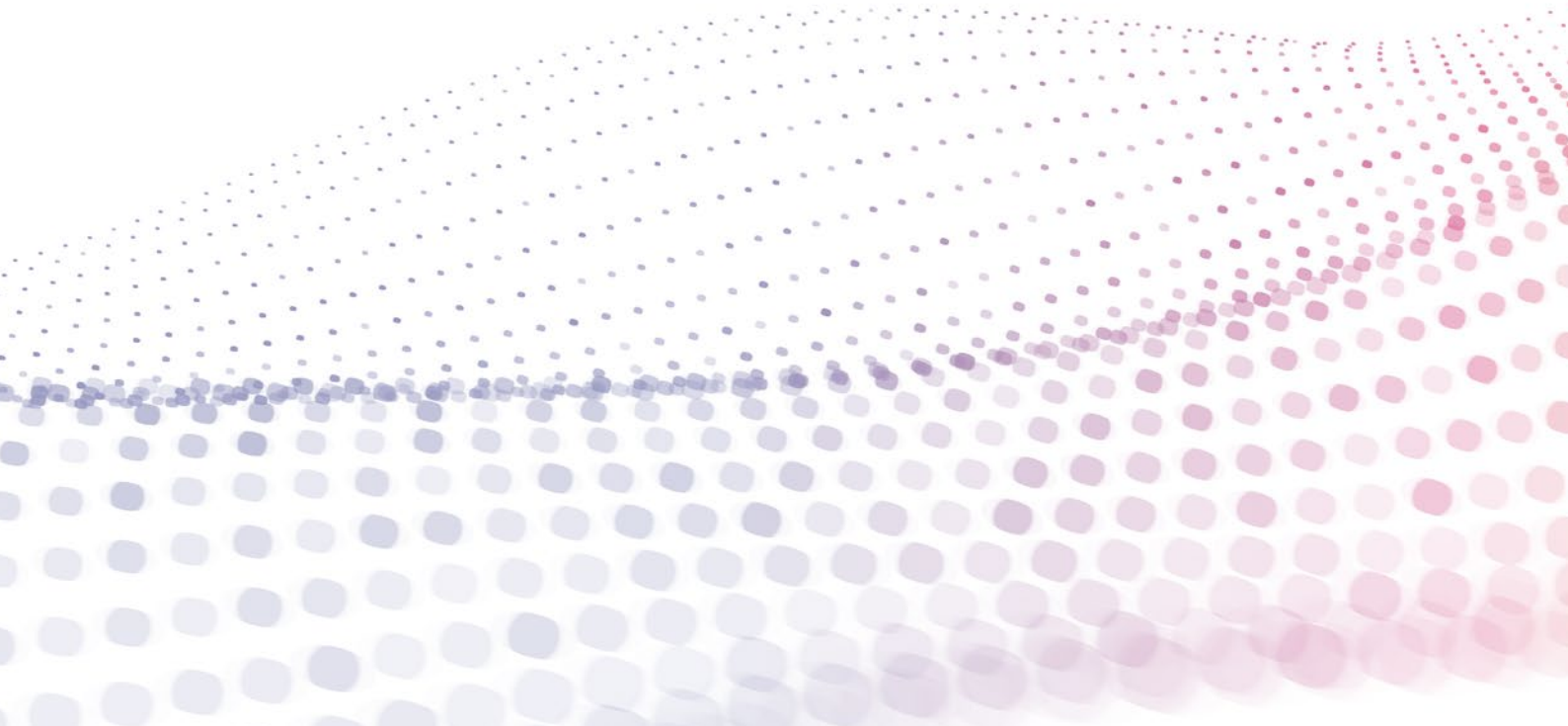
MarkLogic

Mongoでの失敗から MarkLogicでの成功まで

MARKLOGICテクニカルホワイトペーパー ・ 2018年1月

ある世界的な商用データプロバイダでは、自社サプライチェーンの核となるデータサービスアプリケーションで MongoDB を使用することを決定しました。しかし、プロジェクト開始から 9 か月後、MongoDB の制約により致命的な問題が発生しました。このホワイトペーパーでは、MongoDB がこのプロジェクトを完了できなかった理由、また MongoDB が解決できなかった問題を MarkLogic がどう解決したのかを紹介します。



目次

課題：リレーショナルデータベースからの脱却

2015 年 8 月、ある世界的な商用データおよび分析プロバイダが、自社の核となるデータサービスプラットフォームを最新化するために、MarkLogic に支援を求めてきました。このプラットフォームには、185 か国からの 1 億件を超えるレコード（この会社が扱う世界中からの全レコードの 33％）が格納されています。また、社内、パートナー、第三者に対してさまざまなデータサービスを提供しています。

従来のデータサービスプラットフォームは、まず 1990 年代半ばに Microsoft SQL Server で構築されましたが、最近では老朽化が見られました。同社の言葉を借りれば「深刻なガス欠状態」にあり、緊急な再開発が必要でした。

図1：データサービスプラットフォームでは、会社が保有する全レコードの3分の1以上を管理し、ビジネス上重要な役割を担っている

最初の決定：MongoDBの採用

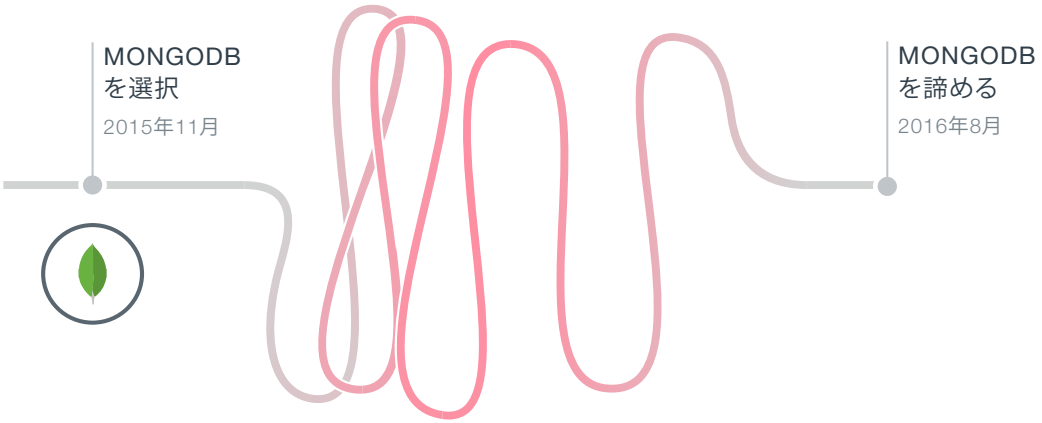
2015 年 11 月、同社は社内の公式対応製品リストに MongoDB を追加したばかりでした。

社内の担当チームは、データサービスアプリケーション内の 1 億件超のレコードを効率的に格納しクエリを柔軟に実行できるかについて、MongoDB の機能を評価しました。このレコードには 100 以上の言語が含まれ、17 の複雑なスキーマ形式（疎、階層型、コード化、変数）で格納されています。評価には MongoDB バージョン 3.2 を使用しました。当初いくつかの懸念があったものの、IT チームは MongoDB の使用を決断しました。

最初の失敗：MongoDBではうまくいかない

MongoDB 利用開始からほぼ 1 年、同社はデータサービスプラットフォームを MongoDB に置き換えるのに苦心していました。MongoDB は、同社の主要コンポーネントであるデータエクスポートサービスの重要要件を満たしていなかったのです。

データエクスポートサービスは、データサービスプラットフォームのデータサプライチェーンにおける中核的なコンポーネントです。その目的は、すべてのリージョン（地域）レコードに対するクエリの実行、任意の条件でのレコードの選択、それらのデータパッケージの効率的な抽出方法を、社内外に提供することでした。



MongoDBと9か月間格闘する

MongoDBにおける課題

データエクスポートサービスを MongoDB に置き換える試みの中で、特に以下の問題が明らかになりました。

1. 複数の大規模なリージョンを対象とした場合、検索を伴う複雑なクエリができない
2. 各リージョンのデータを全部メモリに読み込まないとクエリができない
3. 諺い文句のようにはシステムを拡張できない
4. ディスクが一杯になってもデータベースの読み取り / 書き込みが可能な状態が続き、データベースが破損する
5. 環境に対応するために、大量の管理サーバーが必要である

問題点の 1、2、3 は互いに関連し合い、結果的にデータエクスポートサービスシステムの重要要件に大きな影響を及ぼしていました。データ利用者は、同社が保有する 1 リージョン内の全レコードに対して任意のキー項目で柔軟にクエリを実行し、結果のデータセットをエクスポートする必要がありました。しかし MongoDB の限られたインデックスでは、「任意のフィールドでクエリを実行するという要件」においてそのリージョン全体の「ワーキングセット」（データおよびインデックス）すべてを RAM に読み込む必要がありました。つまり、自社の AWS 環境内の最大メモリの EC2 インスタンスの一部をスピンアップしなければなりませんでした。これには多額の費用がかかってしまいます。[1] また場合によってはこれに加えて、メモリのスラッシング、応答時間の大幅な低下やタイムアウトが発生しました。これは MongoDB の限界として良く知られているものです。[2,3]

ディスクが一杯でデータベースが破損するという問題点 4 は、ディスク容量が枯渇した場合に発生する MongoDB の問題として良く知られています。[4,5] 今回も、ディスクスペースの枯渇により、長時間かかるトランザクションが失敗しているのに後続のトランザクションが実行されることがありました。しかし、再起動して復旧すると、これらの「成功した」トランザクションは失われます。

1 http://www.ec2instances.info/?min_memory=244&region=ap-southeast-2

2 https://stackoverflow.com/questions/6453584/what-does-it-mean-to-fit-working-set-into-ram-for-mongodb

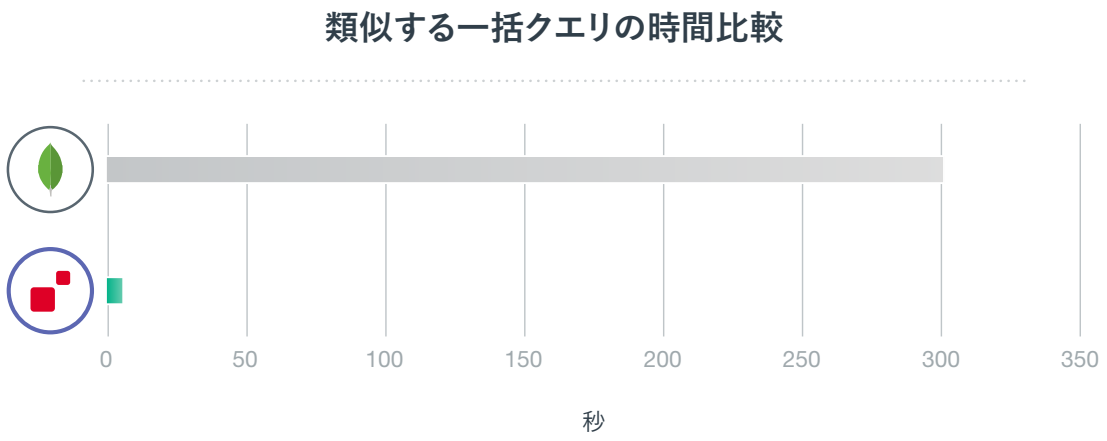
3 https://docs.mongodb.com/manual/faq/diagnostics/#faq-memory

4 https://jira.mongodb.org/browse/SERVER-13811

5 https://jira.mongodb.org/browse/SERVER-9552

同社が**問題点 5**、つまり大量の管理サーバーを問題点として挙げたのは、さまざまな種類のサーバーによって構成される MongoDB スタックの管理・拡張が煩雑だからです。[6] MongoDB は、MarkLogic よりも製品のセットアップと拡張が複雑です。[7, 8, 9]

2016 年 8 月、MongoDB による重要機能の実現に失敗した後、同社は MarkLogic 社に連絡し、MarkLogic がこの問題の解決にどう役立つかを把握しようとしました。



新しい機会：MarkLogicを試すことにする

事前打ち合わせ

MarkLogic は 2016 年 9 月に同社の担当チームと会い、1 か月かけて打ち合わせを行いました。同社はデータエクスポートサービスシステムで直面した問題の詳細と、MongoDB で実現できなかったクエリについて説明しました。同社は MarkLogic との話し合いの後、1 億件超の全商用データレコードを対象としたサービス要件実現の可否を確認する MarkLogic 主導による POC に関心を持ちました。

MarkLogic の SE（セールスエンジニアリング）はサンプルのレコードと MongoDB クエリをいくつか受け取り、さっそく作業を開始しました。まず、Dimension Data クラウドに MarkLogic の 3 ノードのクラスタをすばやく立ち上げました。次に、受け取ったサンプルから 2000 万件のダミーデータレコードを作成する Python スクリプトを作成しました。さらにサンプルの MongoDB クエリを MarkLogic クエリに変換し、MarkLogic の性能を測定しました。

6 http://stackoverflow.com/questions/13090601/how-to-determine-the-mongodb-server-type

7 https://docs.mongodb.com/v3.0/core/sharded-cluster-architectures-production/

8 https://serverfault.com/questions/727404/what-is-the-minimum-amount-of-servers-for-a-production-mongodb-cluster

9 https://docs.mongodb.com/manual/tutorial/deploy-shard-cluster/

MarkLogicのデモ機能

打ち合わせ中、SE チームは以下の内容をその場で提示できました。

- 約 5000 件の商用データレコードの 1 秒あたりの読み込み率をテスト — この結果から 1 億件のドキュメントの読み込みに 5.5 時間要すると推定
- 2000 万件のダミーレコードから 1 件の商用ドキュメントを返すまでのクエリ応答時間をテスト：最大で 0.002 秒
- 2000 万件のダミーレコードから最大 1600 万件のドキュメント URI を返すまでのクエリ応答時間をテスト：3.5 秒

10 月 6 日、MarkLogic は担当チームと再び会い、課題を改めて確認したうえで MarkLogic による対応策を具体的に説明し、技術的な作業の結果を説明しました。MarkLogic 社は、求められている以下の内容を MarkLogic で実現できる自信があると説明しました。

- **エンタープライズ機能** – ACID トランザクション、柔軟性の高いバックアップ機能、ビルトインのロールベースのアクセスセキュリティ
- **拡張性** – 大量データの素早い読み込み、クエリ、抽出
- **柔軟性なクエリ** – 構造化および非構造化データへの効率的なクエリ
- **システムパフォーマンス** – 大量のデータセットに 1 秒未満で応答

同社はこの最初の結果に興奮し、MongoDB では、1 か国のレコード（約 1500 万件）対象の類似した一括クエリで、マッチするドキュメントを返すのに 5 ～ 10 分以上かかることもあると話しました。

MongoDB では、クエリの柔軟性、大規模時のパフォーマンス、エンタープライズ機能に関して致命的な問題がありました。MarkLogic は最初の紹介でこういった問題を解決できたため、同社は非常に喜びました。

POCの対象範囲

ここで重要なのは、データレコードの複雑さ、多様性、コード化されたフィールドを考慮したとき、求められるクエリを実現するには、スキーマ非依存のストレージモデルとユニバーサルインデックスが必要不可欠だということです。また、1 リージョン内のビジネスレコードは何千万件にも及び、ユーザーによる検索やデータリクエストは何百万件にもなるため、大規模な場合でもパフォーマンスが良いことが必要です。

最後に商用データレコードでは、階層構造やコード化されたフィールドがバージョンごとに異なる可能性があるにも関わらず、任意のフィールドでフィルタリングやクエリを実行する必要があります。つまり、「必要に応じてレンジインデックスを作成する」という MarkLogic の標準的なアプローチではうまくいきません。レコードは極めて多様になりうるため、新規データソースの新しい値やデータフィールドに備えてレンジインデックスを作成しておくことは不可能なのです。

何回かの打ち合わせの後、同社の担当チームと MarkLogic は、MarkLogic の SE チームが行う概念実証（POC）の範囲と条件についてすぐに合意に達しました。POC の目標には、MarkLogic のインストールと設定、データの読み込み、システムサイジング、クエリ応答時間、リニアな拡張性、調査結果の報告が含まれていました。

POCの成功基準

- 社内の AWS 環境で 3 ノードの MarkLogic クラスタを構築し、設定する
- 1 億件以上の商用データレコードを MarkLogic に読み込み、リージョンコレクション（集まり）として整理する
- 拡張性とパフォーマンスの観点からメモリ、ストレージ、ネットワークおよびシステムのパラメータを確認する
- データの読み込みとクエリの観点から MarkLogic クラスタのリニアな拡張性を実証する
- テストクエリで 3 分以内の応答時間を実現し、クエリの条件にマッチする商用データ ID を返す

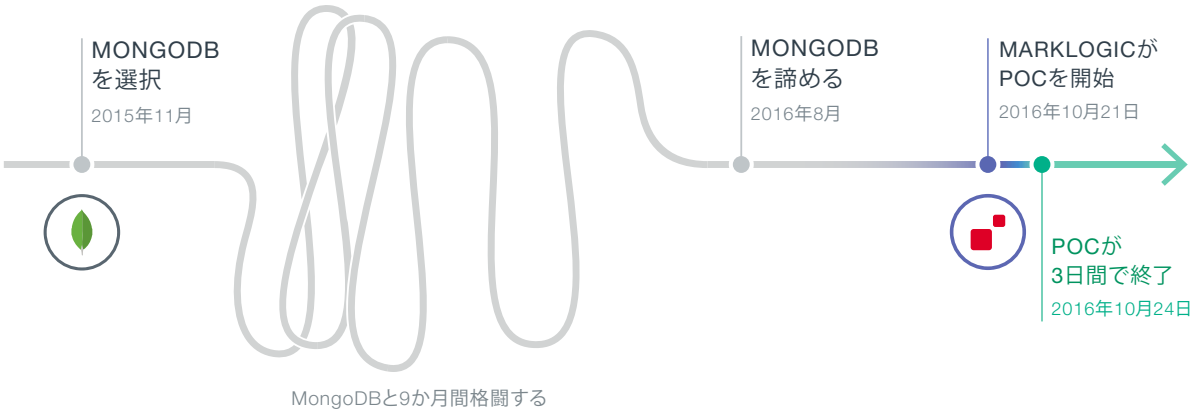
POCの成果

同社の AWS 環境へのアクセス権を取得したうえで、POC は 10 月 20 日に本格的に始まりました。MarkLogic は、合意済みタスクの完了に注力し、進捗レポートを毎日提出しました。

そしてたった 3 日間で、POC の主要タスクはすべて完了しました。つまり MarkLogic は、MongoDB が 9 か月かかって実現できなかったことを 3 日で実現したのです。以下が達成されました。

- 3 ノードの MarkLogic クラスタの設定
- 1 億 5000 万件以上の商用データドキュメントの読み込み（併せて読み込み率を測定）
- 高可用性用のレプリカフォレストのセットアップ、データのコピー。ノードのフェイルオーバー時における高可用性の証明
- 1 か国内の商用データコレクション（最大で 1500 万レコード）に対する指標的クエリの応答時間
- ウォームキャッシュ vs. コールドキャッシュ、および単独処理 vs. 並行処理の各条件で実行された指標的クエリの応答時間
- ストレージ、RAM 消費、メモリ使用などに関するシステム構成とサイジングの指標の把握

その翌週、MarkLogic は担当チームとテクニカルレビューを行い、結果のそれぞれについて、テスト手順の詳細を確認し、あらゆる質問に答えました。



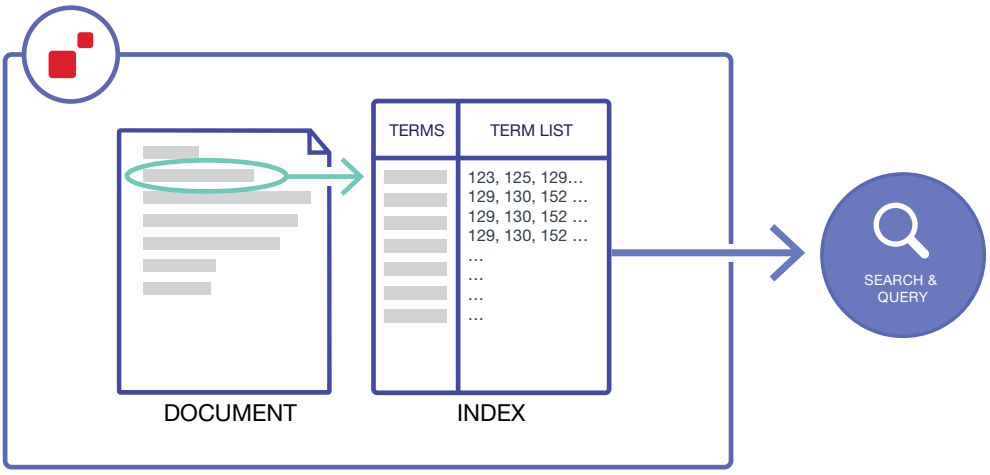


図5：MarkLogicのインデックスで高速かつ無制限のクエリが可能

この段階で、同社の担当チームは MarkLogic の次のような機能に特に感銘を受けました。

- **柔軟性が高く、設定可能な MLCP によるデータの取り込み**
  MongoDB から大量の GZIP ファイルに直接ダンプされた行区切りの JSON 生データに対して、MLCP を使って迅速にパーシングとロードを行い、MarkLogic 内で整理した

- **データの高速読み込み**
  MarkLogic は、1 億 5000 万件以上のデータレコード（要件より 5000 万件多い）を約 8 時間で MarkLogic に取り込み、これらのデータをすぐにクエリ可能にした

- **インデックス付きデータに対する柔軟なクエリとパフォーマンス**
  1 か国の約 1500 万件の商用データレコードに対してサンプルクエリを実行し、マッチした URI 数百万件を 2 秒以内に返す

最後のハードル

POC の最後のタスクとして求められたのは、大規模なデータコレクションに対して柔軟性の高いインデックスなしのクエリを MarkLogic で実行し、並行処理によってデータセットをクライアントに効率的に返す方法を示すことでした。

MarkLogic は検索機能を念頭に開発され、検索とインデックスによって大量の異種データに容易にアクセスできるように、一から設計されています。MarkLogic の進化の基となる設計理念として、「データの格納はソリューションの一部でしかない」というものがあります。つまり、データを迅速かつ容易に取り出し、さまざまなユーザーに応じて有意義な形で提示する必要もあります。データはまた、コモディティハードウェアで拡張可能なエンタープライズ仕様のソフトウェアソリューションで確実に管理する必要があります。

商用データの構造はバージョンごとにすぐに変わる可能性があり、またデータエクスポートサービスの利用者が常に任意のフィールドでクエリできる必要がありました。**ここで、MarkLogic が同一ユーザーリクエスト内でインデックス付きのデータとインデックスのないデータを対象にクエリを実行でき、また妥当な時間内で大量のデータセットをユーザーに返せることを実証する必要がありました。**

MongoDB は、この課題を拡張によって効率的に解決できませんでした。このようなリクエストに対して結果を返すのに、MongoDB では 10 ～ 15 分かかることもあったということです（前述の MongoDB の問題点 1、2、3 を参照）。

この課題を解決するために、MarkLogic は以下の戦略を講じました。

I. インデックスが肝心

MarkLogic の SE チームは、関連性のある検索可能なすべてのフィールドに、可能な限りいつでもインデックスを付けることが重要だと繰り返し強調しました。MarkLogic の強みは、洗練されたインデックス付けと、インデックスを活用したクエリの卓越したパフォーマンスにあります。

整数型のパスレンジインデックスを新しい商用データフィールドに追加するプロセス

- このフィールドを含むドキュメントの数：2,531,039 件
- 再インデックス付けに必要な時間：40 分
- ディスク使用 / インデックスサイズの増分：20MB

MarkLogic チームはさらに、多様かつ疎なデータを対象に、MarkLogic でいかにすばやく容易にレンジインデックスを追加できるかを示しました。MarkLogic では、効率を最大化するために、ユニバーサルインデックスを使用して対象フィールドを含むドキュメントのみを再インデックス付けします。

> MarkLogicの正しい使い方はインデックスを使用して問題解決することです」
>
> クリストファー・リンブラッド、MarkLogic 創業者＆チーフアーキテクト

II. ユニバーサルインデックスを使用したレンジクエリ

MarkLogic チームは、ユニバーサルインデックスを使用して有効な（または無効な）値とドキュメントを照合し、マッチする（あるいはマッチしない）ドキュメントを抽出する擬似レンジクエリの作成方法を示しました。

また、一方向の高速「等価」ルックアップに MarkLogic のユニバーサルインデックスがどのように使われるかについて説明しました。チームはこの機能を活用し、「コーディング」アプローチによって、希望の値または日付範囲に「マッチする」すべてのドキュメントを迅速に見つけ出すことができました。

以下は、日付範囲に対する擬似レンジクエリに、ユニバーサルインデックスのみを使用する例です。

```
let $results := json:to-array(cts:uris((), ('document', 'eager'), cts:and-query((
   cts:collection-query(("Country A")),
   cts:json-property-scope-query("Business", …
      cts:json-property-value-query("Val",
         (for $n in 1 to fn:days-from-duration(fn:current-date() - xs:date($last-date))
```

このアプローチによって、「インデックスのない」レンジクエリの応答時間が極めて速くなりました。

インデックスのないレンジクエリの実行における応答時間

- テストクエリは、値が「1000 より大きい」インデックスのないフィールドを含むドキュメントの URI を返す
- 対象のコレクションに含まれるドキュメントの合計数：最大で 1300 万点
- フィールドの値が「> 1000（1000 より大きい）」ドキュメント数：194,737 点
- URI を返すのに要する時間（コールドキャッシュ）：最大で 4.2 秒
- URI を返すのに要する時間（ウォームキャッシュ）：最大で 2.4 秒

III. ディスクに対するバッチおよび並行処理のクエリ

前述のアプローチ（上記 I と II）とは別に、MarkLogic チームは、インデックスのないフィールドに関してディスクに対して直接クエリを実行できることを証明し、またこの性能を測定することを求められました。この課題の解決にはさらに複雑な作業が伴いました。以下の処理を実行可能なテストユーザースクリプトおよび MarkLogic エンドポイントを含むテスト手順を作成する必要があったからです。

- レコード（URI）制約、スレッドプールサイズ、バッチサイズをパラメータ化してインデックスなしのクエリ条件に基づいてドキュメントを処理する

- MarkLogic の REST エンドポイントを呼び出し、インデックス付きの条件を使用してすべての URI クエリを実行し、マッチするすべての URI を返す

- URI のグループを 2 つめの MarkLogic エンドポイントに送信し、インデックスなしの条件でバッチ処理を行う

- 最終的にマッチしたドキュメントを完全な形で返し、結果を数えてまとめ、全過程を通じたパフォーマンスを測定する

インデックスのないフィールドについてディスクで直接クエリを実行する機能

ディスク利用で直接「インデックスなし」のクエリ条件を満たす 2 部構成のクエリプロセス

- 対象のコレクションに含まれるドキュメントの合計数：1300 万点
- フィールドの値が「> 1000（1000 より大きい）」ドキュメント数：194,737 点
- ワーカーのプールサイズ：96
- バッチサイズ：2000
- MarkLogic の REST エンドポイント A から URI を返すのに要する時間（コールドキャッシュ）：4.2 秒
- MarkLogic から最終的なマッチするドキュメントを返すのに要する時間（コールドキャッシュ）：42.1 秒
- MarkLogic の REST エンドポイント A から URI を返すのに要する時間（ウォームキャッシュ）：1.2 秒
- MarkLogic から最終的なマッチするドキュメントを返すのに要する時間（ウォームキャッシュ）：14.8 秒

このテストをパスするため、MarkLogic チームは以下のようなコンポーネントとテストワークフローを作成しました。

1. クエリを上から順番に実行する：

- 「インデックス付き」条件を使用して、MarkLogic の REST エンドポイントから初期 URI の全リストを取得する

- すばやく処理できる部分をまず実行し、ほとんどのレコードを最も効率的な方法でフィルタリングして除外する（利用可能なインデックスをすべて使用してドキュメントの候補リストを可能な限り絞り込む）

- URI のレキシコンインデックスを使用して、できるだけ効率的な方法で候補ドキュメントの URI リストを「呼び出し」アプリケーション（または「調整」クエリ）に戻す

2. URIのリストをグループに分割する：

- URI のグループを MarkLogic クラスタの REST エンドポイントに送り、クエリのうち、より低速の「インデックスのない」部分を並列で実行する

- 保留中の関数を使用してバッチクエリを処理し、できるだけ迅速に結果を「呼び出し」アプリケーションに戻す

3. 結果の各グループを最終的な結果に追加する：

- バッチリクエストで受け取ったそれぞれの URI について、手順 2 の MarkLogic エンドポイントは、「インデックスなし」のクエリ条件に基づいて URI を評価し、結果を「呼び出し」アプリケーションに戻します。そのうえで、「呼び出し」アプリケーション（POC 用の Python テストハーネスなど）は、完全にマッチしたドキュメントの最終リストを追加し、集約できます。

これにより、このアプローチがパフォーマンス要件実現に非常に有効なことが証明され、クエリのバッチ処理および並行処理をディスクで行う際の MarkLogic の性能が示されました。

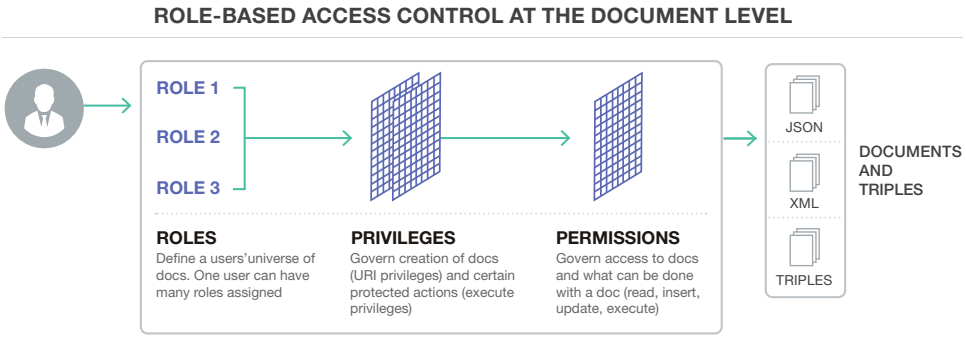


図 **6:** ロールベースのアクセスコントロール（**RBAC**）は、ユーザー権限を管理するために**MarkLogic**が標準で使用する主要なアプローチ

会社の成功のために、きめ細かいセキュリティを維持する

重要なこととして、データエクスポートサービスのすべての機能およびデータアクセスのニーズは、セキュリティおよびアクセス制御の対象となります。この会社の顧客は、アクセス権（リージョンごと、商用データのタイプごとなど）を持つ商用データレコードに対してのみ閲覧や抽出が許可されています。

データサービスアプリケーションには、商用データのリージョンやタイプが数多く含まれています。このため MongoDB では、このようなセキュリティを実装するために Base-64 文字列でアクセス制御をエンコードし、各データリクエストに対するビットごとの比較処理を実行してユーザー権限を確認するという複雑なスキームが必要でした。このアプローチは実装が難しいだけでなく、維持管理はさらに困難です。

この会社では重要な情報を保持し、その一部を特定の顧客に公開しています。MarkLogic の承認機能により、ドキュメントへのアクセス、XQuery および JavaScript コードの実行、ドキュメントの作成を制御するツールが提供できます。特定のユーザー権限ごとに、何が実行可能なのかを決めることができます。

MarkLogicにはデフォルトで、RBAC（ロールベースのアクセス制御）セキュリティモデルがあります。このモデルでは、各ユーザーに任意の数のロールを割り当て、そのロールを任意の数の権限やパーミッションに対応付けます。権限では、ドキュメント作成と機能実行（URI および実行権限）を管理します。パーミッションでは、ドキュメントに対する操作（読み取り、挿入、更新、実行）を管理します。セキュリティチェックにより、要求されたアクションの実行前に、ユーザーに必要とされる資格要件を確認します。セキュリティ情報は、MarkLogic 内の専用のセキュリティデータベースに格納されます。

例えば、RBAC と承認機能（オーソリゼーション）では、社員に対して特定リージョンのドキュメントの読み取りと更新を、また顧客に対してはこのリージョンのドキュメントの読み取りのみを許可し、当該リージョンへのアクセス権を持たない顧客に対してはドキュメントの存在そのものを把握できないようにできます。

さらに、MarkLogic では以下のことが可能です。

- 高度な暗号化
- 外部 KMS（鍵管理システム）対応
- 要素レベルのセキュリティ
- 監査
- 外部認証
- リダクション
- コンパートメントセキュリティ

同社は、MarkLogic により、商用データセットに対する階層型でロールベースの適切なアクセス制御を実現しました。これにより、あらゆるデータアクセスパターンのセキュリティを確保する、非常に透明性が高く、管理可能な手段が提供されました。

MarkLogicによる革新

この会社は、過去 9 か月間にわたって MongoDB と格闘してきましたが、MarkLogic では要件を満たす POC を 3 日間で作成できました。最終的には MongoDB の導入は失敗に終わりましたが、これは MongoDB に拡張性がなく、バックアップや検索ができなかったためです。さらに、環境に対応するために大量の管理サーバーを必要とし、データ破損の問題も発生しました。一方、MarkLogic はバックアップ、拡張性、迅速なクエリパフォーマンスといったエンタープライズ機能を証明できました。

同社は現在、データエクスポートサービスシステムを MarkLogic に移行中です。これにより、フル機能のセルフサービスポータルを社内外に提供できるようになります。ここでは、データサービスアプリケーションプラットフォームに格納されている商用データのビジネスレコードから、カスタムのデータパッケージを取得できます。

今後は、第三者に対しても徐々にサービスを拡張する予定です。第三者自身が自分のデータを MarkLogic に直接アップロードし、ハーモナイズさせたうえで、この混合データセットを既存の商用データセットと比較してリアルタイムで分析できるようになります。MarkLogic のデータセキュリティ機能により、第三者とのデータ共有を適切に行えます。MarkLogic を使うと、データの出自のトラッキング、必要なレベルのアクセスのみの提供、保有データの信頼性と説明責任を実現できます。

この会社では MarkLogic を使用して、最終的には優れたエンドユーザーエクスペリエンスと商用データへの画期的なアクセス方法を提供し、レベルアップしたビジネスデータサービスでビジネスの可能性を広げる予定です。

MARKLOGIC K.K.
150-0001 東京都渋谷区神宮前1-5-8 神宮前タワービルディング 13F
03 4540 0337 | jp.marklogic.com | MarkLogic-JP@marklogic.com

150-0001 東京都渋谷区神宮前1-5-8 神宮前タワービルディング 13F
03 4540 0337
jp.marklogic.com | MarkLogic-JP@marklogic.com