MarkLogic

リレーショナルを超えて

MARKLOGICホワイトペーパー • 2015年9月

30年以上にわたって活用されてきたリレーショナルデータベースでは、分断され多様かつ変化し続ける大量のデータの処理がますます困難になってきています。このため複数のリーダー企業がリレーショナルを超えた新世代のデータベースMarkLogicへと移行しはじめています。

現状の確認

データの扱いに関して、改善の必要があるでしょうか？　あるいは今のままで大丈夫でしょうか？　以下の質問に答えていただくと、現状がわかります。「はい」が多いほど、今のデータベースはニーズに応えられていないということになります。

		はい	いいえ
ビジネスに関する質問	1. 重要なデータのうち、まだデータベースにまだ入っていないものがある。	✓	×
	2. 本質的に同じデータが複数のデータベースに含まれている。	✓	×
	3. IT部門が一元管理していないデータソースがたくさんある。	✓	×
	4. 大規模なITプロジェクトのうち、予算超過やスケジュールの遅れが発生しているものが存在する。	✓	×
	5. データベースのスキーマが複雑すぎて、理解できる人がほとんどいない。	✓	×
技術に関する質問	6. データモデリングのせいで、アプリケーション開発が遅れたり止まったりしたことがある。	✓	×
	7. 「取りあえず動かす」ために、リレーショナルテーブル内の列名を変更したり、列名の解釈を変えたことがある。	✓	×
	8. データベースのスキーマが頻繁に変更される（月1回など）。またその変更がうまくいかないことがある。	✓	×
	9. 拡張方法を決めるのにかなりの時間とリソースを使っている。	✓	×
	10. ミドルティアが複雑なために、パフォーマンスの問題やバグが発生している。	✓	×

目次

はじめに 1

今日の ビッグデータの世界 2

- Volume（量）
- Velocity（速度）
- Variety（多様性）
- Veracity（正確さ）
- Variability（変動性）

複雑さの海に溺れる 3

- データサイロとETL
- 「シャドーIT」とセキュリティの崩壊
- 高コスト、プロジェクトの失敗、不可能な革新

リレーショナルデータベースではうまくいかない理由 4

- リレーショナルデータベースは変化に対応できない
- リレーショナルデータベースは多様なデータを扱うように作られていない
- リレーショナルデータベースはスケーラビリティと弾力的な拡張・縮退用に作られていない
- リレーショナルデータベースは多様なワークロードを扱うように作られていない
- リレーショナルデータベースは現代的アプリケーション開発には向かない

今日のデータのための新世代のデータベース 8

- MarkLogicの特徴的な機能の概要
- リーダー組織がMarkLogicでさらに多くを実現

リレーショナルを超えてデータを活用する 12

- 推奨される次のステップ
- さらに詳しく

はじめに

ここ数十年においてテクノロジーが急激に進化した結果、ビジネスのやり方もあらゆる局面で変化しました。今日、これまでになかった量のデータが収集されており、巨大かつスマートなアプリケーションが構想されています。では、組織にとって最も重要なアセットであるデータを格納する「データベース」も変化したのでしょうか。ほとんどの場合、これは全く変わっていません。

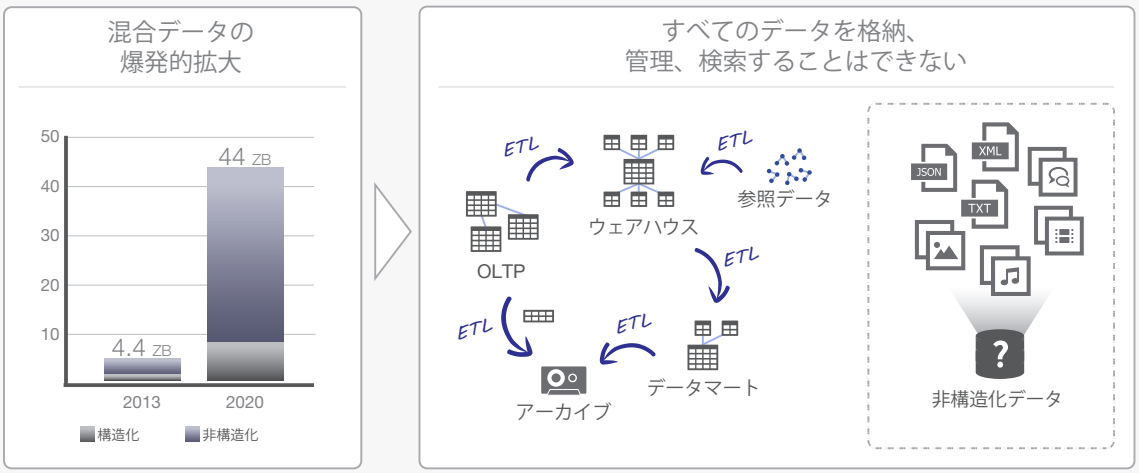
データの格納と管理の主流テクノロジーであるリレーショナルデータベースは、最初に登場した30年以上前（冷戦時代）とほぼ同じに見えます。当時、データは小さく、きれいで、構造化され、変化しないものと考えられていました。これはそういうものしか格納できなかったからです。しかしデータとはそのようなものではありません。今日の世界では、データとは巨大で頻繁に発生し、多様で変化するものです。この現実に直面しなくてはなりません。管理対象となるシステムも数個ではなく、数百個に及び、またペタバイト級のデータを扱わなくてはなりません。

「ビッグデータ」という新しい世界はエキサイティングなチャンスをもたらしますが、片やIT部門が対処すべき新たな課題とみなされることが極めて多いのも事実です。現在、IT部門は、そのほとんどの時間を現状維持のために使っています。複雑に絡み合うデータサイロや、データをあちこちに移動させるための大量のETL（抽出/変換/ロード）を管理する必要があります。また各個人や業務部門が独自に問題を解決しようとした結果、「シャドーIT」ならびにセキュリティ問題が発生してきました。業界を問わず、高コストや長期間のプロジェクトはごく当たり前になっています。また恒常的な保守サイクルから抜け出せず、全データの最大活用ができません。こういったことが起こるのは、元来向いていない問題をリレーショナルデータベースで解決しようとしたためです。

今日、すべての問題をリレーショナルモデルだけで解決することはできず、変化へのニーズに対応するため新しい種類のデータベースが受け入れられはじめています。MarkLogicはこういった世代交代の先頭に立ち、今日のあらゆるデータに適したデータベースを提供します。大量かつ多様なデータの格納、管理、検索に対応する柔軟なデータモデルがあり、また組織に必要なエンタープライズ機能がすべて備わっています。このユニークなコンビネーションにより、リレーショナルを超えてこれまでになかった価値をより多くのデータから得ることができます。

変化が必要な理由

分断された多様かつ変化し続ける大量のデータの処理は、ますます困難になってきています。この問題に対処し、リスク削減、短期間でのスマートなアプリケーション開発、データの業務活用を促進するには、新しいアプローチが必要です。



ビッグデータに関するこれらのVを総合すると『今日のデータは大量で、速く、複雑で、変化する』ということになります」

今日のビッグデータの世界

かつては技術的制約により、どのデータもほぼ同じでした。データはゆっくりと順番にデータセンターに届き、事前設定されたテーブル内の行と列に表形式データとして整理されました。ビジネスもITも変化のペースは遅く、この方法でも問題はありませんでした。しかしそのような1980年代と現在では、状況は大きく異なっています。

今日のビッグデータには、3つのV、つまり「Volume（量）」「Velocity（速度）」「Variety（多様性）」の特徴があるといわれています。これに加えてもう2つのV、「Veracity（正確さ）」「 Variability（変動性）」の重要性も高まっています。全体的に見てこれらの「V」を総合すると、「今日のデータは大量で、速く、複雑で、変化する」ということになります。

VOLUME（量）

デジタルの世界は年40%の成長を続けており、2013年の4.4ゼタバイトから2020年には44ゼタバイトに成長すると予測されています（1ゼタバイトは1兆ギガバイト）。[1] この状況において、データベースが紙に替わって「システム・オブ・レコード」（記録システム）として「あらゆるもの」を格納します。今日では、以前よりも多くのシステムから多様かつ大量のデータを扱う必要があります。これを効率的かつ安全に少ないオーバーヘッドで管理することが求められています。データ格納コストは継続して減少しており、消費者ならび規制監督庁は組織が全データを格納することを求めています。

VELOCITY（速度）

今日の世界ではすべてが加速しており、データの作成ならびに変化のスピードも速くなっています。市場、新規経営陣、オンデマンドサービス、企業買収やスピンオフに伴う急激な変化に対処するための新規業務要件に合わせて、データのクエリ・利用法もすぐに変化します。今日、意思決定にかけられる時間は数日ではなく数分です。こういった意思決定を支えるデータを適切な形式で適時効率的に提供する必要があります。スポーツの試合のデータ提供や銀行での不正検知など、リアルタイムでのデータ入手はもはや願望ではなく必須要件です。またアプリケーション開発期間は、年から週単位へと大幅に短縮されています。これに加えて、忍耐力やロイヤルティが低下しパーソナライズ要求が高いユーザーに対応する必要があります。

VARIETY（多様性）

これらの「V」の中で最も困難なのが多様性です。今日のデータは以前に比べてさらに多様で、いろいろな種類のものがあります。約20%が構造化データ（トランザクション、表形式データなど）で、80%が非構造化データ（ドキュメント、テキスト、メール、画像、ビデオなど）です。[2] 新しく登場してきた非構造化データソースは間違いなく問題となります。ある調査によると、企業がビッグデータに対する新しいアプローチを採用する理由の64%は、多様で新しくストリーミングとして入ってくるデータソースに対応するためです。[3] しかし実際には、組織にとっては多様な構造化データの方が問題は大きく、急激に増加し変化し続けるさまざまな形式、サイズ、タイプの構造化データの扱いに苦慮しています。新規アプリケーション、M＆A、データの再活用などにより、同一構造化データのさまざまなバージョンが発生します。

VERACITY（正確さ）

正確さとは、データの真実性や一貫性に関するものです。データは極めて価値が高いアセットであり、組織はデータが正確であり破損していないことを十分な時間をかけて確認します。このためデータのリネージ（経緯）やライフサイクルの把握が重要です。これには、どこでデータが発生したか（出自）、その後の履歴（誰がどのように変更したのか）、保持（どのくらい期間保持すべきか）、重要性（最適な回答を与えるデータはどれか）などが含まれます。これに加えて、組織には強力なデータガバナンスポリシーが必要で、これを使ってユーザーのデータアクセスを細かく設定して保護できます。こういった点は、単に高度な分析だけではなく監査や規制においてもさらに重要性を増しています。

VARIABILITY（変動性）

変動性とは、コンテキストによってデータの意味が違ってくるということです。Forresterの主席アナリストであるブライアン・ホプキンスによると、変動性とは「自然言語における意味の差異のことであり、またこの問題をビッグデータ技術によってどのように解決するのか」ということです。[4] 例としては「sub」という言葉だけでは、「潜水艦」

1 IDC、「Digital Universe」、（2014年4月）<http://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm>

2 Khan et al、「Big Data:Survey, Technologies, Opportunities, and Challenges」Scientific World Journal、（2014年）<http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4127205/#B53>

3 New Vantage Partners、「Big Data Executive Survey:Themes & Trends」（2012年）<http://newvantage.com/wp-content/uploads/2012/12/NVP-Big-Data-Survey-Themes-Trends.pdf>

4 Hopkins, Brian、「Blogging From the IBM Big Data Symposium - Big Is More Than Just Big」（2011年）<http://blogs.forrester.com/brian_hopkins/11-05-13-blogging_from_the_ibm_big_data_symposium_big_is_more_than_just_big>

> ビッグデータの成功を阻む最大の要因は、データサイロだとされています」

のことかSubway® のサンドイッチのどちらなのかわかりません。これは単に自然言語の問題にとどまりません。利用者やデータモデリング担当者が基本的なエンティティを異なった方法で記述している場合もあります。例えばノースカロライナ（North Carolina）州は、「N Carolina」と表現されることも、また単に「NC」と表現されることもあります。この場合、データベースはこれらが同じものだと認識できるのでしょうか。またそもそも「州」という概念を認識できるのでしょうか。人間はコンテキストに基づいて有用な知識を読み解くことができますが、データベースにはこういった意味の把握は困難です。人、場所、モノがさまざまな方法で記述されているデータが増えた場合、この問題はさらに大きくなります。

複雑さの海に溺れる

今日のビッグデータは巨大なチャンスと考えることもできるはずですが、厄介な問題だとみなされることも多々あります。現在、IT部門はほとんどの時間を何とかやっていくことに費やしています。もしこの複雑な問題に正面から取り組まなかった場合、会社全体が沈没してしまう可能性もあります。

データサイロとETL

ビッグデータの成功を阻む最大の要因は、データサイロだとされています。[5] これは、複雑なエンタープライズアーキテクチャの図を見るとすぐにわかります。相互互換性がない旧来のシステム同士が組み合わされた結果、複雑で融通が利かないアーキテクチャが生まれ、そこでのデータの共有や活用は不可能です。ほとんどの組織においては、このアーキテクチャと細かいビジネスルールとの対応関係を把握しているのは少数のエキスパートのみです。この結果、当然ながらほとんどのビジネスインテリジェンスプロジェクトにおいては、多くの時間がデータソースの特定とプロファイリングに費やされています。[6]

データサイロは意図的に作られた訳ではなく、短期的な解決策の結果として生まれています。データベースのほとんどは、特定のアプリケーションあるいはデータタイプに特化して設計されています。これらのデータベースからデータを取り出したり、他の目的用に他のデータベースで利用するには、ETL（抽出/変換/ロード）処理で新しい対象データベースのスキーマにマッチさせる必要があります。ほとんどの組織では頻繁にETLが発生し、この結果、新たなデータサイロが生み出されています。

サイロが増殖するほど、保守や互いの関連付けが困難になります。この結果、多様なアプリケーションを結びつけるために応急処置的なメンテナンスコードが使われ始めますが、これは根本的な問題解決にはなりません。このやり方では問題はさらに複雑になり、そのうち何かが動かなくなったり、フラストレーションが溜まった開発者が離脱したり、新規プロジェクトが滞り全く進まなくなることさえあります。

「シャドーIT」とセキュリティの崩壊

IT部門が把握・管理していないソフトウェアを使って社員や業務部門が個々の問題を解決しようとした結果、CIOの目が届かない企業データが出てきます。CIOの多くは「シャドーIT」アプリケーションが数十個使われていると考えていますが、実際には数百個使われていることが多いです。ある調査によると、企業には驚くべきことに923個のクラウドサービスがあり、そのうち社内のセキュリティ要件を満たしているのはわずか9.3%でした。[7] このような変化の直接の原因は、ビジネスのニーズに対応できないことであり、これにより組織全体において大きなリスクと非効率性がもたらされています。

このようなことが起こっているのは、セキュリティの欠如に起因するコストが継続的に拡大し、サイバー犯罪による攻撃がより洗練されたものとなっているためです。たった一度のデータ漏洩だけでも組織の評判は大きく損なわれ、そのコストも大きなものとなります。ある研究によるとサイバーセキュリティ事案1件当たりの平均コストは540万ドル、1レコード当たり188ドルです。[8] 残念ながらデータ保護はこれまでにないほど困難になっていますが、これはデータサイロが大量に存在するために侵入可能な部分が増え、脆弱性、データ漏洩が増えているからです。

5 Oracle、「IT Assessment Complexity Survey」（2015年）<http://www.oracle.com/us/corporate/features/it-complexity-assessment-survey-2281110.pdf>

6 Boris Evelson、「Boost Your Business Insights By Converging Big Data And BI」Forrester、（2015年3月25日）<https://www.forrester.com/Boost+Your+Business+Insights+By+Converging+Big+Data+And+BI/fulltext/-/E-RES115633>

7 Skyhigh「Cloud Adoption and Risk Report - Q1 2015」（2015年6月）<http://info.skyhighnetworks.com/rs/skyhighnetworks/images/WP%20CARR%20Q1%202015.pdf>

8 Ponemon Institute「2013 Cost of Data Breach Study:Global Analysis」Symantec、（2013年）<https://www4.symantec.com/mktginfo/whitepaper/053013_GL_NA_WP_Ponemon-2013-Cost-of-a-Data-Breach-Report_daiNA_cta72382.pdf>

> 未だにリレーショナルデータベースのベンダーは、1960年代のアイディアに基づき、70年代のデータ問題を解決するため、80年代のプログラミングを使用している90年代の製品を販売しています」

高コスト、プロジェクトの失敗、不可能な革新

残念ながら、ITプロジェクトでは多くの場合、期日を守れず予算も超過すると予想できます。高コストとプロジェクトの失敗は当たり前のことで、実際、予算1500万ドル以上のITプロジェクトの半数は、45%の予算超過、7%の遅延となります。また想定されていた機能の56%しか実現されません。もっと悪いことには、失敗したITプロジェクトの17%はあまりにダメージが大きく、会社存続の危機までもたらす可能性があります。[9] こういったプロジェクトでは長い時間をかけたプランニング、大量の人員、優秀な人材から構成される大規模なチームがあったとしても、このような結果となってしまうのです。

これらのチームは予算超過で期待外れのプロジェクトに取り組んでいる一方、組織の成功に重要な意味を持つ革新的プロジェクトに費やす時間はありません。ビッグデータに携わる人がいなければ、これから価値を得ることはできません。今日、データベース予算の95%はリレーショナルデータベースに費やされていますが、そこに格納されているのは企業データの約20%でしかありません。[10] この結果、企業内の残り80%のデータを管理するためのお金は5%しか残っていません。従来の方法を変えない限り、CIOやIT部門は大量のレガシーシステムの面倒をみつづけなくてはならず、より多くのリソースを革新やビジネス変革に割いている競合他社に負けてしまいます。

リレーショナルデータベースではうまくいかない理由

多くの場合、今日のビッグデータに関する問題や複雑さの原因はリレーショナルデータベースにあります。これはリレーショナルデータベースに根本的な欠陥があるということではなく、そもそもこれは現代のデータを扱うように設計されていないということです。米国連邦政府の前CIOヴィヴェック・クンドラ氏が2009年に「構造化されたリレーショナルデータベースという観点からデータをとらえる考え方は死んだ」といったのはこのためです。

1970年代後半に発明されたリレーショナルデータベースは、階層型メインフレームシステムに代わって90年代初期には主要なデータベースとなりました。リレーショナルデータベースは、初期のコンピューティングのニーズにも合致していました。これによりデータはアプリケーションから分離され、カスタムコーディングも減りました。また一般的なクエリ言語であるSQLでデータへのクエリをより細かく制御できるようになりました。

リレーショナルデータベースの約40年の歴史において、各ベンダーの製品を取り巻くエコシステムは継続的に改善されてきましたが、データ管理の根本的なモデルは変わっていません。実際のところ、リレーショナルデータベースベンダーが提供しているのは、1960年代のアイディアに基づいて70年代の問題を解決するために80年代にコーディングされた90年代の製品なのです。[11] 現在求められているのは、リレーショナルデータベースが提供している以上のものなのです。その理由は次のようになります。

リレーショナルデータベースは変化に対応できない

リレーショナルデータベースは、データを表形式（行と列）に整理します。これはMicrosoft Excelのスプレッドシートのようなものです。各行がユニークなエントリを、また各列がユニークな属性を表します。1つの列を主キーとして選択して、テーブル内の各行を一意に特定します。

例えば、顧客と顧客が購入した商品をリレーショナルデータベースでモデル化する場合、まず「Customers」（顧客）テーブルを作成し、その中に「CustomerID」列を作成して、これを主キーとして使用します。また各顧客に関する属性として、「FirstName」、「LastName」、「Address」といった列を追加し、格納されるデータの型を定義します。[12] 次に、「CustomerID」を購入に関する別のテーブル「Orders」にリンクします。「Orders」テーブルの各行には一意の識別子があり、「Customer」テーブルの主キーを参照します。

このプロセスを繰り返してさまざまなテーブルを作成していきます。この際、この設計がエンティティならびに参照整合性に関するすべての制約を満たすようにします。またすべてが適切に「正規化」されている必要があります。つまり列が重複せず、またこれらの列がすべて主キーに依存し、複数のテーブルにおいて情報の重複が全くないようにします。こういった制約により、リレーショナル

9 McKinsey & Company、「Delivering large-scale IT projects on time, on budget, and on value」（2012年10月）<http://www.mckinsey.com/insights/business_technology/delivering_large-scale_it_projects_on_time_on_budget_and_on_value>

10 Carl Olofson、「Worldwide Database Management Systems 2014–2018 Forecast and 2013 Vendor Shares」IDC、（2014年6月）<http://www.idc.com/getdoc.jsp?containerId=248952>

11 Edgar "Ted" F. Codd氏が最初に有名な論文をIBM社内で発表したのは1969年のこと。その後、1970年にこの論文は社外にも公表された。（E.F.Codd、「A Relational Model of Data for Large Shared Data Banks.」Communications of the Association for Computing Machinery、Vol 13 No 6 377～387ページ）

12 1980年代、リレーショナルデータベースの列名には、8文字また大文字小文字のいずれか、という制約があった。このため列名は「fname」「lname」のようなものであった。現在、SQL標準においては列名は30文字までとなっている。

> 構造化されたリレーショナルデータベースという観点からデータをとらえる考え方は死んだ」
>
> Vivek Kundra, Federal CIO, July 21, 2009; Open Government and Innovations Conference

モデルの特徴であるデータの一貫性と高速なクエリが実現されます。データモデルすなわちスキーマ設計のプロセスでは、どのようなテーブルを作成するのか、またどのような列名にするのかを担当チームが決めていきます。これは重要なプロセスであり、最終的には廊下の目立つところに張り出されるような巨大なER図として表現されます。

このアプローチには2つの問題があります。まず、これには数か月、データベースのサイズによっては数年かかります。リレーショナルのスキーマは複雑なだけでなく、データのロードやアプリケーション構築の前にモデリングがすべて終わっている必要があります。2つめの問題として、データベース上に構築されたアプリケーションの改修が必要になった場合、時間と労力が大量にかかります（数か月から数年）。リレーショナルモデルは繊細で複雑な熱帯雨林のエコシステムのようなもので、小さな変更を1つ加えるだけで、データベースやアプリケーションに雪崩式に致命的な影響が発生する可能性があります。表への行の追加や置換といったシンプルな変更でさえ、数億円規模のお金がかかる可能性があります。[13]

今日、変更は頻繁に発生しますが、リレーショナルデータベースではデータモデリングに時間と労力がかかるので、これが大きな問題となります。将来にわたって絶対に変えなくてもよいできるだけ「完璧」なモデルを作成・再作成するために、データモデリングやETL処理に年間数千億円規模のお金が費やされています。しかし変更は避けられません。

リレーショナルデータベースは多様なデータを扱うように作られていない

データベース支出の95%はリレーショナルデータベースに対するものですが、リレーショナルデータベースで扱える構造化データは全体の20%でしかありません。これは驚くべきことです。[14] 構造化データの扱いはますます困難になってきていますが、残り80%の非構造化データにいたっては、驚くべき価値を秘めているもかかわらず完全に手つかずになってしまっています。

企業はこれまで、主要なトランザクションデータや顧客の基本情報のいくつかだけを格納していました。しかし今や、重要な一部のデータだけを選り好みすることはできず、ほぼすべてのデータを格納する必要があるのです。これを実現するためのインフラのコストがリーズナブルになったことで、リスク抑制とコスト削減が可能になりました。また顧客、パートナー、規制監督庁の側も、企業に対して自分たちも利用できる形式で全てを格納するように期待しています。

増加する構造化データはリレーショナルデータベースにとって問題となります。それぞれのデータソースの構造が違っているからです。前述したように新規データソースへの対応は面倒であり、スキーマがますます複雑になります。これは新しいデータがこれまでと同一の分野や概念を表している場合でも同じです。

またリレーショナルデータベースにとっては、増加する非構造化データも問題です。リレーショナルデータベースの列や行は値セットを格納するのには最適ですが、ほとんどの情報はそれ以外のものでできています。例えば個人の医療記録について考えてみましょう。これには、**値**（名前、生年月日）、**関係性**（家族とのまた病院との関係、疾病と投薬の関係）、**地理情報データ**（住所）、**メタデータ**（出自、セキュリティ属性）、**画像**（CTスキャン）、**フリーテキスト**（医者のカルテ、音声の書き起こし）などが含まれます。

これらすべてをMicrosoft Excelのスプレッドシートに入れることを考えてみましょう。この際いろいろと頭を使う必要があり、以下のような困難な選択を迫られます。長い文章は分割して、表内のセルを埋めていくべきか。後から追加された新規データソースはどうやって格納するのか。メタデータ用の列はいくつ準備したらよいのか。さまざまなエンティティ間の関係性はどう扱うのか。ドキュメント内部の構造どうするのか。どのようなインデックスを作成しなくてはならないのか。行や列で定義されていない要素でデータのフィルタリングを行いたい場合はどうしたらよいのか。

リレーショナルモデルであらゆるものを扱おうと頑張ってみたところで、そもそもこれは多様なデータを扱うように作られていないという事実は変わりません。

13　フォーチュン100のテクノロジー関連のリーダー企業によると、1列足すのに最大で1年かかり、またお金も1億円以上かかる可能性がある。マスターデータ管理などを含むより複雑なデータモデリングプロジェクトにおいては、5年以上かかった例も報告されている。

14　IDC（2014年6月）

> データベース支出の95%はリレーショナルデータベースに対するものですが、リレーショナルデータベースで扱える構造化データは全体の20%でしかありません。これは驚くべきことです」

リレーショナルデータベースはスケーラビリティと弾力的な拡張・縮退用に作られていない

今日、組織には数百万人のユーザーと数ペタバイトのデータがあります。また、クラウド内でアプリケーションを実行し、さまざまな場所にある数百台ものデスクトップ、タブレット、モバイルデバイスに対してダイナミックなコンテンツを提供しています。こうした新たな現実に対処するには、スケーラビリティ（データやユーザーの増加に伴う容量の追加）に加え、エラスティシティ（システム拡張のしやすさ。一般的にはユーザーの需要がなくなった場合の縮退を指す）が必要になります。

残念ながら、リレーショナルデータベースを拡張するのは困難です。リレーショナルデータベースは、単一のサーバー上で実行することによりテーブルのマッピングの一貫性を維持し、分散コンピューティングの問題を回避しています。こうした設計では、システム拡張が必要になると、処理能力が高く、より大容量のメモリとストレージを備えた大型で複雑な高コストの専有ハードウェアを購入する必要が生じます。さらに、アップグレードも課題となります。取得プロセスに時間がかかるうえ、実際の変更時には、通常、システムをオフラインにする必要があるためです。これらすべてが、ユーザー数が増加し続けている最中で発生するため、プロビジョニングが十分でないリソース不足の環境では、制約とリスクがより一層増加します。

こうした懸念事項に対応するため、リレーショナルデータベースのベンダーはさまざまに改善した製品を出しています。今日、進化したリレーショナルデータベースでは以前よりも複雑なアーキテクチャを利用できますが、これは「マスター/スレーブ」モデルに基づいています。「スレーブ」とは、レプリケーションされたあるいは「シャーディング」されたデータ（分割して複数のサーバーすなわちホストに分散されている）を扱うことで、マスターサーバーのワークロードを軽減する追加サーバーのことです。共有ストレージ、インメモリ処理、レプリカの有効活用、分散キャッシュをはじめとした機能強化や新しいリレーショナルアーキテクチャにより、リレーショナルデータベースの拡張性は確実に向上しています。しかしながら実際には、障害が発生しうるシステムや箇所を見つけるのはそれほど難しくありません。[15]

またリレーショナルデータベースの機能強化には、コストがかかり大きなトレードオフもあります。例えば、データをリレーショナルデータベースに分散する際、パフォーマンス維持のために、通常は事前定義されたクエリを利用します。つまり、パフォーマンスのために柔軟性が犠牲にされます。また、リレーショナルデータベースはその設計上、縮退ができません(つまり弾力性に欠けています)。データの分散後に容量を追加した場合でも、データの「分散をやめてもう一度まとめる」ことはほぼ不可能です。

リレーショナルデータベースは多様なワークロードを扱うように作られていない

「多様なワークロード」というのは、オペレーショナル（業務）ならびに分析のワークロードのことです。オペレーショナルワークロードには、リアルタイムでの日々の業務トランザクション（例：多数の顧客による購入）などがあります。分析ワークロードとは、ビジネスインテリジェンスやデータマイニングの作業のことです（一定期間内の購入の集計など）。

データベースは、1990年代半ばにオペレーショナル（業務）ワークロード用のOLTP（オンライントランザクション処理）システムと分析ワークロード用のOLAP（オンライン分析処理）システムの2つに分かれました。OLTPシステムでは、構築されたアプリケーション用にデータをモデル化し、一貫性のあるスピーディなトランザクションが求められます。一方OLAPシステムでは、集計やトレンドなど、スライシングやダイシング用にデータをモデル化します。最近になって、さまざまなシナリオに対応できる洗練された最適なモデルが作成されました。これによりデータモデリング担当者たちが「スタースキーマ」「スノーフレークスキーマ」「OLAPハイパーキューブ」という言葉を使い始めました。

残念ながら、オペレーショナルシステムと分析システムに分けられたことにより、データマート、データウェアハウス、リファレンスデータデータストア、アーカイブが乱立する結果になりました。オペレーショナルシステムからのデータは、ETL経由で中央のデータウェアハウスに移動され、業務上のあらゆる意思決定に利用されます。し

15　例えばオラクルRACはクラスタ認識型ファイルシステムを利用する、「クラスタ化された」リレーショナルデータベースだが、これは依然として共有ディスクのサブシステム上にある。

> ここでの問題は、リレーショナルモデルは複数のユーザーグループに情報を適切な形式ならびにタイミングで提供できるように作られていないため、IT部門の作業が複雑になってしまうことです」

かしこのように分断されていると、新規の利用法や利用法の変更に対応できません。このため、特定データのサブセットをデータマートに移動させるためのETL処理が必要です。またリファレンスデータを扱うためのシステムも別途作成されます。そしてアーカイブシステムが、すべてのシステムからのあらゆる履歴データを扱います。新しい質問や新規アプリケーションが出てくるたびに、改善されたモデルが新規作成されます。これまでと同じモデルというものは存在しません。最初はスキーマはシンプルでデータベースの数もそれほど多くないかもしれませんが、これはすぐに数百個のレベルまで拡大します。

このような理由などで、多くのIT部門は組織内の大量のシステムの保守にほとんどの時間とお金を使っています。ここでの問題は、リレーショナルモデルは複数のユーザーグループに情報を適切な形式ならびにタイミングで提供できるように作られていないため、IT部門の作業が複雑になってしまうことです。

> 従来のリレーショナルアーキテクチャでは、結果としてパフォーマンスが損なわれ、バグが増える可能性があります」

リレーショナルデータベースは現代的アプリケーション開発には向かない

現代のアプリケーションは、Java、JavaScript、Python、C#などのオブジェクト指向プログラミング言語で作成されます。これらの言語では、データ構造をデータやコード（属性やメソッドなど）を含む「オブジェクト」として扱いますが、これはリレーショナルデータベースによるデータの扱い方と大きく異なっているということが問題となります。このためデータベースとアプリケーション開発の間にミスマッチが発生します。

このミスマッチを回避するため、ORM（オブジェクトリレーショナルマッピング）という開発手法が利用されます。これはアプリケーション層内のオブジェクトとリレーショナルデータベーススキーマとして表現されたデータ間のアクティブ/アクティブの双方向マッピングです。ORMを使うと、アプリケーション開発者はビジネスルールならびロジックを扱うことができ、アプリケーション開発の観点から有用な方法でデータを表示できます。このアプローチでは、データベースは単にデータが永久保存されるストアドプロシージャが格納される場所だということになります。ORMのツールはいろいろあり、リレーショナルデータベースでのアプリケーション開発がシンプルになります。ORMツールの例としては、Hibernate（Java）、ActiveRecord（Ruby on Rails）、Doctrine（PHP）、SQLAlchemy（Python）などがあります。

しかし残念ながら、ORMはリレーショナルデータベースに起因する問題の回避策としては、それほど有効ではないと考えられています。ORMは「コンピュータサイエンスのベトナム戦争」と呼ばれることさえあります。つまり「最初は良かったけれど時間の経過とともに複雑化し、それほど時間がかからないうちにはっきりしない判断基準、条件、出口戦略に囚われて身動きができなくなって」しまいます。[16] ORMは良い点よりも悪い点の方が多いことを書いた本も次々と出版されています。[17]

ORMでは、オブジェクト内のデータの興味深い部分を保持できずに、データの一部が失われバラバラになり、処理のオーバーヘッドも増えます。またこれは最初の正規化処理の際に既に複数の表にデータを分割した後に発生します。前述の個人の医療記録の例に戻ると、多様な関連データの一部分ごとに、リレーショナルデータベース内の複数のテーブルに分けて入れていくことになります。複数のテーブルに対してデータを「切り分け」た後、アプリケーション層でデータを再度まとめてユーザーに提示する必要があります。このやり方だと、ビジネスエンティティ（帳票や書類など）を表現するには、オーバーヘッド、マッピング、カスタムフレームワーク、ジョインが大量に必要になります。

こういった従来のリレーショナルアーキテクチャでは、結果としてパフォーマンスが損なわれ、プログラムのバグが増える可能性があります。今日のスピーディなアプリケーション開発サイクルにはおいては、ユーザーはよりインタラクティブで対応力に優れたやり取りを求めていますが、リレーショナルモデルではうまくいきません。ミスマッチなリレーショナルモデルで何とかしようとする代わりに、抽象的でなくパフォーマンスを改善できる新しいモデルが受け入れられ始めています。

今日のデータのための新世代のデータベース

MarkLogicは、NoSQL（“Not only SQL”）データベースで、ここ30年使われてきた「一台であらゆる状況に対応」しようとするリレーショナルデータベースからのシフトの先頭に立っています。MarkLogicが今日のデータに最適であることを示す機能はたくさんありますが、特に以下の4つによってMarkLogicはユニークな存在となっています。

1. 柔軟なデータモデルで今日の多様かつ変化する分断されたデータを格納する
2. ビルトインの検索とクエリで任意の時点のデータからの価値を高める
3. スケーラビリティと弾力的な拡張・縮退で大量かつ量が変化するデータを扱う
4. エンタープライズ機能でミッションクリティカルなエンタープライズアプリケーションを実現

16　Ted Neward、ブログ「The Blog Ride」（2006年6月26日）<http://blogs.tedneward.com/2006/06/26/The+Vietnam+Of+Computer+Science.aspx>

17　Martin Fowlerの「OrmHate」<http://martinfowler.com/bliki/OrmHate.html>、Jeff Atwoodの「Object-Relational Mapping Is the Vietnam of Computer Science」、Laurie Vossの「ORM Is an Anti-Pattern」<http://seldo.com/weblog/2011/08/11/orm_is_an_antipattern>、Yegor Bugayenkoの「ORM is An Offensive Anti-Pattern」<http://www.yegor256.com/2014/12/01/orm-offensive-anti-pattern.html>など多数。

> 最初のMarkLogicプロジェクトは60日で終わりました。従来の技術では3000日かかると予想されたプロジェクトです」
>
> Paolo Pelizzoli, Global Head of Architecture, Global Technology Operations at Broadridge Financial Solutions

MARKLOGICの特徴的な機能の概要

柔軟なデータモデル

MarkLogicはマルチモデルデータベースで、JSON、XML、RDFトリプル、地理情報データ、ラージバイナリ（画像、ビデオなど）をネイティブに格納し素早くクエリできるように設計されています。この機能により、MarkLogicはリレーショナルデータベースよりも多様なデータ型ならびにデータ構造の扱いに適しています。またデータの変更に応じたデータモデルの変更も容易です。

JSONとXMLはどちらもドキュメント形式で、MarkLogicは主にこの形式でデータを格納します。リレーショナルデータベースの表とは違って、ドキュメントは人間が読んでもわかりやすく、今日の組織が扱っている多様で複雑なデータをより自然にモデリングできます。よりリッチなデータモデルを利用することで、データからより多くの価値を得ることができます。

またドキュメントは現代のアプリケーション開発に適しています。というのもミスマッチが発生しないからです。ドキュメントモデルでは、アプリケーションスタックのあらゆる層においてデータの整合性を保持できます。例えば、JSONをデータベース、アプリケーション層、UIで利用できます。これはよりアジャイルなアプローチであり、現代のwebアプリケーション開発におけるJavaScriptの拡大にも対応しています。

MarkLogicデータモデルのもう一つの長所としては、スキーマ（構造）をロード前に定義しなくてもよいということがあります。これは「スキーマ非依存」と呼ばれます。MarkLogicでは、スキーマが異なるドキュメントを格納でき、また他に影響を及ぼさずに特定のスキーマを変更できます。この強力な機能を使うと、モデルが異なるリレーショナルデータの表を素早く組み合わせることができます。これらをすべてMarkLogic内部で行います。

またMarkLogicにはグラフデータベース機能もあり、セマンティックのRDFトリプルをネイティブに格納できます。簡単に言うと、セマンティックは2つのエンティティ（人、場所、モノ）を関係性に基づいて結びつけるデータモデルで、トリプル（3つ組）の形を取ります。トリプルを相互に関連付けるとマシンで読み取り可能なグラフとなり、これによって新しいファクト（事実）を推論できます。MarkLogicでは数千億個のトリプルをJSONやXMLと一緒に、あるいはJSONやXMLの内部に格納できます。

ドキュメントストアとトリプルストアを組み合わせることができるエンタープライズデータベースは、MarkLogicだけです。このユニークな機能により、より早くより簡単にデータを最も有用な形式でモデリングでき、より大きな価値が得られるようになります。全体的に見て、MarkLogicのデータモデルはリレーショナルモデルよりもかなり柔軟です。これにより、よりスマートなアプリケーションをより短期間で開発し、変更にすぐ対応できるアジリティをもったプラットフォームが提供できます。

このような改善は、リレーショナルデータベースに比べて劇的なものとなります。「最初のMarkLogicプロジェクトは60日で終わりました。従来の技術では3000日かかると予想されたプロジェクトです」とパオロ・ペリッツオーリ氏（Broadridge Financial Solutions、Global Head of Architecture, Global Technology Operations）は述べています。

NOSQLデータベースの比較

ここ数年、変化への対応が差し迫った課題となったことから新しいデータ管理技術が爆発的に登場してきました。これらはすべて今日のデータの扱いを改善することを目的としています。さまざまなタイプのオープンソースNoSQLデータベースが大量にあります（ドキュメントストア、グラフストア、カラムストア、キーバリューストアなど）。これらには、コモディティのハードウェアによる水平拡張といった共通の特徴もありますが、実はそれぞれは大きく異なっています。NoSQLについて詳しく知りたい場合は、電子書籍『エンタープライズ NoSQL for Dummies』をご覧ください。 https://info.marklogic.com/nosql-for-dummies-jp.html から無料でダウンロードできます。

MarkLogicは他のNoSQLソリューションと同等に、ビッグデータの量、多様性、速度に対処するように設計されています。これに加えて従来のリレーショナルデータベースの信頼性の基礎となるエンタープライズ機能も備わっています」

ビルトインの検索とクエリ

データベースで、データを素早くかつ正確に検索するにはインデックスが必要です。データベースのインデックスは本の末尾にある索引と似ています。本の場合、書籍に含まれる情報がリスト化され、全部読まなくても必要な情報に素早くアクセスできます。ほとんどのデータベースにおいては、インデックス付けはデータ格納のための二次的なタスクと見なされることが一般的です。インデックス付けにおいては、どの質問に答えるためにどのようなインデックスを作成すべきか、また各インデックスによるパフォーマンスへの影響はどの程度か、またインデックス管理方法などをユーザーが決めなくてはなりませんが、これはかなり困難です。また、全文検索を行うには、全文インデックスが必要です。リレーショナルデータベースの場合、これを実現するには別途ソフトウェアを追加してこれを保守する必要があります。

一方、MarkLogicには標準機能として最高のインデックス機能ならびに全文検索が備わっています。MarkLogicでは、ロード時にデータの中身と構造にインデックスを付けます。またインデックスの種類もいろいろあり（レンジインデックス、トリプルインデックス、地理情報インデックスなど）、これらをオン／オフできます。MarkLogicのインデックスを使うとシンプルなクエリだけでなく洗練されたクエリにも容易に答えることができます。この際、JavaScript、XQuery、SPARQL（セマンティック用のクエリ言語）、またもちろんSQLといったさまざまなクエリ言語が使えます。洗練されたクエリの例としては「マイケル・ジョーダンが現役時代に一緒にプレーしたプロ選手の生涯収入とそのランキングを表示せよ。その際、ニューヨークに住んでいる人、また2015年1月以降に信頼できる情報源に掲載されたものに限る。結果を関連度に応じてランキングすること」などが考えられます。

こういった多面的な問いに答えるのは容易ではなく、これをリレーショナルデータベースで実現するのは極めて困難か不可能です。リレーショナルデータベースでは、マイケル・ジョーダンと一緒にプレーした選手との関係のモデリングや、テキストドキュメントである情報源に掲載されたものを探すのは困難です。またリレーショナルデータベースでは、Googleなどの検索エンジンのような関連度ランキングは不可能で、単純な値順で結果を返すことしかできません。

一方MarkLogicでは、このような洗練されたクエリを比較的わずかなコーディングで実現できます。関連度や全文検索の問題は、MarkLogicの豊かなデータモデルや強力なインデックスで解決できます。これらはリレーショナルデータベースにおけるSQLと同様の質問に答えられるだけでなく、他にもさまざまなことができます。質問が変わったとしても大丈夫です。MarkLogicでは将来における新しい想定されていなかったクエリにも対応できます。この際に、リレーショナルデータベースのように、ユーザーがデータやインデックスの設定を変更する必要はありません。またMarkLogicは、数百テラバイトのデータに対しても結果を1秒以下で返します。これらのデータはシステム内に一貫性と信頼性を持って保持されます。

スケーラビリティと弾力的な拡張・縮退

MarkLogicはシングルサーバーアーキテクチャに留まらず、分散システムとして大規模拡張できます。MarkLogicは「水平拡張（スケールアウト）」します。つまり、連携された複数のサーバー上で実行され、各サーバーが負荷を分担します。このアプローチでは、数百台のサーバー、数ペタバイトのデータ、数十億件のドキュメントを扱えるほか、毎秒数万件のトランザクションを処理できます。これらは廉価なコモディティハードウェアで実現されます。またオンプレミスだけでなく、AWSのようなクラウドなどあらゆる環境で利用できます。

大規模拡張は素晴らしいですが、さらに重要なのはMarkLogicの「弾力的な拡張・縮退」（エラスティシティ）です。MarkLogicのユニークなアーキテクチャにより、クラスタ内にノードを迅速かつ簡単に追加・削除することでデータベースはパフォーマンスのニーズに継続的に対応できます。またコストがかかる過剰なプロビジョニングを回避できます。ここでは複雑なシャーディングやアーキテクチャレベルでの回避策は不要で、ノードの追加・削除の際にデータは自動的にクラスタ内でリバランスされます。これも、MarkLogicの管理が簡単な理由の一つです。

エンタープライズ機能

NoSQLはちゃんとしたアプリケーションに利用できない、NoSQLはスタートアップ会社用である、あるいは重要ではないデータの置き場所である、といった誤解が一般に流布しています。しかしこれはMarkLogicには当て

> MarkLogicは米国国防総省、大手投資銀行、医療機関、国際的メディアなど、失敗が許されない業界におけるミッションクリティカルシステムでの実績があります」

はまりません。

MarkLogicには、従来のリレーショナルデータベースが持つ信頼性の基盤となる重要なエンタープライズ機能がすべて備わっていますが、これらの機能はすべてエンタープライズデータの格納と管理にとって極めて重要です。MarkLogicの主なエンタープライズ機能としては以下のものがあります。

- ACIDトランザクションによりデータの一貫性を確保し、データの消失や欠損を回避する
- 認証済みセキュリティにより、エンタープライズデータセンター内でMarkLogicを利用
- 高可用性と災害対応機能によりデータを常に利用可能
- パフォーマンスモニタリングにより、リソースのプロビジョニングと使用を監視
- エンタープライズ管理ツールにより、一般的なタスクを自動化

これらの機能は単に備わっているというだけでありません。MarkLogicはこれらすべてに関して、米国国防総省、大手投資銀行、医療機関、国際的メディアなど、失敗が許されない業界におけるミッションクリティカルシステムでの実績があります。

リーダー組織がMARKLOGICでさらに多くを実現

数百の組織が、MarkLogicを使って変化や革新に対応し、将来のビジネスを強化しています。数多くの成功事例のうち、いくつかをご紹介します。

メガバンクでのオペレーショナル取引システム

トップ5の投資銀行では、20台のリレーショナルタベースを1台のMarkLogicで置き換えました。現在、MarkLogicはデリバティブトレードストアとして利用されており、1日当たり10万件の取引を、また同時に3200万件のリアルタイムのディールを扱っています。このような大量取引では、キャッシュフローリスクが1億ドルを超えます。この銀行ではMarkLogicによって大幅なコスト削減を実現したほか、デリバティブ取引業務のグローバルでの全体像をリアルタイムで正確に把握できるようになりました。

HEALTHCARE.GOVにおける数百万人の保険申し込み

CMS（Centers for Medicare and Medicaid Services）は、HealthCare.gov（通称オバマケア）を通じて数百万人のアメリカ国民に健康保険を紹介しています。保険プランに対して、国の複数のデータソースに含まれている個々人の加入要件を検証するため、数十万人の同時接続を実現しています。またデータ損失も皆無です。このシステムを使って、最初の2年間で1200万人のアメリカ国民が健康保険に申し込みました。この成功は、ある州で健康保険システムが失敗したのとは極めて対照的です。この失敗により、この州はリレーショナルデータベースのベンダー相手に裁判を起こしています。[18]

BBCのIPLAYERストリーミングサービスのパフォーマンスを改善

iPlayerは、英国BBCのテレビストリーミングサービスです。現時点で、iPlayer上の番組の中には30億回以上視聴されているものもあります。BBCは大規模な拡張とパフォーマンス要件に対応するため、番組のメタデータを格納・提供する主要コンポーネントをリレーショナルからMarkLogicに移行しました。BBCでは既に2012年のオリンピックの際にMarkLogicでのダイナミックなコンテンツ配信プラットフォームの構築に成功しており、iPlayerでもMarkLogicの拡張性と柔軟性を活用したいと考えました。稼働後、SQLでは20秒だったクエリがMarkLogicでは20ミリ秒と大幅に改善されました。

リレーショナルを超えてデータを活用する

古いやり方から新しいやり方への移行は、最初は非常に面倒に思えるかもしれません。このため、まずは小規模のプロジェクトから始めて徐々に拡大していった方が良いでしょう。NoSQLへの移行計画立案の際に参考になる推奨事項をいくつか挙げておきます。

推奨される次のステップ

緊急性があると認識する

現状に満足しているためにNoSQLの導入が進まない場合、変化が必要なことを認識しないと先に進みません。

18　Shelby Stebens、Reuters、「Oracle sues Oregon officials in healthcare website dispute」（2015年2月27日）<http://www.reuters.com/article/2015/02/27/us-usa-healthcare-oregon-idUSKBN0LV2LK20150227>

> MarkLogicでは製品開発期間が短縮され、業務部門とIT部門の関係も強化されます。この場合、ITは解決策をもたらすビジネス上重要な部門と認識されます。解決の障害とはみなされません」
>
> Andrea Powell, CIO of CABI (The Centre for Biosciences and Agriculture International))

緊急性があることを共有することで、ビジョンが実現され賛同が得られます。

適切なチームを結成する

成功するためには素晴らしいテクノロジーだけでなく、素晴らしい人々とプロセスが必要です。ここでは組織内の意思決定者ならびに実装担当者を特定することが重要で、またプロジェクト実現を妨げているのは何かを理解する必要があります。

適切なプロジェクトから始める

適切な利用法から始めることが極めて重要です。通常は、小規模ながら十分なインパクトがあるもので、他への影響がないものが最適です。MarkLogicのエキスパートが、MarkLogic利用に最適なプロジェクトの選択をお手伝いします。

早い段階でMARKLOGICに協力を依頼する

誰よりもNoSQLに詳しいMarkLogicのエキスパートが担当してお手伝いできます。こういった援助が極めて大切な、開発の早い段階から参加させていただくのが重要です。

さらに詳しく

- 電子書籍『エンタープライズNoSQL for Dummies』日本語版：NoSQLデータベースの概要を紹介する無料の電子書籍。
  http://info.marklogic.com/nosql-for-dummies-jp.html
- 『MarkLogicデータベースの概要』– MarkLogicのユニークな機能のご紹介。
  https://jp.marklogic.com/product/marklogic-database-overview/
- ホワイトペーパー『インサイドMarkLogicサーバー』：MarkLogicの強力な機能を実現している内部構造をご紹介。
  http://jp.marklogic.com/resources/inside-marklogic-server-jp/
- 弊社との打ち合わせ：具体的な案件については、MarkLogicの営業担当者までお問い合わせください。MarkLogic-JP@marklogic.com

© 2017 MARKLOGIC CORPORATION. ALL RIGHTS RESERVED. このテクノロジーは、米国特許番号 7,127,469B2、米国特許番号 7,171,404B2、米国特許番号7,756,858 B2、米国特許番号7,962,474 B2で保護されています。MarkLogicは、米国およびその他の国におけるMarkLogic Corporationの商標または登録商標です。本書に記載されているその他の商標は、各企業の所有物です。

MARKLOGIC K.K.

150-0043 東京都渋谷区道玄坂 1-12-1渋谷マークシティウェスト 22 階

+81 3 4360 5354 | jp.marklogic.com | MarkLogic-JP@marklogic.com

マークロジック株式会社 MARKLOGIC K.K.
150-0043 東京都渋谷区道玄坂 1-12-1渋谷マークシティウェスト 22 階 | +81 3 4360 5354
jp.marklogic.com | MarkLogic-JP@marklogic.com