(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7465870号
(P7465870)

(45)発行日　令和6年4月11日(2024.4.11)

(24)登録日　令和6年4月3日(2024.4.3)

(51)国際特許分類
Ｇ０６Ｆ　16/28　(2019.01)

ＦＩ
Ｇ０６Ｆ　16/28

請求項の数　13　（全25頁）

(21)出願番号　特願2021-521186(P2021-521186)
(86)(22)出願日　令和1年10月15日(2019.10.15)
(65)公表番号　特表2022-505230(P2022-505230A)
(43)公表日　令和4年1月14日(2022.1.14)
(86)国際出願番号　PCT/US2019/056363
(87)国際公開番号　WO2020/081586
(87)国際公開日　令和2年4月23日(2020.4.23)
審査請求日　令和4年7月26日(2022.7.26)
(31)優先権主張番号　201841039503
(32)優先日　平成30年10月18日(2018.10.18)
(33)優先権主張国・地域又は機関
インド(IN)
(31)優先権主張番号　16/439,532
(32)優先日　令和1年6月12日(2019.6.12)
最終頁に続く

(73)特許権者　502303739
オラクル・インターナショナル・コーポレイション
アメリカ合衆国カリフォルニア州９４０６５レッドウッド・シティー，オラクル・パークウェイ５００
(74)代理人　110001195
弁理士法人深見特許事務所
(72)発明者　カトカデ，プラルハド
アメリカ合衆国、９４０６５ カリフォルニア州、レッドウッド・シティー、オラクル・パークウェイ、５００、オラクル・インターナショナル・コーポレイション
(72)発明者　ライクマン，ナタリア
最終頁に続く

(54)【発明の名称】　多次元データベース環境における依存関係分析のためのシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
多次元データベースにおける依存関係分析のためのシステムであって、
１つまたは複数のマイクロプロセッサを含むコンピュータと、
前記コンピュータ上で動作する多次元データベースサーバとを備え、前記多次元データベースサーバは、少なくとも１つの多次元データベースキューブをサポートし、前記多次元データベースは、複数の次元を備え、前記複数の次元の各々は、複数のメンバを備え、前記システムはさらに、
動的メンバを備え、前記動的メンバは、一組の前記複数のメンバに依存しており、
前記システムは、依存関係分析を実行して、前記動的メンバが依存している前記一組の前記複数のメンバを判断し、
前記依存関係分析は、１つまたは複数のマイクロプロセッサを含む前記コンピュータにステップを実行させ、前記ステップは、
前記動的メンバをトークン化することにより、前記動的メンバの１つまたは複数のトークンを生じさせるステップと、
前記動的メンバの前記１つまたは複数のトークンの各々を分析するステップと、
前記動的メンバの前記１つまたは複数のトークンの各々の前記分析に基づいて、前記動的メンバが依存している前記一組の前記複数のメンバが、静的な依存要素のみを備えているか、それとも１つまたは複数の実行時依存要素を備えているかを判断するステップとを備える、システム。

【請求項２】
前記依存関係分析の実行は、前記少なくとも１つの多次元データベースキューブをサポートする前記多次元データベースサーバの起動動作時に実行される、請求項１に記載のシステム。
【請求項３】
前記依存関係分析は、１つまたは複数のマイクロプロセッサを含む前記コンピュータにステップを実行させ、前記ステップは、
前記動的メンバの前記１つまたは複数のトークンの各々の前記分析に基づいて、前記動的メンバが依存している前記一組の前記複数のメンバが、静的な依存要素のみを備えていると判断するステップをさらに備え、
前記判断に基づいて、前記動的メンバが依存している全ての静的な依存要素は、フェッチされて入力オドメータに転送される、請求項１または２に記載のシステム。
【請求項４】
前記依存関係分析は、１つまたは複数のマイクロプロセッサを含む前記コンピュータにステップを実行させ、前記ステップは、
前記動的メンバの前記１つまたは複数のトークンの各々の前記分析に基づいて、前記動的メンバが依存している前記一組の前記複数のメンバが、１つまたは複数の実行時依存要素を備えていると判断するステップをさらに備え、
前記動的メンバが１つまたは複数の実行時依存要素に依存しているとの前記判断に基づいて、前記１つまたは複数の実行時依存要素のアレイが生成され、
前記１つまたは複数の実行時依存要素の前記生成されたアレイに基づいて、前記多次元データベースキューブの実行時依存要素メンバのリストが生成される、請求項１または２に記載のシステム。
【請求項５】
前記動的メンバに対するクエリは、前記多次元データベースキューブ上で実行される、請求項４に記載のシステム。
【請求項６】
前記動的メンバに対する前記クエリは、前記多次元データベースキューブの前記実行時依存要素メンバの前記リストを利用して、前記実行時依存要素メンバのリスト上の前記実行時依存要素メンバの各々に関連付けられた値をフェッチする、請求項５に記載のシステム。
【請求項７】
多次元データベースにおける依存関係分析のための方法であって、
１つまたは複数のマイクロプロセッサを含むコンピュータが、前記コンピュータ上で動作する多次元データベースサーバを設けるステップを備え、前記多次元データベースサーバは、少なくとも１つの多次元データベースキューブをサポートし、前記多次元データベースは、複数の次元を備え、前記複数の次元の各々は、複数のメンバを備え、前記方法はさらに、
１つまたは複数のマイクロプロセッサを含む前記コンピュータが、動的メンバを設けるステップを備え、前記動的メンバは、一組の前記複数のメンバに依存しており、前記方法はさらに、
前記コンピュータが、依存関係分析を実行して、前記動的メンバが依存している前記一組の前記複数のメンバを判断するステップを備え、
前記依存関係分析の実行は、
前記動的メンバをトークン化することにより、前記動的メンバの１つまたは複数のトークンを生じさせるステップと、
前記動的メンバの前記１つまたは複数のトークンの各々を分析するステップと、
前記動的メンバの前記１つまたは複数のトークンの各々の前記分析に基づいて、前記動的メンバが依存している前記一組の前記複数のメンバが、静的な依存要素のみを備えているか、それとも１つまたは複数の実行時依存要素を備えているかを判断するステップとを備

える、方法。
【請求項８】
前記コンピュータが、前記少なくとも１つの多次元データベースキューブをサポートする前記多次元データベースサーバの起動動作時に前記依存関係分析を実行するステップをさらに備える、請求項７に記載の方法。
【請求項９】
前記依存関係分析の実行は、
前記動的メンバの前記１つまたは複数のトークンの各々の前記分析に基づいて、前記動的メンバが静的な依存要素のみに依存していると判断するステップをさらに備え、
前記方法はさらに、
前記コンピュータが、前記動的メンバが静的な依存要素のみに依存しているとの前記判断に基づいて、前記動的メンバが依存している全ての静的な依存要素をフェッチして入力オドメータに転送するステップを備える、請求項７または８に記載の方法。
【請求項１０】
前記依存関係分析の実行は、
前記動的メンバの前記１つまたは複数のトークンの各々の前記分析に基づいて、前記動的メンバが１つまたは複数の実行時依存要素に依存していると判断するステップをさらに備え、
前記方法はさらに、
前記コンピュータが、前記動的メンバが１つまたは複数の実行時依存要素に依存しているとの前記判断に基づいて、前記１つまたは複数の実行時依存要素のアレイを生成するステップと、
前記コンピュータが、前記１つまたは複数の実行時依存要素の前記生成されたアレイに基づいて、前記多次元データベースキューブの実行時依存要素メンバのリストを生成するステップとを備える、請求項７または８に記載の方法。
【請求項１１】
前記コンピュータが、前記動的メンバに対するクエリを前記多次元データベースキューブ上で実行するステップをさらに備える、請求項１０に記載の方法。
【請求項１２】
前記動的メンバに対する前記クエリは、前記多次元データベースキューブの前記実行時依存要素メンバの前記リストを利用して、前記実行時依存要素メンバのリスト上の前記実行時依存要素メンバの各々に関連付けられた値をフェッチする、請求項１１に記載の方法。
【請求項１３】
請求項７～１２のいずれか１項に記載の方法をコンピュータに実行させるプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
著作権表示
この特許文献の開示の一部は、著作権保護の対象となる題材を含んでいる。著作権保有者は、この特許文献または特許開示が特許商標庁の包袋または記録に掲載されているため、この特許文献または特許開示を誰でも複写複製できることに対して異議はないが、その他の点では全ての如何なる著作権をも保有する。
【０００２】
優先権の主張
本願は、２０１９年６月１２日に出願された出願番号第１６／４３９，５３２号である「多次元データベース環境における依存関係分析のためのシステムおよび方法（SYSTEM AND METHOD FOR DEPENDENCY ANALYSIS IN A MULTIDIMENSIONAL DATABASE ENVIRONMENT）」と題される米国特許出願、および、２０１８年１０月１８日に出願された出願番号第２０１８４１０３９５０３号である「多次元データベース環境における依存関係分析のためのシステムおよび方法（SYSTEM AND METHOD FOR DEPENDE

NCY ANALYSIS IN A MULTIDIMENSIONAL DATABASE ENVIRONMENT）」と題されるインド特許出願に対する優先権の利益を主張し、これらの出願は引用によって本明細書に援用される。

【０００３】

発明の分野：

本発明の実施形態は、一般にデータベースおよびデータウェアハウジングに関し、特に多次元データベース環境における依存関係分析のためのシステムおよび方法に関する。

【背景技術】

【０００４】

背景：

多次元データベースコンピューティング環境は、企業が重要なビジネス情報を適切な人々に必要なときに届けることを可能にし、複数の既存のデータソースからのデータを活用および統合し、フィルタリングされた情報をエンドユーザコミュニティに、それらのユーザのニーズに最も適合する形式で配信する能力を含む。ユーザは、リアルタイムで、なじみのあるビジネス次元に沿ってデータと対話して利用することができ、思考スピードアナリティクスが可能になる。これらは、本発明の実施形態を使用することができるタイプの環境のいくつかの例である。

【発明の概要】

【課題を解決するための手段】

【０００５】

概要：

実施形態によれば、ある実施形態に係る多次元データベースにおける依存関係分析のためのシステムおよび方法が本明細書に記載されている。動的メンバは、同一のアウトラインからの他のメンバ（式の依存要素と呼ばれる）に依存していてもよく、当然のことながらこれらの依存要素は、各交点の元の式を評価するために最初に計算されなければならない。したがって、ハイブリッドデータ集約モデルにおいて動的メンバの式を評価するために、システムはまず、この式が依存している全てのメンバのリストを準備し得る。さらに、動的メンバの依存要素のリストは、「実行時依存要素」および「静的な依存要素」として分類されてもよい。動的メンバの実行時依存要素は、各交点について異なっている依存要素であるのに対して、静的な依存要素は、交点に関係なく一定である依存要素である。動的メンバの実行時依存要素および静的な依存要素を識別するこの分析プロセスは、ハイブリッドデータ集約モデルにおける依存関係分析と称される。

【０００６】

実施形態によれば、多次元データベースにおける依存関係分析のための例示的な方法は、１つまたは複数のマイクロプロセッサを含むコンピュータに、上記コンピュータ上で動作する多次元データベースサーバを設けることができ、上記多次元データベースサーバは、少なくとも１つの多次元キューブをサポートし、上記多次元データベースは、複数の次元を備え、上記複数の次元の各々は、複数のメンバを備え、上記方法はさらに、１つまたは複数のマイクロプロセッサを含む上記コンピュータに、動的メンバを設けることができ、上記動的メンバは、一組の上記複数のメンバに依存している。上記方法はさらに、依存関係分析を実行して、上記動的メンバが依存している上記一組の上記複数のメンバを判断することができる。

【０００７】

実施形態によれば、上記依存関係分析の実行は、上記動的メンバをトークン化することにより、上記動的メンバの１つまたは複数のトークンを生じさせるステップを備えることができる。そして、上記依存関係分析は、上記動的メンバの上記１つまたは複数のトークンの各々を分析し、上記動的メンバの上記１つまたは複数のトークンの各々の上記分析に基づいて、上記動的メンバが静的な依存要素のみに依存していると判断することができる。このような場合、上記方法は、上記動的メンバが静的な依存要素のみに依存しているとの上記判断に基づいて、上記動的メンバが依存している全ての静的な依存要素をフェッチ

して入力オドメータに転送することができる。

【０００８】

実施形態によれば、上記依存関係分析の実行は、上記動的メンバをトークン化することにより、上記動的メンバの１つまたは複数のトークンを生じさせるステップを備えることができる。そして、上記依存関係分析は、上記動的メンバの上記１つまたは複数のトークンの各々を分析し、上記動的メンバの上記１つまたは複数のトークンの各々の上記分析に基づいて、上記動的メンバが１つまたは複数の実行時依存要素に依存していると判断することができる。そして、上記方法は、上記動的メンバが１つまたは複数の実行時依存要素に依存しているとの上記判断に基づいて、上記１つまたは複数の実行時依存要素のアレイを生成することができる。また、上記方法は、上記１つまたは複数の実行時依存要素の上記生成されたアレイに基づいて、上記多次元データベースキューブの実行時依存要素メンバのリストを生成することができる。

【図面の簡単な説明】

【０００９】

【図１】ある実施形態に係る、多次元データベース環境の一例を示す図である。

【図２】ある実施形態に係る、多次元データベースと動的フローとの併用を示す図である。

【図３】ある実施形態に係る、多次元データベースと動的フローとの併用をさらに示す図である。

【図４】ある実施形態に係る、例示的な関数スタックを示す図である。

【図５】ある実施形態に係る、例示的なデータセットを示す図である。

【図６】ある実施形態に係る、例示的なデータセットを示す図である。

【図７】ある実施形態に係る、依存関係分析のための例示的な方法のフローチャートである。

【図８】ある実施形態に係る、依存関係分析のための例示的な方法のフローチャートである。

【発明を実施するための形態】

【００１０】

詳細な説明：

上記は、他の特徴とともに、同封の明細書、特許請求の範囲および図面を参照すると明らかになるであろう。具体的詳細は、さまざまな実施形態を理解できるようにするために記載されている。しかし、これらの具体的詳細がなくてもさまざまな実施形態を実施できることは明らかであろう。同封の明細書および図面は、限定的であるよう意図されてはいない。

【００１１】

多次元データベース環境は、その一例がＯｒａｃｌｅ　Ｅｓｓｂａｓｅを含んでいるのだが、この多次元データベース環境を使用して、場合によっては複数のデータソースからの大量のデータを統合して、フィルタリングされた情報をエンドユーザに、それらのユーザの特定の要求に応えるような態様で配信することができる。

【００１２】

図１は、ある実施形態に係る、多次元データベース環境１００の一例を示す図である。

図１に示されるように、ある実施形態に従うと、データベース層として動作する多次元データベース環境は、１つまたは複数の多次元データベースサーバシステム１０２を含み得て、１つまたは複数の多次元データベースサーバシステム１０２の各々は、物理コンピュータリソースまたはコンポーネント１０４（たとえば、マイクロプロセッサ／ＣＰＵ、物理メモリ、ネットワークコンポーネント）と、オペレーティングシステム１０６と、１つまたは複数の多次元データベースサーバ１１０（たとえば、Ｅｓｓｂａｓｅサーバ）とを含み得る。

【００１３】

ある実施形態に従うと、中間層１２０は、たとえばプロバイダサービス１２２（たとえば、ハイペリオンプロバイダサービス）、管理サービス１２４（たとえば、Ｅｓｓｂａｓ

ｅ管理サービス）またはスタジオ／統合サービス１２６（たとえば、Ｅｓｓｂａｓｅスタジオ／Ｅｓｓｂａｓｅ統合サービス）などの１つまたは複数のサービスを含み得る。中間層は、多次元データベース環境とともに使用するために、ＯＤＢＣ／ＪＤＢＣ１２７，１２８または他のタイプのインターフェイスを介して、メタデータカタログ１２９および／または１つまたは複数のデータソース１３０（たとえば、リレーショナルデータベース）へのアクセスを提供することができる。

【００１４】

ある実施形態に従うと、１つまたは複数のデータソースは、多次元データベースの提供に使用するために、ＯＤＢＣ／ＪＤＢＣ１３２または他のタイプのインターフェイスを介して、１つまたは複数の多次元データベースサーバによってアクセス可能でもある。

【００１５】

ある実施形態に従うと、クライアント層１４０は、多次元データベース（たとえば、スマートビュー、スプレッドシートアドイン、スマートサーチ、管理サービス、ＭａｘＬ、ＸＭＬＡ、ＣＡＰＩまたはＶＢ　ＡＰＩアプリケーション、オラクル・ビジネス・インテリジェンス・エンタープライズ・エディション・プラス、または他のタイプの多次元データベースクライアントなど）へのアクセスを可能にする１つまたは複数の多次元データベースクライアント１４２（たとえば、Ｅｓｓｂａｓｅサーバクライアント）を含み得る。クライアント層は、たとえば管理サービスコンソール１４４またはスタジオ／統合サービスコンソール１４６などの、中間層におけるサービスとともに使用するためのコンソールも含み得る。

【００１６】

ある実施形態に従うと、クライアント層と中間層とデータベース層との間の通信は、ＴＣＰ／ＩＰ、ＨＴＴＰまたは他のタイプのネットワーク通信プロトコルのうちの１つ以上によって提供することができる。

【００１７】

ある実施形態に従うと、多次元データベースサーバは、１つまたは複数のデータソースからのデータを統合して、多次元データベース、データ構造またはキューブ１５０を提供することができ、次いで、多次元データベース、データ構造またはキューブ１５０にアクセスして、フィルタリングされた情報をエンドユーザに提供することができる。

【００１８】

一般に、多次元データベースにおける各データ値は、キューブの１つのセルに格納され、特定のデータ値は、その座標をキューブの次元に沿って指定することによって参照可能である。１つの次元からのメンバの、１つまたは複数の他の次元の各々からのメンバとの交点がデータ値を表す。

【００１９】

たとえば、売上げ志向型ビジネスアプリケーションで使用され得るキューブ１６２を示す図１に示されるように、クエリが「売上高」を示す場合、このシステムは、このクエリを、全ての「売上高」データ値を含むデータベース内のデータ値１６４のスライスまたは層として解釈することができ、「売上高」は、「実績」および「予算」と交差している。多次元データベースにおける特定のデータ値１６６を参照するために、クエリは、たとえば「売上高、実績、１月」を指定することによって各次元上のメンバを指定することができる。データベースをさまざまにスライスすることによって、データのさまざまな観点が提供される。たとえば、「２月」のデータ値のスライス１６８は、時期次元が「２月」に固定されたそれらのデータ値の全てを検討する。

【００２０】

データベースアウトライン

ある実施形態に従うと、多次元データベースの開発は、データベースアウトラインの作成から始まり、このデータベースアウトラインは、データベース内のメンバ間の構造的関係を定義し、データベース内のデータを編成し、連結および数学的関係を定義する。データベースアウトラインの階層ツリーまたはデータ構造内に、各次元は、１つまたは複数の

メンバを備え、これらの１つまたは複数のメンバは、他のメンバをさらに備えていてもよい。次元の仕様は、その個々のメンバの値をどのように連結するかをシステムに指示する。連結は、ツリーのブランチの中の一群のメンバである。

【００２１】

次元およびメンバ

ある実施形態に従うと、次元は、データベースアウトラインにおける最も高い連結レベルを表す。部門機能（たとえば、時間、勘定、製品ライン、市場、事業部）に関連するビジネスプランの構成要素を表すために標準次元が選択されてもよい。標準次元に関連付けられる属性次元は、ユーザがメンバ属性または特徴に基づいて標準次元のメンバをグループ化して分析することを可能にする。メンバ（たとえば、製品Ａ、製品Ｂ、製品Ｃ）は、次元の個々の構成要素である。

【００２２】

次元およびメンバ関係

ある実施形態に従うと、多次元データベースは、家族に関する用語（親、子、兄弟、子孫および先祖）および階層に関する用語（世代およびレベル、ルートおよびリーフ）を使用して、データベースアウトライン内のメンバの役割および関係を説明する。

【００２３】

ある実施形態に従うと、親は、その下にブランチを有するメンバである。たとえば、「利幅」は、「売上高」および「売上原価」（ＣＯＧＳ）の親であってもよい。子は、その上に親を有するメンバである。上記の例では、「売上高」および「売上原価」が、親である「利幅」の子である。兄弟は、同一世代内の子であって同一の直接の親の子である。

【００２４】

ある実施形態に従うと、子孫は、親の下のブランチの中のメンバである。たとえば、「利益」、「在庫」および「比率」は、メジャーの子孫であってもよく、この場合、「利益」、「在庫」および「比率」の子もメジャーの子孫である。先祖は、メンバの上のブランチの中のメンバである。上記の例では、「利幅」、「利益」およびメジャーが「売上高」の先祖であってもよい。

【００２５】

ある実施形態に従うと、ルートは、ブランチの中の最上位のメンバである。たとえば、メジャーは、「利益」、「在庫」および「比率」のルートであってもよく、したがって、「利益」、「在庫」および「比率」の子のルートであってもよい。リーフ（レベル０）メンバは、子を持たない。たとえば、期首「在庫」、追加および期末「在庫」がリーフメンバであってもよい。

【００２６】

ある実施形態に従うと、世代とは、次元内の連結レベルのことをいう。ツリーのルートブランチは、「世代１」であると考えられ、世代番号は、ルートからリーフメンバに向かって大きくなっていく。レベルとは、次元内のブランチのことをいい、世代で使用された番号順序とは逆に番号付けられ、レベル番号は、リーフメンバからそのルートに向かって小さくなっていく。

【００２７】

ある実施形態に従うと、ユーザは、世代またはレベルに名前を割り当てて、その名前をその世代またはレベル内の全てのメンバの省略表現として使用することができる。

【００２８】

疎および密な次元

多次元データベース内のデータセットは、多くの場合、２つの特徴を共有する。すなわち、データがスムーズかつ均一に配信されないこと、および、メンバの組み合わせの大部分についてはデータが存在しないことである。

【００２９】

ある実施形態に従うと、これに対処するために、このシステムは、２つのタイプの標準次元、すなわち疎な次元および密な次元を認識することができる。疎な次元は、利用可能

なデータ位置が埋められる割合が比較的低い次元であり、密な次元は、次元のあらゆる組み合わせにおいて１つまたは複数のセルが占められる確率が比較的高い次元である。多くの多次元データベースは、メンバの組み合わせの大部分についてデータ値を欠いているという点において、元来疎である。

【００３０】

データブロックおよびインデックスシステム

ある実施形態に従うと、多次元データベースは、データブロックおよびインデックスを使用して、データを格納してデータにアクセスする。このシステムは、疎な標準次元のメンバの固有の各組み合わせについて多次元アレイまたはデータブロックを作成することができ、各データブロックは、その疎な次元のメンバの組み合わせのための密な次元のメンバを表す。各データブロックについてインデックスが作成され、このインデックスは、疎な標準次元のメンバの組み合わせを表し、少なくとも１つのデータ値が存在する疎な標準次元のメンバの固有の各組み合わせについて入力またはポインタを含む。

【００３１】

ある実施形態に従うと、多次元データベースサーバは、データ値を検索する際、インデックスによって提供されるポインタを使用して、適切なデータブロックを突き止め、そのデータブロック内でデータ値を含むセルを突き止めることができる。

【００３２】

管理サービス

ある実施形態に従うと、管理サービス（たとえば、Ｅｓｓｂａｓｅ管理サービス）は、ユーザがサーバ、アプリケーションおよびデータベースを設計、開発、維持および管理することを可能にする単一アクセスポイントを提供する。

【００３３】

スタジオ

ある実施形態に従うと、スタジオ（たとえば、Ｅｓｓｂａｓｅスタジオ）は、データモデル化、キューブ設計および分析アプリケーション構築に関連するタスクを実行するためのウィザード方式のユーザインターフェイスを提供する。

【００３４】

スプレッドシートアドイン

ある実施形態に従うと、スプレッドシートアドインは、多次元データベースをスプレッドシートと統合して、接続、ピボット、ドリルダウンおよび計算などの高度なコマンドをサポートする。

【００３５】

統合サービス

ある実施形態に従うと、統合サービス（たとえば、Ｅｓｓｂａｓｅ統合サービス）は、多次元データベースに格納されたデータとリレーショナルデータベースに格納されたデータとの間の統合に使用するためのメタデータ方式の環境を提供する。

【００３６】

プロバイダサービス

ある実施形態に従うと、プロバイダサービス（たとえば、ハイペリオンプロバイダサービス）は、Ｊａｖａ（登録商標）ＡＰＩ、スマートビューおよびＸＭＬＡクライアントのためのデータソースプロバイダとして動作する。

【００３７】

スマートビュー

ある実施形態に従うと、スマートビューは、たとえばハイペリオン・フィナンシャル・マネジメント（Hyperion Financial Management）、ハイペリオン・プランニング（Hyperion Planning）およびハイペリオン・エンタープライズ・パフォーマンス・マネジメント・ワークスペース（Hyperion Enterprise Performance Management Workspace）データのための共通のインターフェイスを提供する。

【００３８】

開発者製品
ある実施形態に従うと、開発者製品は、カスタマイズされた企業分析アプリケーションの迅速な作成、管理および実装を可能にする。
【００３９】
ライフサイクル管理
ある実施形態に従うと、ライフサイクル管理（たとえば、ハイペリオン・エンタープライズ・パフォーマンス・マネジメント・システム・ライフサイクル・マネジメント（Hyperion Enterprise Performance Management System Lifecycle Management））は、企業パフォーマンス管理製品がアプリケーション、リポジトリまたは個々の人工物を製品環境全体にわたって移動させることを可能にするための手段を提供する。
【００４０】
ＯＬＡＰ
ある実施形態に従うと、オンライン分析処理（ＯＬＡＰ）は、ユーザが企業データを分析することを可能にする環境を提供する。たとえば、財務部門は、予算編成、活動基準原価計算、財務成績分析および財務モデリングなどのアプリケーションにＯＬＡＰを使用して、「ジャストインタイム」の情報を提供することができる。
【００４１】
ある実施形態に従うと、ＯＬＡＰシステムは、複数の次元においてデータを編成することができ、データセットの検索者／ユーザが、さまざまな次元を横断する有向検索を行って最終的に対象の結果に到達することを可能にする。ＯＬＡＰシステムは、データを次元の交点にあるものと見なすことができる。言い換えれば、ＯＬＡＰシステムの根底にあるデータは、全ての次元のクロス積の具体例である多次元データベースとして編成されて格納されることができる。これにより、ユーザ／検索者は、アドホックな態様で対象の次元に沿って詳細の階層を横断して、特定の対象データに到達することができる。緩やかに変化するデータは、現在のデータセット内のメタデータとして表すことができる。
【００４２】
ハイブリッド多次元データベース
ある実施形態に従うと、このシステムは、多次元データベース（たとえば、Ｅｓｓｂａｓｅ）コンピューティング環境における動的フロー（本明細書では、いくつかの例では、クエリ処理動的フロー（ＱＰＤＦ）と称される）の使用をサポートする。この動的フロープロセスは、たとえば集約ストレージオプション（ＡＳＯ）、ブロックストレージオプション（ＢＳＯ）または他のタイプのストレージコンテナのハイブリッド使用を可能にし、受信された入力クエリをボトムアップモードで処理するための一般的なフローを提供する。このアプローチを使用してキューブのサイズを小さくすることができ、動的メンバの効率的な計算を提供する。
【００４３】
たとえば、ある実施形態に従うと、疎な動的メンバにアクセスするクエリの場合、このシステムは、集約ストレージエンジンを使用して要求を満たすことができる。集約ストレージエンジンによって処理できないクエリの場合、このシステムは、ブロックストレージエンジンを利用して要求を満たすことができ、これは、たとえばデータを集約ストレージ一時表領域に持って行くことを含む。
【００４４】
たとえば、ある実施形態に従うと、動的フローは、コンピュータシステムによって実行されると、多次元データベース上で動作して、（１）入力クエリを拡張して全ての基本／計算データを求め、（２）拡張されたクエリを分析して、依存関係および計算順序を求め、（３）先行するステップに従って計算ユニットを定義し、（４）定義された計算ユニットを用いて処理フローを構築して、それらを接続し、（５）処理フローを実行して、入力クエリに対する応答を決定することができる。
【００４５】
図２は、ある実施形態に係る、多次元データベースと動的フローとの併用を示す図であ

る。
【００４６】
一般的な多次元環境では、入力クエリに応答するようにシステムを準備するために、データベースサーバは、特定の次元について値を事前に計算し、それらの事前に計算された値を、その後のルックアップのためにキューブに格納する。
【００４７】
ある実施形態に従うと、動的フローが代わりに使用される場合には、動的クエリ処理をサポートする能力は、データベースサーバがこのような値の事前計算および格納を回避することを可能にするため、パフォーマンスを向上させて、空である可能性があるセルの保管を減少させる。
【００４８】
図２に示されるように、ある実施形態に従うと、このシステムは、１つまたは複数のクエリプロセッサ２００、たとえば多次元式（ＭＤＸ）クエリプロセッサ２０２および／またはスプレッドシートエクストラクタ（ＳＳＥ）２０４クエリプロセッサを含み得て、これらのクエリプロセッサは、クライアントからの入力クエリ２０８の受信２０６を可能にして、データソースからの一組のデータ（多次元データベースによって提供されて多次元データベースを介してアクセス可能にされる）を取得したり、アクセスしたり、調べたりする。
【００４９】
ある実施形態に従うと、プリプロセッサコンポーネント２１０は、データ取得レイヤ２１２またはデータフェッチコンポーネント（いくつかの環境では、カーネルベースのオドメータレトリーバ（検索部）、またはメモリに格納されたオドメータもしくはデータ構造を組み込むことができ、カーネルベースのオドメータレトリーバまたはオドメータもしくはデータ構造は、データブロックへのポインタを管理したり、制御情報を入れたり、格納されたメンバへの複数のポインタの複数のアレイのうちの１つのアレイの役割を果たす）と、アグリゲータコンポーネント２１４と、カルキュレータコンポーネント２１６とを含み得て、これらのレイヤおよびコンポーネントの各々は、コンピュータシステムによって実行可能なソフトウェアまたはプログラムコードとして提供されることができる。
【００５０】
一般に、ある実施形態に従うと、プリプロセッサは、多次元データベースに対する処理のために、１つまたは複数のクエリプロセッサから入力クエリを受信する（２１８）。アグリゲータは、データの階層的集約を実行するように適合されている。カルキュレータは、データに対して計算を実行し、以下でさらに説明するようにアグリゲータと協働してデータ取得レイヤ（適宜オドメータを含む）をキューブ内での入力および／または検索のうちの少なくとも１つに利用し、入力クエリに対する応答を処理するように適合されている。
【００５１】
ある実施形態に従うと、このシステムは、たとえば集約ストレージオプション（ＡＳＯ）２２２、ブロックストレージオプション（ＢＳＯ）２２４または他のタイプのストレージコンテナ２２６のうちの１つ以上などの１つまたは複数のストレージコンテナ２２０を含み得て、これらのストレージコンテナの各々は、データソースまたは多次元データベースに対して読み書きされたデータ２３０間のインターフェイスとしての役割を果たすことができ、どちらのデータがプリプロセッサでの集約および計算に必要であってもそのような役割を果たすことができる。
【００５２】
図３は、ある実施形態に係る、多次元データベースと動的フローとの併用をさらに示す図である。
【００５３】
図３に示されるように、ある実施形態に従うと、データベースサーバが入力クエリを受信したことに応答して、アグリゲータは、カルキュレータと連係して動作して（２４０，２４２）、動的フロー２４４の一部としてクエリを処理することができ、これらも同様に

、コンピュータシステムによって実行可能なソフトウェアまたはプログラムコードとして提供されることができる。

【００５４】

たとえば、図３に示されるように、動的フロープロセスは、この例では１つまたは複数のＡＳＯ、ＢＳＯまたは他のタイプのストレージコンテナのハイブリッド使用を可能にし、これらのストレージコンテナを使用してボトムアップモードでクエリを処理するための一般的なフローを提供する。

【００５５】

ある実施形態に従うと、このシステムは、入力クエリを処理し始めると、まず、入力クエリを調べることによって、どの特定のデータまたは他の情報、すなわちメタデータを取得する必要があるかを判断する。次いで、システムは、その入力クエリについて最初の計算ユニット２５０を定義することができ（２４６）、この最初の計算ユニット２５０は、ストレージコンテナから一組のデータを取得する集約／計算プロセスを封入している。

【００５６】

ある実施形態に従うと、計算ユニットによる使用のために、データバッファ２６０（本明細書では、いくつかの例では、１つまたは複数の出力バケットと称される）は、各計算ユニットがデータを読み／書きする（２５２）ことができ、かつ、ストレージコンテナから受信された（２５４）データの一時保管を可能にするデータ構造として動作する。

【００５７】

ある実施形態に従うと、動的フローがＢＳＯタイプのストレージコンテナとともに使用される場合、動的フロープロセスは、入力クエリの事前分析および要求されたポイントのその基本データへの拡張を実行する。

【００５８】

しかし、このような拡張された基本データの量はかなり多いであろう。

これに対処するため、および拡張されたデータの量を減らすために、ある実施形態に従うと、関連付けられたカーネル構造（たとえば、上記のカーネル側のオドメータなど）の十分な拡張なしに、カーネルからのデータのフェッチ中に第１の動的集約を実行することができる。

【００５９】

ある実施形態に従うと、次いで、動的フローは、入力クエリを拡張し、全ての基本／計算データを求め、拡張されたクエリを分析して依存関係および計算順序を求めるように動作する。

【００６０】

依存関係分析

ある実施形態に従うと、多次元データベース（たとえば、Ｅｓｓｂａｓｅ）は、「ｍ」個の次元を備え得て、各次元は、「ｎ」個のメンバを有し、次元の各メンバは、ロードされた入力値または動的メンバとともに格納されることができ、この動的メンバの値は、この動的メンバがクエリにおいて要求されると、実行時における実際の取得中に計算される。次元メンバは、本質的には階層的である。次元の濃度は、次元メンバの総数であり得て、全ての次元メンバ間に構築される各組み合わせは、多次元データベースキューブにおける交点を表す。各キューブは、それに関連付けられた格納されたまたは計算された値を有し得て、組み合わせの各座標は、交点値に対する意味（たとえば、ビジネス上の意味などの抽出された情報）を表す。

【００６１】

ある実施形態に従うと、多次元データベースの動的メンバは、有効な式を有するメンバまたは一時的なメンバ（ＥｓｓｂａｓｅでのＭＤＸ、要求などの、多次元データベースキューブに関連付けられたまたは多次元データベースキューブと通信可能な言語を介して要求当たり作成される）を備え得る。このような式は、このメンバが依存している同一のアウトラインからの他のメンバの計算から生じる値の複雑な条件計算を伴う単純演算式または複雑な式であり得る。したがって、このような動的メンバの式は、その値を表し、各交

点について評価される。

【００６２】

ある実施形態に従うと、動的メンバは、同一のアウトラインからの他のメンバ（式の依存要素と呼ばれる）に依存していてもよく、当然のことながらこれらの依存要素は、各交点の元の式を評価するために最初に計算されなければならない。したがって、ハイブリッドデータ集約モデル（たとえば、「ハイブリッドＥｓｓｂａｓｅ」（ボトムアップ））において動的メンバの式を評価するために、システムはまず、この式が依存している全てのメンバのリストを準備し得る。さらに、動的メンバの依存要素のリストは、「実行時依存要素」および「静的な依存要素」として分類されてもよい。動的メンバの実行時依存要素は、各交点について異なっている依存要素であるのに対して、静的な依存要素は、交点に関係なく一定である依存要素である。動的メンバの実行時依存要素および静的な依存要素を識別するこの分析プロセスは、ハイブリッドデータ集約モデルにおける「依存関係分析」と称される。

【００６３】

ある実施形態に従うと、ブロックストレージデータベースのためのハイブリッド集約は、集約ストレージデータベースの効率と同様の効率でブロックストレージデータ計算が実行される集約モデルである。ハイブリッド集約は、疎な集約を除去し、サイズおよびメモリフットプリントを小さくし、バッチルーチンを高速化することによって、高速パフォーマンスの利点を提供する。実装の際の考慮事項は簡素化される。なぜなら、ユーザは、レベル０計算の頻繁な使用にブロックストレージを使用するか、多くの上位レベルの集約に集約ストレージを使用するか、計算パフォーマンスを促進するためにキューブが次元線に沿って分割されるパーティション化されたモデルを設計するかをもはや考えなくてもよいからである。ブロックストレージデータベースでは、大きな疎な次元を格納しなければならず、それらを動的にすることにより、クエリ時にブロックＩ／Ｏが過剰になり、パフォーマンスに影響を及ぼす。非常に大きな疎な次元を格納することにより、バッチ集約時間が長くなり、これらの疎な次元の数およびサイズに関連して大きくなるデータベースサイズが大きくなることになり得る。このような欠点があるにもかかわらず、ブロックストレージは、その強力な機能のために広く使用されている。集約ストレージは、より多くの次元およびより大きな次元を有する大きなデータベースを可能にするように設計されている。ブロックストレージとは異なって、それは、優れたクエリパフォーマンスを実現するために大きな疎な次元を事前に集約しなくてもよい。大きな次元にわたって高速の動的集約を容易にする集約ストレージエンジンがカギになる。ハイブリッドは、可能であればＡＳＯ計算エンジンを利用し、必要に応じてＢＳＯ計算エンジンに切り換える。

【００６４】

ある実施形態に従うと、依存関係分析は、多次元データベース環境においてハイブリッドフローの一部を形成することができ、このハイブリッドフローは、ユーザが実際の評価を開始する前に全ての必要な依存要素を識別することによってボトムアップアプローチで動的メンバを評価することを可能にする。本明細書に記載されている依存関係分析は、冗長で旧式の再帰的なトップダウントリップを無くすことによってパフォーマンスを向上させる。

【００６５】

ある実施形態に従うと、依存関係分析は、各々の動的な（または、一時的な）メンバについて行われて、ＢＳＯ（ブロックストレージオプション）キューブおよびＡＳＯ（集約ストレージオプション）キューブの両方のキューブにおける実行時依存要素および静的な依存要素のリストを収集する。これにより、メンバは、ボトムアップアプローチでの実行のためのハイブリッドフローに加わることができる。

【００６６】

以下の例において、ある実施形態に係る、例示的な動的メンバを示す、以下に示されるサンプル式を有する動的メンバについて検討する。行と行との間のイタリック体の文章は、上記の行における関数を説明している。

【００６７】
【数１】

1) IF (@ISLEV (“Market”, 0) and (@ISLEV (“SITE”, 1)
ユーザクエリにおける現在のメンバが「市場」次元のレベル０にあり、ユーザクエリにおける現在のメンバが「場所」次元のレベル１にある場合
2) IF (@ISMBR (“New York”))
そして、現在のメンバが「ニューヨーク」である場合
3) @PARENTVAL (“Market”, “Sales”);
「市場」および「売上高」次元における親値を返す
4) ELSE
5) 6;
さもなければ、「６」を返す
6) END IF;
7) ELSE
8) @PARENTVAL (“Product”, “Sales”);
上記の条件が満たされない場合、「市場」および「売上高」次元における親値を返す
9) ENDIF;

【００６８】
ある実施形態に従うと、上記の動的メンバに続いて、依存関係分析のために２つの主要なステップがとられる。第１のステップでは、サーバの起動中に、動的メンバに基づいて依存関係分析が実行される。第２のステップでは、第１のステップの結果を使用して実際の依存要素がフェッチされる。
【００６９】
<u>第１のステップ－サーバの起動中に依存関係分析を実行する</u>
ある実施形態に従うと、第１のステップは、動的な式の中に存在している実行時依存要素次元および依存関係パターンのリストを検出するというものである。このロジックは、サーバの起動中に実行されるため、実際の取得中のＭＤＸクエリターンアラウンドタイムには寄与しない。
【００７０】
ある実施形態に従うと、キューブのロード（たとえば、サーバの起動）中に、本明細書におけるシステムおよび方法は、キューブのアウトライン全体、すなわちメンバを一つ一つ読み込んで、各メンバについて式を調べて各メンバについて依存要素を収集することができる。この依存関係情報は全て、サーバの起動中に収集される。依存関係情報が全て静的であるので、このような収集ステップはユーザクエリに依拠しない。次いで、この依存関係マッピングは、動的メンバに対するクエリが呼び出されると、格納されて呼び出されることができる。
【００７１】
ある実施形態に従うと、依存関係分析の第１のステップの入力は、動的な式を、トークン化された文字列のリストに分解する。

【００７２】
ある実施形態に従うと、このようなステップの出力は、少なくとも２つである。第１の出力は、動的な式が「実行時」依存関係を有する次元のアレイである。上記の動的メンバの例を見ると、実行時依存関係次元は、製品および市場である。これらの次元は、格納された依存関係マッピングに基づいて、実行時におけるこの式メンバの実際の取得中に実行時依存要素メンバの正確なリストを生成する（たとえば、それらの次元は、静的であろうと実行時であろうと、上記の例における依存関係次元である製品および市場のための依存関係マッピングを形成する）。
【００７３】
ある実施形態に従うと、第２の出力は、式の中に存在しているパターンの検出であり、これは、「ＩＦ」（条件ロジック／条件文）条件において式によって使用される「文脈」依存の次元のリストが、式の各「ＩＦ」の本文部分で使用されるものと異なっているか同一であるかを示す。このパターン検出は、式においてＩＦ－ＥＬＳＥネスティングが増加するとより複雑になる。
【００７４】
ある実施形態に従うと、以下の例において、キューブ／データベースという用語は、複数の次元を備える実際のサーバまたはキューブまたはデータベースを意味し得て、各次元のメンバは、階層的であり、アウトライン表示またはツリー表示で表すことができる。同様に、ＭＤＸという用語は、多次元データベースで使用される多次元式（ＭＤＸ）を意味し得る。また、トップダウンまたは実行時依存要素関数という用語は、実際のクエリに依存する関数を意味し得て、その結果として生じる値は、実際の実行中のクエリの文脈の中で評価される。＠ＰＡＲＥＮＴ、＠ＣＵＲＲＭＢＲなどのようないくつかのトップダウン関数があり得る。
【００７５】
ある実施形態に従うと、上記の例では、第１の段階からの高レベル出力は、動的メンバが依存している次元のリストと、ＩＦ　ＥＬＳＥパターンが存在しているか否かの判断と、「ＩＦ」条件において実行時依存要素関数だけが使用する次元のリストとで構成されるであろう。高レベル出力では、上記の例が依存している次元のリストは、市場、製品を含む。この例では、ＩＦ　ＥＬＳＥパターンが存在している。そして、「ＩＦ」条件で使用される次元のリストは、市場、場所を含む。
【００７６】
ある実施形態に従うと、依存関係分析の第１の段階は、スタック（すなわち、関数スタックおよび引数スタック）の状態の方法が各クエリについて作成されるように提供され得る。これらのスタックは、クエリにおける各呼び出しチェーンに基づき得る。
【００７７】
ある実施形態に従うと、上記のように、依存関係分析のための入力は、トークン化されたリストに分解されるクエリの式であり得る。そのため、システムおよび方法は、内部で使用される全ての必要な変数を宣言して、それらをそれぞれのスタックに分類することができる。
【００７８】
ある実施形態に従うと、以下の擬似コードは、依存関係分析においてとられる一次ステップを表す。
【００７９】
【数２】

入力トークンのリストにおける各文字列トークンについて

Do

内部の２つのスタックである関数スタックおよび引数スタックをクリアする

入力リストにおいて第１の関数名の項目の場所を突き止める

第１の関数が見つかれば、それを関数スタックに追加する

While (現在の呼び出しチェーンの終わりではない（括弧などの開始および終了引数の数を一致させることによって検出される）)

{

If (トークンが関数名である)

{

関数のレコードを取得して、ポインタを保存する;

関数名レコードを関数スタックに追加する

}

Else if (トークンが、関数によって使用される引数である)

{

引数を引数スタックに追加する

}

Else if (トークンが引数セパレータである)

Continue

}

If(分析するための呼び出しチェーンを見つけた)

{

個々の呼び出しチェーンを分析するための関数を呼び出す

}

内部の２つのスタックをクリアする

Done

【００８０】

次に、上記の擬似コードは、例の形式で記載することができる。以下の例では、ある実施形態に係る例示的な関数スタックを示す図４を参照して、以下のアウトライン式を使用することができる。

【００８１】

【数３】

```
@WeightedSumX (@Range (“Entered Delta”, USD:ZAR), _FCCS_Rates_, “Rate.Average”,
@CONCATENATE (“Rate_”, @NAME(@CURRMBR(Currency))) + @PRIOR(“Reporting”);
```

【００８２】
ある実施形態に従うと、図に示されるように、上記の式から生じる３つの個々の関数スタックがある。第１の関数スタック４１０は、関数Ａ（@ＷｅｉｇｈｔｅｄＳｕｍＸ）４０１と、関数Ｂ（@ＲＡＮＧＥ）４０２とを備える。これは、第１の関数呼び出しチェーンの終了を表し、この呼び出しチェーンにおける残りの値である「Ｅｎｔｅｒｅｄ　Ｄｅｌｔａ」およびＵＳＤ：ＺＡＲは、引数であり、引数スタックに配置されるであろう。
【００８３】
ある実施形態に従うと、第２の関数スタック４２０は、関数Ａ（@ＷｅｉｇｈｔｅｄＳｕｍＸ）と、関数Ｃ（@ＣＯＮＣＡＴＥＮＡＴＥ）４０３と、関数Ｄ（@ＮＡＭＥ）４０４と、関数Ｅ（@ＣＵＲＲＭＢＲ）４０５とを備える。これは、第２の関数呼び出しチェーンの終了を表し、この呼び出しチェーンにおける残りの値である「Ｒａｔｅ＿」およびＣｕｒｒｅｎｃｙは、引数であり、引数スタックに配置されるであろう。
【００８４】
ある実施形態に従うと、第３の関数スタック４３０は、関数Ｆ（@ＰＲＩＯＲ）４０６を備える。これは、第３の関数呼び出しチェーンの終了を表し、この呼び出しチェーンにおける残りの値である「Ｒｅｐｏｒｔｉｎｇ」は、引数であり、引数スタックに配置されるであろう。
【００８５】
ある実施形態に従うと、上記の例における各呼び出しチェーンを分析することができ、実行時次元を指摘することができる。上記の例では、実行時次元は、ＣｕｒｒｅｎｃｙおよびＲｅｐｏｒｔｉｎｇである。
【００８６】
ある実施形態に従うと、上記の例は、いかなるＩＦ－ＥＬＳＥパターンも含んでいない。
【００８７】
第２の段階：第１の段階の出力を使用して実際の依存要素をフェッチする
ある実施形態に従うと、第１の段階が実行されると（なお、第１の段階は、それが初めてロードされた場合、サーバの起動中に、キューブの耐用期間において一度実行されることができる）、第２の段階が開始し得る。実際の実行時依存要素をフェッチするために使用される下記の第２の段階は、特有であり、取得要求（たとえば、ＭＤＸクエリ）に依存する。したがって、依存要素の実際のフェッチのロジックは、各取得要求について（すなわち、実際のクエリ実行時中に）実行される。依存関係分析の第１の段階において収集された情報は、クエリ実行時依存要素の発見および収集に使用される。
【００８８】
ある実施形態に従うと、第２の段階において、システムおよび方法は、クエリオドメータおよび（第１の段階から得られた）実行時依存要素のリストをその入力とみなす。
【００８９】
ある実施形態に従うと、次に、システムおよび方法は、第１の段階において検出された実行時依存要素次元のみのオドメータからメンバの組み合わせ（または、交点）を構築する。
【００９０】
ある実施形態に従うと、システムおよび方法は、各交点について、動的メンバの内部プリコンパイルプログラムを実行して、各交点に特有であってかつ異なっている依存要素メンバをフェッチする。次いで、システムおよび方法は、各交点について得られた全ての依存要素を出力として蓄積して、それをクエリの拡張オドメータに入れることができる。このステップは、スタックマシンを介して、メタデータについてのみプログラムを評価する。多次元データベースは、値取得のための式プログラムの実行を既に知っている。しかし、システムおよび方法は、値取得をスキップすることによって、同一の式を実行して依存要素メンバのみをフェッチすることを可能にする。
【００９１】
ある実施形態に従って、以下の動的メンバについて検討する。この例では、動的メンバ

は、「Ｔｅｓｔ」と命名されるであろう。
【００９２】
【数４】

```
1)  IF (@ISLEV (“Market”, 0) and (@ISLEV (“SITE”, 1)
2)      IF (@ISMBR (“New York”))
3)              @PARENTVAL (“Market”, “Sales”);
4)      ELSE
5)              6;
6)      END IF;
7)  ELSE
8)      @PARENTVAL (“Product”, “Sales”);
9)  ENDIF;
```

【００９３】
図５は、ある実施形態に係る、例示的なデータセットを示す図である。
図５において、市場５００、場所５１０、製品５２０および売上高５３０などのいくつかの次元が示されている。
【００９４】
ある実施形態および以下の例の目的に従って、ユーザがＭＤＸクエリを取得するための以下の要求を送信したとする。
【００９５】
【数５】

```
SELECT {[Cola], [Old Fashioned], [Dark Cream], [Grape]} ON ROWS, {[East].Children,
[East]} ON COLUMNS FROM [TPDNTest.TPDNTest] WHERE ([Jan], [Test],
[Scenario]);
```

【００９６】
ある実施形態に従うと、［ＴＰＤＮＴｅｓｔ．ＴＰＤＮＴｅｓｔ］は、テスト目的で作成された内部キューブを備え得る。動的メンバである「Ｔｅｓｔ」を実行することにより、図６に示されるデータセットが得られる。
【００９７】
ある実施形態に従うと、動的メンバである「Ｔｅｓｔ」について検討すると、第１の段階（依存関係分析）は、市場および製品の次元を実行時依存関係次元として返した。「Ｔｅｓｔ」は、ＩＦ－ＥＬＳＥパターンも有している。この情報は、入力オドメータとともに、第２の段階によって、実際の実行時依存要素をフェッチするための入力とみなされる。したがって、第２の段階の出力としての新たに求められる依存要素は、メジャー次元における売上高である。また、製品次元には４つのメンバ、すなわちコーラ、ルートビア、クリームソーダおよびフルーツソーダがある。
【００９８】
ある実施形態に従うと、出力は、以下を備え得る：第１の出力は、この式が「実行時」依存関係を有している次元のアレイ、すなわち上記の例では製品、市場の２つの次元である。これらの次元は、実行時におけるこの式メンバの実際の取得中に実行時依存要素メンバの正確なリストを生成することができる。
【００９９】

ある実施形態に従うと、次の出力は、式の中に存在しているパターンの検出であり、これは、「ＩＦ」条件において式によって使用される依存要素次元のリストが、式の各「ＩＦ」の本文部分で使用されるものと異なっているかまたはそこから生じたものではないことを示す。このパターン検出は、式においてＩＦ－ＥＬＳＥネスティングが増加するとより複雑になり、実行時に各交点について依存要素メンバを正確にフェッチするための依存関係分析の極めて重要な部分である。

【０１００】

ある実施形態に従うと、第２の段階のステップの概要は、以下の通りである。

ある実施形態に従うと、第２の段階は、入力オドメータからの実行時依存要素次元（市場および製品）の全てのメンバの中で全ての数学的組み合わせを形成して、各組み合わせについてプログラムを実行して依存要素をフェッチすることができる。

【０１０１】

ある実施形態に従うと、市場および製品からの入力オドメータメンバは、以下の通りである。

【０１０２】

【数６】

製品: {[100-10], [200-10], [300-10], [400-10]} OR {[Cola], [Old Fashioned], [Dark Cream], [Grape]} ここで、第２の組は、対応する製品名の別名である

市場: {[New York], [Massachusetts], [Florida], [Connecticut], [New Hampshire], [East]}

【０１０３】

ある実施形態に従うと、全組み合わせは、以下に示される通りである。

【０１０４】

【数７】

{([100-10], [New York]), ([100-10], [Massachusetts]), ([100-10], [Florida]),
([100-10], [Connecticut]), ([100-10], [New Hampshire]), ([100-10], [East]),

([200-10], [New York]), ([200-10], [Massachusetts]), ([200-10], [Florida]),
([200-10], [Connecticut]), ([200-10], [New Hampshire]), ([200-10], [East]),

([300-10], [New York]), ([300-10], [Massachusetts]), ([300-10], [Florida]),
([300-10], [Connecticut]), ([300-10], [New Hampshire]), ([300-10], [East]),

([400-10], [New York]), ([400-10], [Massachusetts]), ([400-10], [Florida]),
([400-10], [Connecticut]), ([400-10], [New Hampshire]), ([400-10], [East])}

【０１０５】

ある実施形態に従うと、次いで、第２の段階は、この全ての組み合わせのリストに対して反復処理を行うことができ、各組み合わせについて、メンバ「Ｔｅｓｔ」の式を、依存要素メンバをフェッチするためだけに評価することができる。第２の段階は、これらのメンバを依存要素としてメジャー、市場および製品の拡張オドメータにそれぞれ追加することができる。

【０１０６】

【数８】

{[Sales]}, {[East]}, {[100], [200], [300], [400]}

【０１０７】
ある実施形態に従うと、依存要素は、実行された式によって、入力オドメータについて一度メタデータのために収集される。また、このロジックは、組み合わせを構築して、より多くの依存要素を発見するために同一のプログラムを実行するという同一のプロセスを繰り返す。今回、それは、実行時依存要素次元の新たに追加されたメンバのみの中で組み合わせを構築する。
【０１０８】
ある実施形態に従うと、さらに、特定の組み合わせについて式を評価するために、式のプリコンパイルされた実行可能なプログラムは、入力された組み合わせとともに、スタックマシンフレームワークに送り込まれることができ、このスタックマシンフレームワークは、プログラムの評価の仕方を知っている。このスタックマシンは、既存の作業であり、その値について式を評価するが、依存要素メタデータをフェッチするためだけに式を評価することはないということを知っている。したがって、このスタックマシンは、ある特徴で強化され、この特徴は、スタックマシンが式のプログラムを組み合わせとともにその入力とみなして、式の値計算部分を呼び出すことなく、メタデータを依存要素としてフェッチするためだけにこのプログラムを実行することを可能にする。依存要素をフェッチするためだけにプログラムが実行されるスタックマシンのこのモードは、「メタデータ」モードと呼ばれる。
【０１０９】
ある実施形態に従うと、プロセスの第２の段階は、発見された依存要素メンバを保持することができる一時コンテナ内で実行可能である。この構造は、メンバの動的に拡張可能なリストを形成するために使用することができ、多次元データベース内で作成されたいかなるオブジェクトも保持することができる。
【０１１０】
図７は、ある実施形態に係る、依存関係分析のための例示的な方法のフローチャートである。
【０１１１】
ある実施形態に従うと、ステップ７０１において、上記方法は、入力パラメータを受信することができる。これは、動的メンバと、ユーザの取得要求を含むオドメータとを含む。
【０１１２】
ある実施形態に従うと、ステップ７０２において、上記方法は、動的メンバが静的であって、実行時依存関係を有していないか否かを確認する。そうであれば、ステップ７０３において、上記方法は、この式プログラムをデフォルトの組み合わせとともにスタックマシンに渡す。上記方法は、デフォルトの組み合わせについて一度だけこのプログラムを実行して、全ての静的な依存要素をフェッチする。次いで、上記方法は、全てのこれらの依存要素を一時コンテナからオドメータに転送する。
【０１１３】
ある実施形態に従うと、ステップ７０４において、上記方法は、動的メンバが実行時依存関係を有しているか否かおよび動的メンバの式の中にＩＦ－ＥＬＳＥパターンが存在しているか否かを確認する。そうであれば、ステップ７０５において、上記方法は、実行時依存要素次元のリストを取得する。上記方法は、実行時依存要素次元の入力オドメータに対して反復処理を行う。実行時依存要素次元のオドメータからのメンバの各組み合わせ（総数＝各オドメータのサイズの乗算）について、上記方法は、現在の式プログラムおよび現在の組み合わせをプログラム実行のためにスタックマシンに渡し、メタデータについてのみプログラムを実行しながら、上記方法は、全ての値計算部分をプログラムから除外し

てメタデータ命令のみを実行し、上記方法は、現在のｃｍｉについての今回のプログラムの実行から得られた依存要素を一時コンテナに格納する。

【０１１４】

ある実施形態に従うと、この時点で、プログラムは、全ての考えられる組み合わせについて実行されており、得られた全ての依存要素は、一時コンテナの中に存在している。ここで、この一時コンテナに対して反復処理を行う。

【０１１５】

ある実施形態に従うと、ステップ７０６において、上記方法は、一時コンテナの中の各メンバについて、依存要素メンバをクエリの拡張オドメータに追加する。

【０１１６】

ある実施形態に従うと、ステップ７０７において、上記方法は、全ての組み合わせを必要とするわけではない。その代わりに、上記方法は、より少ない数の組み合わせを選択し、各縦軸は、全ての組み合わせの中に一度だけ登場する。この場合、組み合わせの総数は、実行時依存要素の各オドメータのサイズの最大値に等しい。上記方法は、実行時依存要素次元のオドメータからのメンバの各組み合わせ（総数＝各オドメータのサイズの最大値）について、現在の式プログラムおよび現在の組み合わせをプログラム実行のためにスタックマシンに渡し、メタデータについてのみプログラムを実行しながら、上記方法は、全ての値計算部分をプログラムから除外してメタデータ命令のみを実行し、次いで、上記方法は、現在のｃｍｉについての今回のプログラムの実行から得られた依存要素を一時コンテナに格納する。

【０１１７】

ある実施形態に従うと、この時点で、プログラムは、全ての考えられる組み合わせについて実行されており、得られた全ての依存要素は、一時コンテナの中に存在している。ここで、上記方法は、この一時コンテナに対して反復処理を行う。

【０１１８】

ある実施形態に従うと、一時コンテナの中の各メンバについて、上記方法は、依存要素メンバをクエリの拡張オドメータに追加する。

【０１１９】

ある実施形態に従うと、全ての動的メンバは、拡張され、すなわち各々の動的メンバの必要な依存要素は全て認識されて、クエリの拡張オドメータに追加される。次いで、この拡張オドメータは、多次元データベース環境におけるハイブリッドフローによって先に進められて、ボトムアップアプローチで完全なオドメータを取得して、然るべき結果をユーザに返す。

【０１２０】

図８は、多次元データベースにおける依存関係分析のための例示的な方法を示す図である。

【０１２１】

ステップ８１０において、上記方法は、１つまたは複数のマイクロプロセッサを含むコンピュータに、コンピュータ上で動作する多次元データベースサーバを設けることができ、多次元データベースサーバは、少なくとも１つの多次元キューブをサポートし、多次元データベースは、複数の次元を備え、複数の次元の各々は、複数のメンバと動的メンバとを備え、動的メンバは、一組の複数のメンバに依存している。

【０１２２】

ステップ８２０において、上記方法は、依存関係分析を実行して、動的メンバが依存している一組の複数のメンバを判断することができる。

【０１２３】

本発明のさまざまな実施形態について上記したが、それらは限定ではなく例として提示されているということが理解されるべきである。これらの実施形態は、本発明の原理およびその実際的用途を説明するために選択されて説明された。これらの実施形態は、本発明が利用されるシステムおよび方法を示しており、新たなおよび／または改良された特徴を

提供することによって、および／または、リソース利用の減少、容量の増加、効率の向上およびレイテンシの減少などのメリットを提供することによってシステムおよび方法のパフォーマンスを向上させる。

【０１２４】

いくつかの実施形態では、本発明の特徴は、全体または一部が、プロセッサと、メモリなどの記憶媒体と、他のコンピュータと通信するためのネットワークカードとを含むコンピュータにおいて実現される。いくつかの実施形態では、本発明の特徴は、コンピュータの１つまたは複数のクラスタが、ローカルエリアネットワーク（ＬＡＮ）、スイッチファブリックネットワーク（たとえば、インフィニバンド）またはワイドエリアネットワーク（ＷＡＮ）などのネットワークによって接続される分散コンピューティング環境において実現される。分散コンピューティング環境は、単一の位置に全てのコンピュータを有していてもよく、またはＷＡＮによって接続されたさまざまな遠隔の地理的位置にコンピュータのクラスタを有していてもよい。

【０１２５】

いくつかの実施形態では、本発明の特徴は、全体または一部が、ウェブ技術を使用してセルフサービスの計量の態様でユーザに届けられる共有の融通性のあるリソースに基づいて、クラウドコンピューティングシステムの一部として、またはクラウドコンピューティングシステムのサービスとして、クラウドにおいて実現される。（アメリカ国立標準技術研究所によって定義されるように）クラウドの特徴は５つある。すなわち、オンデマンド・セルフサービス、幅広いネットワークアクセス、リソースの共用、スピーディな拡張性、およびサービスが計測可能であること、である。クラウド実装モデルは、パブリック、プライベートおよびハイブリッドを含む。クラウドサービスモデルは、ソフトウェア・アズ・ア・サービス（ＳａａＳ）、プラットフォーム・アズ・ア・サービス（ＰａａＳ）、データベース・アズ・ア・サービス（ＤＢａａＳ）、およびインフラストラクチャ・アズ・ア・サービス（ＩａａＳ）を含む。本明細書におけるクラウドは、ハードウェアと、ソフトウェアと、ネットワークと、共有の融通性のあるリソースをセルフサービスの計量の態様でユーザに届けるウェブ技術との組み合わせである。本明細書におけるクラウドは、別段の定めがない限り、パブリッククラウドの実施形態、プライベートクラウドの実施形態、ハイブリッドクラウドの実施形態、ならびにクラウドＳａａＳ、クラウドＤＢａａＳ、クラウドＰａａＳおよびクラウドＩａａＳを含むがこれらに限定されない全てのクラウド実装モデルを包含する。

【０１２６】

いくつかの実施形態では、本発明の特徴は、ハードウェア、ソフトウェア、ファームウェアもしくはそれらの組み合わせを使用して、またはハードウェア、ソフトウェア、ファームウェアもしくはそれらの組み合わせの助けを借りて、実現される。いくつかの実施形態では、本発明の特徴は、本発明の１つまたは複数の機能を実行するように構成またはプログラムされたプロセッサを使用して実現される。いくつかの実施形態では、プロセッサは、本明細書に記載されている機能を実行するように設計されたシングルチップもしくはマルチチッププロセッサ、デジタル信号プロセッサ（ＤＳＰ）、システムオンチップ（ＳＯＣ）、特定用途向け集積回路（ＡＳＩＣ）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）もしくは他のプログラマブルロジックデバイス、ステートマシン、ディスクリートのゲートもしくはトランジスタロジック、ディスクリートのハードウェアコンポーネント、またはそれらの任意の組み合わせである。いくつかの実現例では、本発明の特徴は、所与の機能に特有の回路によって実現されてもよい。他の実現例では、これらの特徴は、たとえばコンピュータ読取可能記憶媒体上に格納された命令を使用して特定の機能を実行するように構成されたプロセッサにおいて実現されてもよい。

【０１２７】

いくつかの実施形態では、本発明の特徴は、処理システムおよび／またはネットワーキングシステムのハードウェアを制御するため、ならびに、プロセッサおよび／またはネットワークが本発明の特徴を利用する他のシステムと対話することを可能にするためのソフ

トウェアおよび／またはファームウェアに組み込まれる。このようなソフトウェアまたはファームウェアは、アプリケーションコード、デバイスドライバ、オペレーティングシステム、仮想マシン、ハイパーバイザ、アプリケーションプログラミングインターフェイス、プログラミング言語および実行環境／コンテナを含み得るが、これらに限定されるものではない。適切なソフトウェアコーディングは、ソフトウェア分野における当業者に明らかであるように、本開示の教示に基づいて熟練のプログラマによって容易に準備することができる。

【０１２８】

いくつかの実施形態では、本発明は、命令が格納された記憶媒体またはコンピュータ読取可能媒体であるコンピュータプログラム製品を含み、これらの命令を使用して、本発明のプロセスまたは機能のうちのいずれかを実行するようにコンピュータなどのシステムをプログラムまたはそうでなければ構成することができる。記憶媒体またはコンピュータ読取可能媒体は、フロッピー（登録商標）ディスク、光ディスク、ＤＶＤ、ＣＤ－ＲＯＭ、マイクロドライブおよび光磁気ディスクを含む任意のタイプのディスク、ＲＯＭ、ＲＡＭ、ＥＰＲＯＭ、ＥＥＰＲＯＭ、ＤＲＡＭ、ＶＲＡＭ、フラッシュメモリデバイス、磁気もしくは光カード、ナノシステム（分子メモリＩＣを含む）、または、命令および／もしくはデータを格納するのに適した任意のタイプの媒体もしくはデバイスを含み得るが、これらに限定されるものではない。特定の実施形態では、記憶媒体またはコンピュータ読取可能媒体は、非一時的な記憶媒体または非一時的なコンピュータ読取可能媒体である。

【０１２９】

上記の説明は、網羅的であるよう意図されたものではなく、本発明を開示されている厳密な形態に限定するよう意図されたものでもない。また、特定の一連のトランザクションおよびステップを使用して本発明の実施形態を説明してきたが、本発明の範囲が記載されている一連のトランザクションおよびステップに限定されないということは当業者に明らかであるべきである。さらに、ハードウェアおよびソフトウェアの特定の組み合わせを使用して本発明の実施形態を説明してきたが、ハードウェアおよびソフトウェアの他の組み合わせも本発明の範囲内であるということが認識されるべきである。さらに、さまざまな実施形態は、本発明の特徴の特定の組み合わせを説明しているが、特徴の異なる組み合わせが本発明の範囲内であって１つの実施形態の特徴を別の実施形態に組み込むことができるものとして当業者に明らかであるということが理解されるべきである。さらに、形状、詳細、実現および用途の点でのさまざまな追加、削減、削除、変化、ならびに他の変形および変更が本発明の精神および範囲から逸脱することなく本明細書においてなされてもよいということが当業者に明らかであろう。本発明のより広い精神および範囲は、以下の特許請求の範囲およびそれらの等価物によって定義されるよう意図される。

【図面】

【図１】

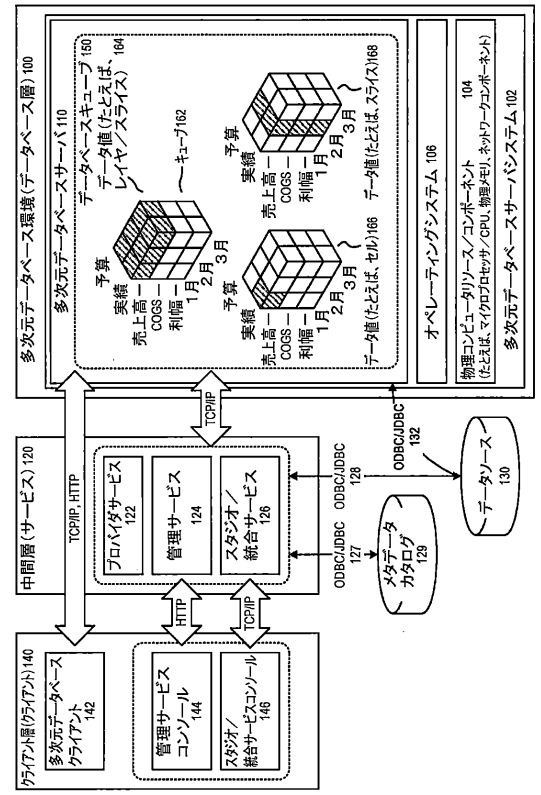

FIGURE 1

【図２】

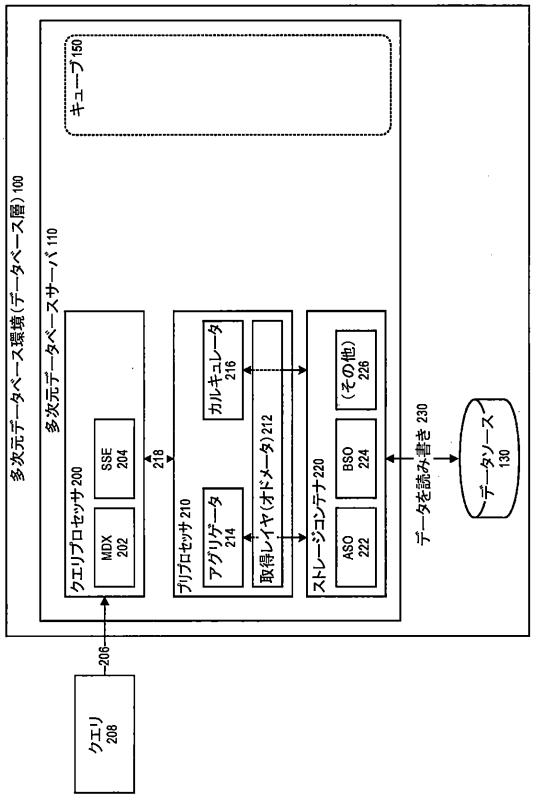

FIGURE 2

【図３】

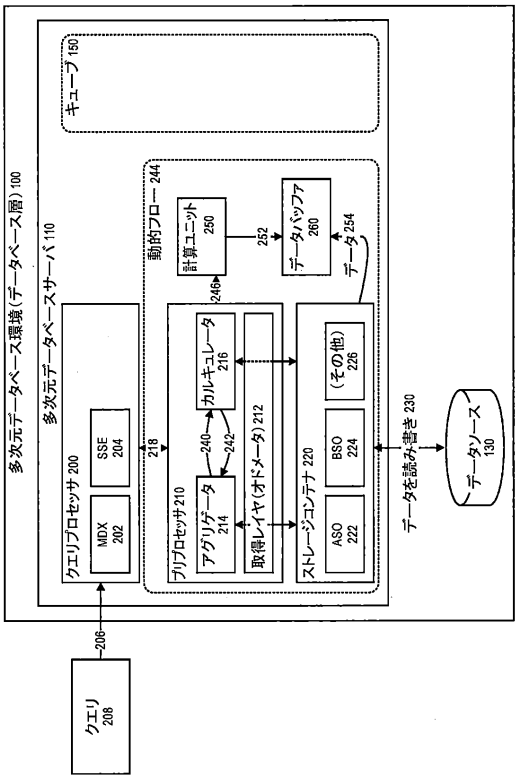

FIGURE 3

【図４】

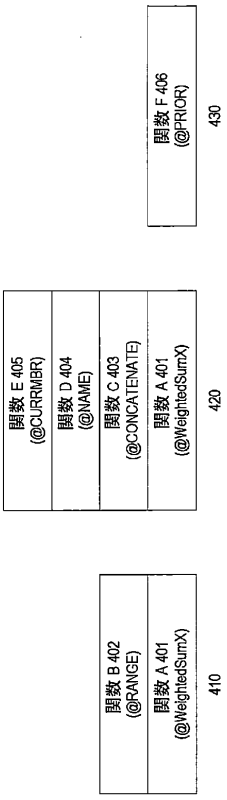

FIGURE 4

【図５】

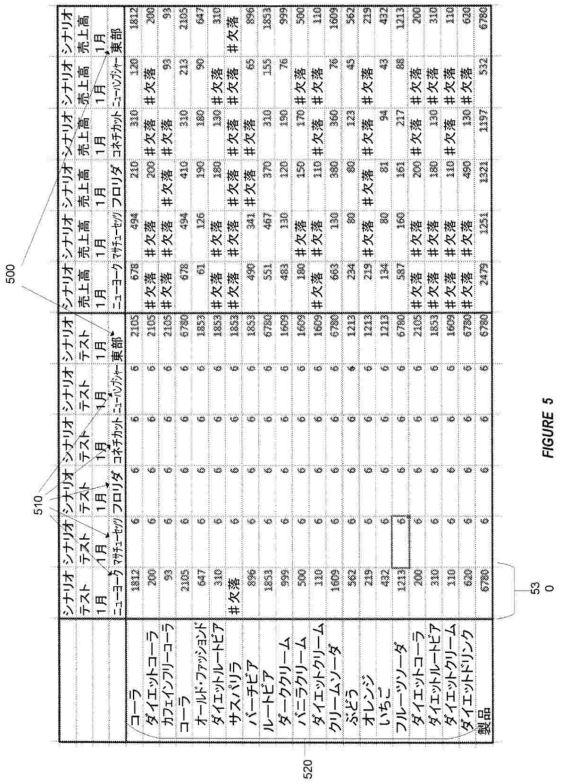

FIGURE 5

【図６】

	1月	テスト	シナリオ			
	ニューヨーク	マサチューセッツ	フロリダ	コネチカット	ニューハンプシャー	東部
コーラ	1812	6	6	6	6	2105
オールド・ファッションド	647	6	6	6	6	1853
ダーククリーム	999	6	6	6	6	1609
ぶどう	562	6	6	6	6	1213

FIGURE 6

【図７】

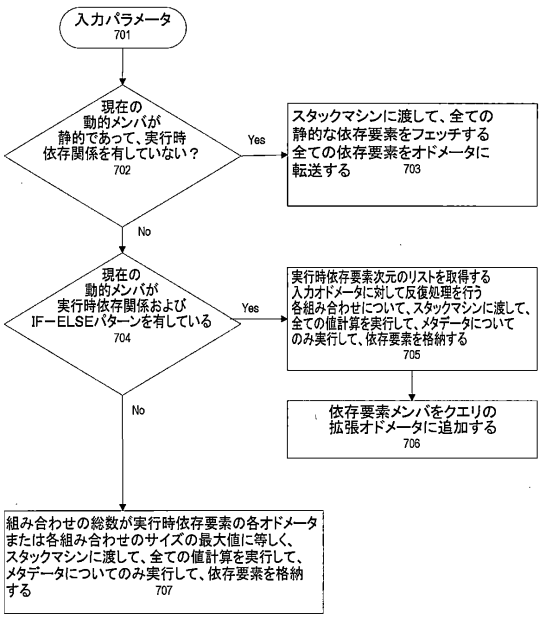

FIGURE 7

【図８】

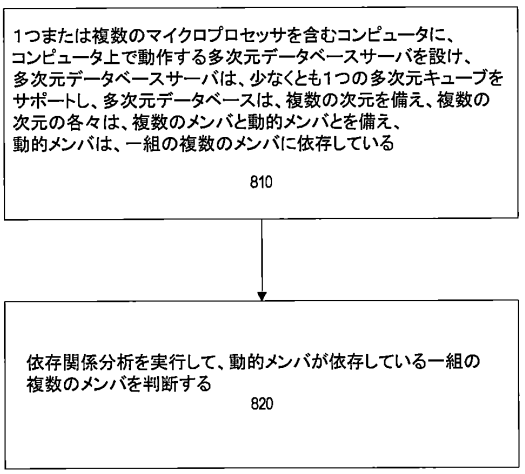

FIGURE 8

フロントページの続き

(33)優先権主張国・地域又は機関
米国(US)
アメリカ合衆国、９４０６５ カリフォルニア州、レッドウッド・シティー、オラクル・パークウェイ、５００、オラクル・インターナショナル・コーポレイション
(72)発明者 リアボイ，セルゲイ
アメリカ合衆国、９４０６５ カリフォルニア州、レッドウッド・シティー、オラクル・パークウェイ、５００、オラクル・インターナショナル・コーポレイション
審査官 齊藤 貴孝
(56)参考文献 米国特許出願公開第２０１７／０１１６３０９（ＵＳ，Ａ１）
特開平０９－２６５４７９（ＪＰ，Ａ）
特表２０１７－５２５０７２（ＪＰ，Ａ）
米国特許出願公開第２０１２／０１８５４２５（ＵＳ，Ａ１）
(58)調査した分野 (Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １６／００－１６／９５８