(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2005-18777
(P2005-18777A)
(43)公開日　平成17年1月20日(2005.1.20)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 12/00	G06F 12/00	513D	5B081
G06F 9/45	G06F 12/00	547Z	5B082
	G06F 9/44	320F	

審査請求 未請求　請求項の数 15　ＯＬ　（全 32 頁）

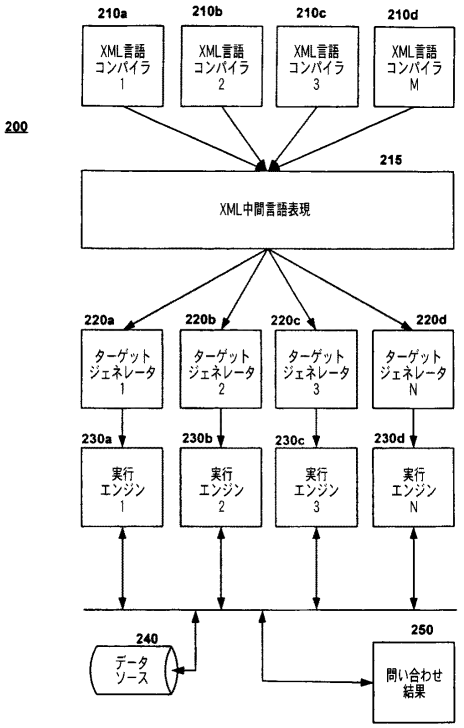
(21)出願番号　特願2004-185647(P2004-185647)
(22)出願日　平成16年6月23日(2004.6.23)
(31)優先権主張番号　10/601,445
(32)優先日　平成15年6月23日(2003.6.23)
(33)優先権主張国　米国(US)

(71)出願人　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　アルパン　エー．デサイ
アメリカ合衆国　９８０３４　ワシントン州　カークランド　１１３　アベニュー　１４４２７

最終頁に続く

(54)【発明の名称】共通問い合わせ実行時システムおよびアプリケーションプログラミングインターフェイス

(57)【要約】
【課題】　共通問い合わせ実行時システムおよびアプリケーションプログラミングインターフェイスを提供すること。
【解決手段】　問い合わせ実行時アーキテクチャおよびそのアーキテクチャに適した例示的アプリケーションプログラミングインターフェイスが提示される。このアーキテクチャは１つまたは複数のＸＭＬ問い合わせおよびビューを入力してそれらの問い合わせを変換させることができ、それらの問い合わせおよびビューを異なるデータモデルの複数のデータソースに対して実行することができる。このアーキテクチャは、入力問い合わせおよびビューをそれぞれの問い合わせまたはビューの意味を表す中間言語表現に変換する、フロントエンドコンパイラを組み込んだものである。このアーキテクチャは、その中間言語表現の、所望の問い合わせ先のデータソースと互換性のあるターゲット言語へのバックエンドコンパイルを可能にする。
【選択図】　図２



【特許請求の範囲】
【請求項１】
実行可能な問い合わせの構築のためのシステムにおいて、アプリケーションとやりとりする方法であって、
前記システムが前記アプリケーションから、１つまたは複数の入力問い合わせをＸＭＬ中間言語表現に変換するために１つまたは複数のコンパイルパラメータおよびコマンドを設定するよう求める１つまたは複数の呼び出しを受け取るステップと、
前記システムが前記アプリケーションから、前記ＸＭＬ中間言語表現を実行可能な問い合わせに変換するよう求める１つまたは複数の呼び出しを受け取るステップと
を備えることを特徴とする方法。
【請求項２】
前記アプリケーションが前記システムから、イベント状況、進捗状況、中間結果、最終結果、エラーメッセージ、警告およびヘルプメッセージからなるグループのうちの１つまたは複数を受け取るステップをさらに備えることを特徴とする請求項１に記載の方法。
【請求項３】
１つまたは複数の環境、コンパイルパラメータおよびコンパイルコマンドを設定するよう求める前記１つまたは複数の呼び出しは、前記システムからのメッセージオブジェクト受け取りを可能にするステップ、問い合わせ許可および実行制限を指定するステップ、前記入力問い合わせおよびコンパイラ型を選択するステップ、および評価コンテキストを確立するステップのうち１つまたは複数を含むことを特徴とする請求項１に記載の方法。
【請求項４】
前記コンパイラ型はＸＰａｔｈ、ＸＳＬＴおよびＸＱｕｅｒｙ言語コンパイラを含むことを特徴とする請求項３に記載の方法。
【請求項５】
前記ＸＭＬ中間言語表現は入力問い合わせの意味論的表現であることを特徴とする請求項１に記載の方法。
【請求項６】
前記ＸＭＬ中間言語を前記実行可能な問い合わせに変換するステップは、ターゲット問い合わせ実行エンジンでの直接実行のために前記ＸＭＬ中間言語を準備するステップを含むことを特徴とする請求項１に記載の方法。
【請求項７】
前記ＸＭＬ中間言語を前記実行可能な問い合わせに変換するステップは、ターゲットジェネレータを使用して前記ＸＭＬ中間言語をターゲット表現に変換するステップを含むことを特徴とする請求項１に記載の方法。
【請求項８】
前記ターゲット表現は、ＸＭＬ言語ターゲット、ＳＱＬ言語ターゲットおよび中間言語ターゲットからなるグループのうち１つまたは複数であることを特徴とする請求項７に記載の方法。
【請求項９】
請求項１に記載のアプリケーションとやりとりする方法を利用することを特徴とする実行可能な問い合わせ構築のためのシステム。
【請求項１０】
問い合わせ結果を生成する入力問い合わせのコンパイルおよび実行のためのシステムであって、
入力問い合わせを受け取る入力装置と、
前記入力問い合わせからＸＭＬ中間言語表現がコンパイルされる１つまたは複数の中間言語コンパイラと、
前記ＸＭＬ中間言語表現が、ターゲット問い合わせを形成する１つまたは複数のターゲット表現に変換される１つまたは複数のターゲットジェネレータと、
問い合わせ先である１つまたは複数のデータソースと、

前記問い合わせ結果を生成するために前記ターゲット問い合わせが前記１つまたは複数のデータソースに対して実行される実行エンジンと
を備えることを特徴とするシステム。
【請求項１１】
前記入力問い合わせは、ＸＰａｔｈ、ＸＳＬＴ、およびＸＱｕｅｒｙのうちの１つまたは複数から形成された問い合わせを含むことを特徴とする請求項１０に記載のシステム。
【請求項１２】
前記ＸＭＬ中間言語表現は前記入力問い合わせの意味を表すことを特徴とする請求項１０に記載のシステム。
【請求項１３】
前記１つまたは複数のターゲットジェネレータは、ＸＭＬ言語生成プログラム、ＳＱＬ言語生成プログラムおよび中間言語生成プログラムのうちの１つまたは複数を含むことを特徴とする請求項１０に記載のシステム。
【請求項１４】
前記１つまたは複数のデータソースは、１つまたは複数の関係データソースおよび非関係データソースを含むことを特徴とする請求項１０に記載のシステム。
【請求項１５】
非関係データソースはスプレッドシートおよびワード処理文書を含むことを特徴とする請求項１４に記載のシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般に、データソースに問い合わせ（ｑｕｅｒｙ）を行うソフトウェアの分野に関し、より詳細には、ＸＭＬ中間言語を使用して１つまたは複数のデータソースに問い合わせを行うことに関する。
【背景技術】
【０００２】
拡張可能なマークアップ言語（ＸＭＬ、eXtensible Markup Language）は、ワールドワイドウェブコンソーシアム（Ｗ３Ｃ（登録商標））に承認された、人間可読タグを用いてデータにマーク付けを行うための汎用構文を提供する文書フォーマット設定のための標準（非特許文献１参照）である。ＸＭＬは、適切に定義されたフォーマットで文書の内容を容易に記述することはできるが、その構造が標準テキスト文書の構造と一致しないため、あるいは他の何らかの非ＸＭＬ対応特性のために、あまり容易には記述できない他のデータソースもある。そうしたデータソースの一例としてスプレッドシートや関係データベースが考えられる。
【０００３】
多種多様なデータプログラミングモデルを有するデータソースに対してＸＭＬライクなサーチを行うという挑戦は、仮想ＸＭＬと呼ばれる。この用語は、一般に、仮想ＸＭＬビューに問い合わせることを含むと解される。仮想ＸＭＬとは、複数のデータアクセスプログラミングモデルにまたがる整合性を確立し、ユーザが、データの実際の記憶フォーマットではなく、自分の思う通りのやり方でそのデータを処理できるようにする概念である。仮想ＸＭＬデータに問い合わせを行うという概念には、データを実際には決してＸＭＬに変換せずに、そのデータをそれがあたかもＸＭＬであるかのように扱うことが関与する。この概念の一利点は、ＸＭＬ符号化のオーバーヘッドが最小限に保たれることである。この仮想ＸＭＬシナリオに、問い合わせ言語を利用して非ＸＭＬデータソースに、そのデータソースがあたかもＸＭＬ問い合わせであるかのように問い合わせを行うことができるという利点があれば、それは望ましいことである。また、実際のデータと仮想ＸＭＬ表現の間のマッピングが忠実度の高いものであることも望ましいことである。
【０００４】
仮想ＸＭＬの実装にはそれに固有の多数の難題がある。１つの問題は効率性である。単

に、例えば、ＸＭＬリーダなどの仮想ＸＭＬインターフェイスを用いてデータソースを顕在化させ、次いで、例えば、ＸＭＬ文書オブジェクトモデル（ＤＯＭ）など既存の問い合わせ実装を用いてそれに問い合わせすることもできるであろう。しかし、その作業はすべて、データソース自体によって実施されるのではなく、ＸＭＬ問い合わせエンジンで行われる。データソース自体、およびそれに関連するデータ管理システムは、異なるデータモデルを有する外部問い合わせシステムよりも、それ専用に設計された言語での方が、そのデータの問い合わせを実施するにはより効率がよいと想定される。

【０００５】

この態様は、仮想ＸＭＬ問い合わせを実装する際に別の問題をもたらす。それはすなわち、ＸＭＬデータモデルが必ずしも基底となるデータモデルおよびその型のシステムにうまく整合するとは限らないことである。基底となるデータソースの型のすべてをＸＭＬの型にマッピングすることもできるであろうが、このプロセスは忠実度を低下させ、非効率でもある。さらに、あるシステムでの型が別のシステムで自明の等価性をもたないこともある。例えば、ＸＭＬでイメージなどのバイナリデータを表現するには、ＸＭＬ文字セットへの高くつく変換（例えばＢａｓｅ６４符号化など）を必要とする。

【０００６】

仮想ＸＭＬに問い合わせを行う従来の試みでは、仮想ＸＭＬを全く具体化せずに、問い合わせ用とマッピング用の２つの異なるデータ構造を構築し、次いで、それらを相前後してトラバースして元のデータソースに直接行われる効率のよい問い合わせを生成することによって、その問題に取り組もうとした。この手法は、最初はうまく行くが、問い合わせおよびマッピング言語の複雑度が増すに従って、展開が極めて困難になる。問い合わせまたはマッピングでの諸概念は、しばしば、ターゲットデータモデルに直接変換されず、複雑なＸＭＬビューを用いて複雑な問い合わせを合成すると多量の意味分析および書き換えが必要になる。

【０００７】

さらに、多数のデータソースに対する、ある言語での問い合わせの問い合わせ表現または問い合わせ結果への変換をサポートすることのできるシステムアーキテクチャは、普通、Ｍ×Ｎ通りのパスという高くつく実装を必要とする（Ｍは入力オプションの数であり、Ｎは出力オプションの数である）。そうした変更コンパイラは、標準アーキテクチャを使用すると膨大な数になることがある。

【０００８】

【非特許文献１】Ｗ３Ｃ、テクニカル・レポーツ・アンド・パブリケーションズ（Technical Reports and Publications）、[online]、２００４年６月１１日、Ｗ３Ｃ、[平成１６年６月２２日検索]、インターネット<URL:http://www.w3c.org/tr>

【発明の開示】

【発明が解決しようとする課題】

【０００９】

したがって、ＸＭＬおよび非ＸＭＬデータソースに対するＸＭＬ問い合わせおよびビューのための仮想ＸＭＬを実装する統一表現および単一システムアーキテクチャが求められている。本発明は、前述のニーズに対処し、統一表現を利用するアーキテクチャと本発明のシステムのユーザのためのアプリケーションプログラミングインターフェイスの両方を用いてそれらを解決する。

【課題を解決するための手段】

【００１０】

問い合わせ結果を生成する入力問い合わせのコンパイルおよび実行のためのシステムが提案され、そのシステムは、入力を受け取る入力装置、その入力問い合わせの意味論的意味の生成のための中間言語コンパイラ、ＸＭＬ中間言語表現をターゲット問い合わせを形成するターゲット言語に変換するターゲットジェネレータ（ｔａｒｇｅｔ　ｇｅｎｅｒａｔｏｒ）（またはターゲット言語コンパイラ）、１つまたは複数のデータソース、および実行エンジンを含む。実行エンジンは、データソースに対して入力問い合わせを実行させ

る。実行エンジンはＸＭＬ中間言語を直接実行することもでき、あるいはまずＸＭＬ中間言語をターゲット言語に変換してからその問い合わせを実行することもできる。入力問い合わせは、任意のＸＭＬ問い合わせまたはビューからのものとすることができ、ターゲット言語は、データソースに問い合わせを行うことのできる任意の問い合わせ言語とすることができる。データソースは、関係型データとすることも非関係型（例えば階層型などの）データとすることもでき、システムは複数のデータソースに問い合わせをすることができる。

【００１１】

アプリケーションとやりとりし、アプリケーションプログラミング言語の機能を定義する一方法について述べる。アプリケーションは問い合わせシステムに、１つまたは複数の入力問い合わせをＸＭＬ中間言語表現に変換するための１つまたは複数のコンパイルパラメータおよびコマンドを設定するよう求める１つまたは複数の呼び出しを送ることができる。さらに、アプリケーションはシステムに、ＸＭＬ中間言語表現を実行可能な問い合わせに変換するよう求める１つまたは複数の呼び出しを送ることもできる。システムはアプリケーションに、例えば、イベント状況、進捗状況、中間結果、最終結果、エラーメッセージ、警告およびヘルプメッセージなどを送ることもできる。

【００１２】

本発明のその他の特徴および利点は、添付の図面を参照して進められる例示的実施形態の以下の詳細な説明を読めば明らかになるであろう。

【００１３】

前述の要約、ならびに以下の好ましい実施形態の詳細な説明は、添付の図面と併せて読めばよりよく理解されるものである。本発明を説明するために、図面に、本発明の例示的構成を示すが、本発明は、開示の具体的方法および手段に限られるものではない。

【発明を実施するための最良の形態】

【００１４】

概要

本発明は、複数の問い合わせ元から複数の種類のデータソースに対する問い合わせを行うという問題に対処する。仮想ＸＭＬデータソースに問い合わせを行うという問題への１つの解決法は、統一中間言語を使用することであると考えられる。本発明のＸＭＬ中間言語は、問い合わせの意味または意味体系を明示的に表すものである。このＸＭＬ中間言語を問い合わせ中間言語（ＱＩＬ）と呼ぶ。

【００１５】

ＱＩＬは、「問い合わせ／ビュー比較」として知られている問題に対処する。一例として、ＸＭＬ問い合わせが、ＸＭＬ、ＸＭＬ仮想またはその他のデータであるデータの仮想ＸＭＬビューに対して実施されると想定する。１つの手法は、そのデータソースをＸＭＬとして具体化することであると考えられるが、これは非常に非効率的であり、システムが使用可能なものより多くのメモリを必要とすることがある。別の手法は、そのビューを仮想化し、その仮想化ビューを用いて問い合わせを構成し、その結果を元のデータに対する演算に変換するものである。ユーザには、論理ＸＭＬデータモデルに対するＸＭＬ問い合わせに見えるが、その実装は、何であれそれが提供する問い合わせシステムを使用して、その固有のデータフォーマットに問い合わせを行う。この手法は、関係データベースで構造化問い合わせ言語（ＳＱＬ）ビューに対するＳＱＬ問い合わせに使用され、過去においてはＸＭＬビューに対するＸＭＬ問い合わせのいくつかの実装で使用されてきた。しかし、ＱＩＬなど、ＸＭＬ中間言語を使用すると、その問い合わせを実行する前に、元の潜在的に複雑なビューを、より小さい原子的な（ａｔｏｍｉｃ）問い合わせ演算に分解することができる。このようにして、複雑なビューに対する問い合わせは、より単純なビューに問い合わせを加えたものに対する問い合わせになる。問い合わせ合成（ｑｕｅｒｙ　ｃｏｍｐｏｓｉｔｉｏｎ）がこれを単なるより単純なビューに対する問い合わせに変え、それによって問題が単純化される。言い換えると、そのＸＭＬビュー自体が単に１つの問い合わせにすぎなくなる。

【００１６】
ＸＭＬ中間言語ＱＩＬは、（１）ＸＭＬ問い合わせとＸＭＬビュー両方の統一表現を提供し、それによって、問い合わせ／ビュー合成問題を大幅に単純化すると共に、（２）すべてのビューを「仮想ＸＭＬ」として処理することによってシステムのインターフェイスを大幅に単純化する。あらゆる可能な言語およびデータモデルごとに１つずつＡＰＩをもつのではなく、それらのＡＰＩすべてが１つの共通データモデル、ＸＭＬ中間言語ＱＩＬの演算子を共用することができる。
【００１７】
ＸＭＬ中間言語ＱＩＬは、公知のコンパイラ問題にも対処する。普通、１つの言語を用いる場合、コンパイラは、Ｎ通りの後置（バックエンド）ターゲットマシンに対してＭ通りの前置（フロントエンド）言語を実装する必要がある（ＭおよびＮは整数である）。各対ごとの組み合わせが実装される場合には、必要とされる組み合わせをカバーするためにＭ×Ｎ通りのコンパイラ実装が必要になる。しかし、共通中間表現を導入することによってその２つが切り離される場合には、コンパイラの計算量は、Ｍ＋Ｎ通りだけに低減される。
【００１８】
本発明は、ＱＩＬを使用する問い合わせ実行時システムのアーキテクチャと、１つまたは複数のアプリケーションプログラムによるそのシステムの使用を可能にする例示的アプリケーションプログラミングインターフェイスの両方を開示するものである。
【００１９】
計算処理装置の例
図１および以下の説明は、本発明を実装することのできる適当な計算処理環境の簡潔で一般的な説明を提供するためのものである。ただし、ハンドヘルド、携帯用およびその他の計算処理装置およびあらゆる種類の計算処理オブジェクトを、本発明と一緒に使用することが企図されていることを理解すべきである。したがって、以下では、汎用コンピュータについて述べるが、これは一例にすぎず、本発明は、ネットワーク／バス相互運用性および対話を有するクライアントなど、他の計算処理装置を用いて実装することもできる。したがって、本発明は、クライアントリソースがほとんど関与しない、または最小限のクライアントリソースが関与するだけのネットワークで接続され、ホスティングされたサービスの環境、例えば、クライアント装置が、単に、電気製品に配置されたオブジェクトなど、ネットワーク／バスへのインターフェイスとして、あるいは他の計算処理装置およびオブジェクトへのインターフェイスとして働くだけのネットワーク化環境で実装することもできる。本質的に、データを記憶することができ、またはそこからデータを取り出すことのできるどのような場所でも、本発明による動作に望ましい、または適した環境である。
【００２０】
必須ではないが、本発明は、装置またはオブジェクト用サービスの開発者による使用のために、オペレーティングシステムを介して実装することができ、かつ／または本発明に従って動作するアプリケーションソフトウェアに含めることもできる。ソフトウェアは、クライアントワークステーション、サーバその他の装置など、１つまたは複数のコンピュータによって実行される、プログラムモジュールなど、コンピュータ実行可能命令の一般的状況で説明することができる。一般に、プログラムモジュールには、個々のタスクを実行し、または個々の抽象データ型を実装するルーチン、プログラム、オブジェクト、コンポーネント、データ構造などが含まれる。通常は、プログラムモジュールの機能性は、様々な実施形態で、所望に応じて組み合わせ、または分散させることができる。さらに、本発明を、他のコンピュータ構成を用いて実施することもできることを当分野の技術者は理解するであろう。本発明での使用に適すると考えられる他の公知の計算処理システム、環境および／または構成には、それだけに限らないが、パーソナルコンピュータ（ＰＣ）、現金自動預入支払機、サーバコンピュータ、ハンドヘルドまたはラップトップ装置、マルチプロセッサシステム、マイクロプロセッサベースのシステム、プログラム可能家庭用電

化製品、ネットワークＰＣ、電気製品、照明器具、環境制御要素、ミニコンピュータ、メインフレームコンピュータなどが含まれる。本発明は、通信ネットワーク／バスその他のデータ伝送媒体を介してリンクされたリモート処理装置によってタスクが実行される分散計算処理環境で実施することもできる。分散計算処理環境では、プログラムモジュールは、ローカルとリモート両方の、メモリ記憶装置を含むコンピュータ記憶媒体に位置することができ、クライアントノードは、交替で、サーバノードとして振舞うこともできる。

【００２１】

このように、図１は本発明を実装することのできる適当な計算処理システム環境１００の一例を示すが、上記で明らかにしたように、この計算処理システム環境１００は、適当な計算処理環境の一例にすぎず、本発明の使用または機能の範囲についてのどのような限定も示唆するものではない。また、この計算処理環境１００は、例示的動作環境１００に示すコンポーネントのいずれか１つまたはその組み合わせに関連してどのような依存性も要件も持つものではないと解釈すべきである。

【００２２】

図１を参照すると、本発明を実装する例示的システムは、コンピュータシステム１１０の形態で汎用計算処理装置を含む。コンピュータシステム１１０のコンポーネントには、それだけに限らないが、処理装置１２０、システムメモリ１３０、およびシステムメモリを含む様々なシステムコンポーネントを処理装置１２０に結合するシステムバス１２１が含まれ得る。システムバス１２１は、様々なバスアーキテクチャのいずれかを使用するメモリバスまたはメモリ制御機構、周辺バス、およびローカルバスを含むいくつかの種類のバス構造のいずれでもよい。例をあげると、そうしたアーキテクチャには、それだけに限らないが、業界標準アーキテクチャ（ＩＳＡ）バス、マイクロチャネルアーキテクチャ（ＭＣＡ）バス、拡張ＩＳＡ（ＥＩＳＡ）バス、ビデオ電子標準協会（ＶＥＳＡ）ローカルバス、および周辺装置相互接続（ＰＣＩ）バス（メザニンバスともいう）が含まれる。

【００２３】

コンピュータシステム１１０は、通常、様々なコンピュータ可読媒体を含む。コンピュータ可読媒体は、コンピュータシステム１１０からアクセスできる任意の使用可能媒体とすることができ、揮発性と不揮発性両方、取り外し可能と取り外し不能両方の媒体を含む。例をあげると、それだけに限らないが、コンピュータ可読媒体にはコンピュータ記憶媒体および通信媒体が含まれ得る。コンピュータ記憶媒体には、コンピュータ可読命令、データ構造、プログラムモジュールまたはその他のデータなどの情報の記憶のための任意の方法または技術で実装された、揮発性および不揮発性、取り外し可能および取り外し不能媒体が含まれる。コンピュータ記憶媒体には、それだけに限らないが、ランダムアクセスメモリ（ＲＡＭ）、読取り専用メモリ（ＲＯＭ）、電気的消去・プログラム可能読取り専用メモリ（ＥＥＰＲＯＭ）、フラッシュメモリその他のメモリ技術、コンパクトディスク読取り専用メモリ（ＣＤＲＯＭ）、書き換え可能コンパクトディスク（ＣＤＲＷ）、ディジタル多用途ディスク（ＤＶＤ）その他の光ディスク記憶装置、磁気カセット、磁気テープ、磁気ディスク記憶装置その他の磁気記憶装置、または所望の情報の記憶に使用することができ、コンピュータシステム１１０からアクセスすることのできる他の任意の媒体が含まれる。通信媒体は、通常、搬送波その他の搬送機構などの被変調データ信号としてコンピュータ可読命令、データ構造、プログラムモジュールまたはその他のデータを実施し、任意の情報送達媒体を含む。「被変調データ信号」という用語は、その特性セットの１つまたは複数を有する信号、またはその信号中に情報を符号化するように変更された信号を意味する。例をあげると、それだけに限らないが、通信媒体には、有線ネットワークや直接配線接続などの有線媒体、および音響、無線周波数、赤外線その他の無線媒体などの無線媒体が含まれる。また、上記のいずれかの組み合わせもコンピュータ可読媒体の範囲内に含むべきである。

【００２４】

システムメモリ１３０は、読取り専用メモリ（ＲＯＭ）１３１およびランダムアクセスメモリ（ＲＡＭ）１３２などの揮発性および／または不揮発性メモリの形態でコンピュー

タ記憶媒体を含む。基本入出力システム１３３（ＢＩＯＳ）は、起動時などに、コンピュータシステム１１０内の要素間での情報転送を支援する基本ルーチンを含み、通常は、ＲＯＭ１３１に記憶される。ＲＡＭ１３２は、通常、処理装置１２０から直ちにアクセス可能であり、かつ／または現在それによって操作されているデータおよび／またはプログラムモジュールを含む。例をあげると、それだけに限らないが、図１にはオペレーティングシステム１３４、アプリケーションプログラム１３５、その他のプログラムモジュール１３６、およびプログラムデータ１３７が示されている。

【００２５】

コンピュータシステム１１０は、その他の取り外し可能／取り外し不能、揮発性／不揮発性コンピュータ記憶媒体を含むこともできる。例としてあげるにすぎないが、図１には、取り外し不能、不揮発性磁気媒体の読取りおよび書込みを行うハードディスクドライブ１４１、取り外し可能、不揮発性磁気ディスク１５２の読取りおよび書込みを行う磁気ディスクドライブ１５１、およびＣＤ－ＲＯＭ、ＣＤＲＷ、ＤＶＤその他の光媒体などの取り外し可能、不揮発性光ディスク１５６の読取りおよび書込みを行う光ディスクドライブ１５５が示されている。この例示的動作環境で使用することのできる他の取り外し可能／取り外し不能、揮発性／不揮発性コンピュータ記憶媒体には、それだけに限らないが、磁気テープカセット、フラッシュメモリカード、ディジタル多用途ディスク、ディジタルビデオテープ、固体ＲＡＭ、固体ＲＯＭなどが含まれる。ハードディスクドライブ１４１は、通常、インターフェイス１４０などの取り外し不能メモリインターフェイスを介してシステムバス１２１に接続され、磁気ディスクドライブ１５１および光ディスクドライブ１５５は、通常、インターフェイス１５０などの取り外し可能メモリインターフェイスによってシステムバス１２１に接続される。

【００２６】

図１に示す前述の各ドライブおよびそれらに関連するコンピュータ記憶媒体は、コンピュータシステム１１０用のコンピュータ可読命令、データ構造、プログラムモジュールおよびその他のデータの記憶を提供する。図１では、例えば、ハードディスクドライブ１４１はオペレーティングシステム１４４、アプリケーションプログラム１４５、その他のプログラムモジュール１４６、およびプログラムデータ１４７を記憶するものとして示されている。これらのコンポーネントは、オペレーティングシステム１３４、アプリケーションプログラム１３５、その他のプログラムモジュール１３６、およびプログラムデータ１３７と同じでも、異なっていてもよい。オペレーティングシステム１４４、アプリケーションプログラム１４５、その他のプログラムモジュール１４６、およびプログラムデータ１４７には、それらが少なくとも異なるコピーであることを示すために、ここでは異なる番号が付されている。ユーザは、キーボード１６２、および、一般にマウス、トラックボールまたはタッチパッドと呼ばれるポインティングデバイス１６１を介してコンピュータシステム１１０にコマンドおよび情報を入力することができる。他の入力装置（図示せず）には、マイク、ジョイスティック、ゲームパッド、衛星放送受信アンテナ、スキャナなどが含まれる。上記その他の入力装置は、しばしば、システムバス１２１に結合されたユーザ入力インターフェイス１６０を介して処理装置１２０に接続されるが、並列ポート、ゲームポート、ユニバーサルシリアルバス（ＵＳＢ）など他のインターフェイスおよびバス構造によって接続されてもよい。システムバス１２１には、ビデオメモリ（図示せず）と交替でやりとりすることのできる、ビデオインターフェイス１９０などのインターフェイスを介してモニタ１９１または他の種類の表示装置を接続することもできる。モニタ１９１とは別に、コンピュータシステムは、スピーカ１９７およびプリンタ１９６など他の周辺出力装置を含むこともでき、出力周辺インターフェイス１９５を介してそれらを接続することができる。

【００２７】

コンピュータシステム１１０は、リモートコンピュータ１８０など、１つまたは複数のリモートコンピュータへの論理接続を使用してネットワーク化または分散環境で動作することもできる。リモートコンピュータ１８０は、パーソナルコンピュータ、サーバ、ルー

タ、ネットワークＰＣ、ピア装置またはその他一般のネットワークノードとすることができ、通常は、コンピュータシステム１１０に関して前述した要素の多くまたはすべてを含むが、図１ではメモリ記憶装置１８１だけが示されている。図１に示す論理接続はローカルエリアネットワーク（ＬＡＮ）１７１、広域ネットワーク（ＷＡＮ）１７３を含むが、他のネットワーク／バスを含むこともできる。そうしたネットワーク環境は、家庭、事務所、企業規模のコンピュータネットワーク、イントラネットおよびインターネットでは一般的である。

【００２８】

ＬＡＮネットワーク環境で使用されるときは、コンピュータシステム１１０は、ネットワークインターフェイスまたはアダプタ１７０を介してＬＡＮ１７１に接続される。ＷＡＮネットワーク環境で使用されるときは、コンピュータシステム１１０は、通常、モデム１７２、またはインターネットなどのＷＡＮ１７３を介して通信を確立するその他の手段を含む。モデム１７２は、内蔵でも外付けでもよく、ユーザ入力インターフェイス１６０、またはその他適当な機構を介してシステムバス１２１に接続することができる。ネットワーク化環境では、コンピュータシステム１１０に関連して示したプログラムモジュール、またはその一部は、リモートのメモリ記憶装置に記憶することもできる。例をあげると、それだけに限らないが、図１にはリモートアプリケーションプログラム１８５がメモリ装置１８１上にあるものとして示されている。図示のネットワーク接続は例示的なものであり、コンピュータ間で通信リンクを確立する他の手段を使用することもできることが理解されるであろう。

【００２９】

パーソナルコンピューティングおよびインターネットの集中を考慮して様々な分散計算処理フレームワークが開発されており、また開発中である。個人も業務ユーザも同様に、アプリケーションおよび計算処理装置用のシームレスに相互運用可能な、Ｗｅｂ対応のインターフェイスを備え、計算処理活動をますますＷｅｂブラウザやネットワーク指向のものとしている。

【００３０】

例えば、米国ワシントン州９８０５２、レッドモンド、ワンマクロソフトウェイのマイクロソフト社（Microsoft Corporation, One Microsoft Way, Redmond, Washington 98052）から入手可能なＭＩＣＲＯＳＯＦＴ（登録商標）の．ＮＥＴ（商標）プラットフォームは、サーバ、Ｗｅｂベースのデータ記憶などの構成要素サービス、およびダウンロード可能なデバイスソフトウェアを含む。本明細書では例示的実施形態を計算処理装置上にあるソフトウェアとの関連で説明するが、本発明の１つまたは複数の部分を、オペレーティングシステム、アプリケーションプログラミングインターフェイス（ＡＰＩ）、またはコプロセッサと表示装置と要求側オブジェクトのいずれかの間の「仲介者（ｍｉｄｄｌｅｍａｎ）」オブジェクトを介して実装し、それによって、本発明による動作を．ＮＥＴ（商標）言語およびサービスのすべてによって実施し、そこでサポートし、あるいはそれらを介してアクセスすることもでき、他の分散コンピューティングフレームワークでも同様である。

【００３１】

例示的実施形態

図２に、本発明の諸態様を実施するシステムアーキテクチャの一例の構成図を示す。中間言語コンパイラおよびターゲット実行エンジンを含む問い合わせ実行時アーキテクチャ２００が示されている。このアーキテクチャは、複数のデータソース上でのＸＭＬ問い合わせのためのデータを受け取り、コンパイルし、解釈し、それにアクセスするように実装された基本的ブロックの例を示す。ＸＭＬまたはＸＭＬ関連の標準に適合する複数のフロントエンド言語コンパイラ２１０を実現することができる。元のまたは入力問い合わせは、図１に示すような複数の入力装置のいずれか１つから受け取り、または入力することができる。図２に戻ると、フロントエンドコンパイラ２１０は、入力問い合わせの意味のコンパイル済みＸＭＬ中間言語表現２１５を生成するために、ＸＭＬ問い合わせ、ＸＭＬビ

ューまたはその他の関連するＸＭＬ言語照会（ｉｎｑｕｉｒｙ）を受け取ることができる。フロントエンドコンパイラ用の典型的な言語の種類には、ワールドワイドウェブコンソーシアム（Ｗ３Ｃ（登録商標））によって公開されている、ＸＰａｔｈ、ＸＭＬスタイルシート言語（ＸＳＬ）、ＸＳＬＴ、ＸＭＬ問い合わせ言語（ＸＱｕｅｒｙ）の標準が含まれる。ＸＭＬビューコンパイラを使用することもでき、特に、ＸＱｕｅｒｙのＷ３Ｃ（登録商標）ＸＭＬ標準ビューを含むこともできる。問い合わせおよびビュー言語と複数のターゲットモデルとの間の中間言語抽象化により、仮想ＸＭＬデータのみならず、Ｍｉｃｒｏｓｏｆｔ（登録商標）．ＮＥＴ（商標）からのものなど、実ＸＭＬデータもデータソースとして使用できるようになる。

【００３２】

複数のバックエンドターゲットジェネレータ２２０を用いて複数の関連するターゲット問い合わせ言語実行エンジン２３０をサポートすることができる。それらバックエンドターゲットジェネレータ２２０はそれぞれ、サポートされる各データソース内のデータモデルに合わせてデータソース上で効率良く機能するように構築することができる。例えば、ＳＱＬデータベースソース用ターゲット問い合わせ言語生成プログラムは、ＳＱＬデータベース管理システムを用いて関係データベースに問い合わせする際の効率を高めるために最適化することができる。したがって、例えば、ターゲット問い合わせ言語生成プログラム２２０ａを実行エンジン２３０ａとの対にして、中間言語表現２１５を特定のデータソース２４０と互換性のあるターゲット機械語に変換することもできる。ターゲットジェネレータは、任意選択で、ＸＭＬ中間言語表現２１５を、例えばマイクロソフト（登録商標）中間言語（ＭＳＩＬ）など、別の中間言語に変換し、それによって実行エンジンが１つまたは複数のデータソース２４０に直接問い合わせを行えるようにすることもできる。

【００３３】

コンパイル済み問い合わせの実行時に、実行エンジンは、普通、例えば、さらなる処理、記憶、ユーザへの表示、または後続ソフトウェアアプリケーションへの提供に使用可能な問い合わせ結果２５０を生成する。ターゲットジェネレータ２２０および対応する実行エンジン２３０は、例えば、ＸＭＬやＳＱＬなどの問い合わせ言語を含むことができる。

【００３４】

このアーキテクチャの別の一態様は、データ使用可能性のモジュール性である。例えば、ＸＭＬ中間言語表現２１５が生成される点において、出力を生成し、それによってその中間言語表現自体を別のシステムで使用可能にし、または遅延ターゲットコンパイルに使用可能にすることもできる。また、ＸＭＬ中間言語自体を、まず特定の非ＸＭＬ中間言語命令問い合わせに実際に変換せずに、直接実行することもできる。したがって、ＸＭＬ中間言語表現を実行するように特に適合された実行エンジンを使用して、コンパイラなしで、ＸＭＬ中間言語を用いて１つまたは複数のデータソースに問い合わせを行うこともできる。システム出力での別のオプションとして、ターゲットジェネレータの出力を、別個のシステムでの実行または別の非システムアプリケーションによる実行のための出力として使用することもできる。

【００３５】

図２の例示的アーキテクチャは、問い合わせ構築に際して、本質的に、大きな柔軟性を有することに留意すべきである。この問い合わせアーキテクチャは、複数の問い合わせからなる中間言語複合を生成できるように、複数の問い合わせを入力できるようにする。さらに、複数のバックエンドターゲットジェネレータおよび実行エンジンを適切に使用して、異なるデータモデルのデータソースに問い合わせをすることもできる。アーキテクチャ２００は、フロントエンドおよびバックエンドの順列数の低減も可能にする。図２には、Ｍ個のフロントエンドコンパイラとＮ個のバックエンドコンパイラが示されている（ＭおよびＮは整数である）。普通、この組み合わせによって、合計Ｍ×Ｎ通りの可能なシステムパスが生じるはずである。しかし、共通の中間言語の利用により、順列の数は、有利にも、Ｍ＋Ｎ通りまで低減される。

【００３６】

図２に示すように生成されるＸＭＬ中間言語は、ＸＭＬ問い合わせまたはビューの一表現である。したがって、それを問い合わせ中間言語（ＱＩＬ）と呼ぶことができる。というのは、それがＸＭＬ問い合わせの意味の明示的表現であるからである。この問い合わせ中間言語（ＱＩＬ）を、すべてのＸＭＬ問い合わせおよびビュー言語コンパイラで共通の意味論的表現とみなすこともできる。ＱＩＬは、通常の抽象構文ツリー（ＡＳＴ）に類似するものであるが、ＱＩＬが言語の構文ではなく、問い合わせの意味体系、または意味を捕捉するという点で異なる。別の違いは、ＱＩＬがグラフ構造であり、ＡＳＴのようなツリー構造ではない点である。

【００３７】

ＱＩＬは、様々な異なるターゲットデータソース（関係データや非関係データなど）にわたる、複数の異なるＸＭＬ問い合わせ言語およびビュー定義言語（ＸＰａｔｈ、ＸＳＬＴ、ＸＱｕｅｒｙなど）の抽象化を可能にする。したがって、ＱＩＬは、互換性のあるＸＭＬ言語のすべてをサポートする共通構造を可能にする。あらゆる演算は、明示的でしかも曖昧ではなく、ＱＩＬ生成の際に支援するフロントエンドコンパイルを、ＱＩＬを使用するバックエンドエンジンから完全に切り離すものであることが望ましい。

【００３８】

問い合わせ中間言語定義の例がワシントン州レッドモンドのマイクロソフトコーポレーションに譲渡された"QUERY INTERMEDIATE LANGUAGE METHOD AND SYSTEM"という名称の同日出願の米国特許出願で提供されている。

【００３９】

図３に、共通問い合わせ実行時アーキテクチャ（例えば、図２に関連して述べた例示的アーキテクチャ）のアプリケーションプログラミングインターフェイス（ＡＰＩ）の一例を表す流れ図を示す。ＡＰＩはシステム中にあるものとみなされ（図２の例など）、そのシステムを利用するアプリケーションにそのアプリケーションとやりとりできるようにする。この通信により、システムリソースがアプリケーションから利用できるようになる。図３に戻ると、入力問い合わせを受け取り、その問い合わせをＸＭＬ中間言語にコンパイルし、その中間言語表現をターゲット言語に変換し、そのターゲットコンパイルを実行し、問い合わせ結果を生成するプロセスが示され、例示的アプリケーションインターフェイスが強調されている。

【００４０】

最初に、アプリケーションが、入力問い合わせ３１０の受け取りを提供し（３１５）またはそれを認識する。このアプリケーションは、問い合わせシステムとインターフェイスして、ソフトウェア環境制御パラメータおよびコンパイル制御オプションを設定する（３２０）よう求める１つまたは複数の呼び出しを行うことができる。このステップでは、１つまたは複数のフロントエンドコンパイラの構成に、入力問い合わせを受け入れさせ、その入力問い合わせのコンパイルのためにソフトウェアおよび制御されたハードウェアを準備させる。このアプリケーションは、監視機能３９０への状況呼び出しを介して、アプリケーションが所望するパラメータ設定の状況および進捗を監視する（３２２）ことができる。

【００４１】

それらの問い合わせシステムパラメータが問い合わせシステムに渡され（３２５）、次いで、アプリケーションは問い合わせシステムとインターフェイスして、入力問い合わせ３１０の中間言語表現３３０がその入力問い合わせのＸＭＬ意味論的意味になるようにコンパイルすることができる。この意味論的意味は、遅延処理または別のシステムでの使用のために、アプリケーション制御を介してシステム３３０から出力または監視することもできる。監視機能３９０は、アプリケーションが、中間言語を生成する際のシステムの性能、ならびに制御および構成設定の登録を監視できるようにする。中間言語表現が生成されると、それをユーザのために監視または出力機能に渡す（３３２）ことができ、必要に応じて別のアプリケーションに渡すこともできる。あるいは、その中間言語表現を、ターゲットジェネレータによるコンパイルのためにソフトウェアモジュールに渡す（３３４）

こともできる。代替として、その中間言語表現を、直接実行のために問い合わせ実行エンジンに渡す（３３６）こともできる。

【００４２】

中間言語表現がターゲットジェネレータに渡された場合には、アプリケーションは、ターゲットジェネレータを選択し、コンパイラを適切に構成し、ターゲットジェネレータの出力を生成するよう求める問い合わせシステムへの呼び出し（３４０）を生成することができる。前述のように、このアプリケーションはターゲットジェネレータの設定および活動を監視し（３４２）、それによってアプリケーションおよび問い合わせシステムの両方の動作を保証することができる。前述したように、この時点でアプリケーションを介して問い合わせシステムを停止または中断することができる。というのは、使用可能な生成物、すなわちターゲット固有の問い合わせが生成されているからである。このターゲット固有の問い合わせは、将来の使用のために記憶することもでき、別のシステムが１つまたは複数のデータソースに問い合わせをするために使用することもできる。

【００４３】

問い合わせシステム処理を続行する場合には、このターゲット別問い合わせを、その問い合わせを実行することのできる実行エンジンに渡す（３４５）ことができる。アプリケーションは、１つまたは複数のデータソース３８０に対してそのターゲット別問い合わせ３５６を実行するよう求める呼び出し３５０を生成することができる。前述のように、このアプリケーションは、問い合わせシステムの適正な動作を保証するために、実行エンジンの設定および活動を監視する（３５２）ことができる。

【００４４】

実行が完了すると、実行エンジンは、問い合わせ結果をアキュムレータに渡し（３５５）、そこで問い合わせシステムは、必要に応じて、その後の処理のために問い合わせ結果を一時的に保持する（３６０）こともできる。この問い合わせ結果には、アプリケーションからの呼び出し３６２を介してアクセスすることができ、その結果を転送または監視することもできる。

【００４５】

アプリケーションプログラミングインターフェイスの例

以下に、本発明によるＡＰＩインターフェイスの例を示す。

Ｉ．フロントエンドコンパイラコマンド：

問い合わせコンパイラを使用して、様々なＸＭＬ問い合わせ言語からXmlExpression、すなわちＱＩＬをカプセル化したものを生成する。以下に、３つのＸＭＬ問い合わせ言語用コンパイラ例を示す。それらは、XPathCompiler、XsltCompiler、およびXQueryCompilerである。これらのコンパイラの典型的使用法には、XmlenvironmentBaseの設定および問い合わせのコンパイルが関与する。

【００４６】

その適切なXmlCompilerEnvironmentBaseコンパイラ環境を設定することを利用して、コンパイル時に、その問い合わせとは無関係の情報をコンパイラに提供することができる。例えば、ＸＰａｔｈ構文だけでは、他のネーム空間への問い合わせを行うことはできない。コンパイラ環境上のXmlNamespaceResolverが、接頭部を適切なネーム空間ＵＲＩにマッピングするように設定された場合には、そのＸＰａｔｈ問い合わせは、他のネーム空間に問い合わせすることができる。コンパイラ環境は、大部分の問い合わせの実行には必要とされないことに留意されたい。

【００４７】

問い合わせのコンパイルは、普通、適切なコンパイラ環境が指定された後で行われる。その時点で、ユーザまたはその他の入力元が実際の問い合わせを提供し、コンパイルすることができる。コンパイルの結果がXmlExpressionである。問い合わせは、しばしば、関連する証拠（Ｅｖｉｄｅｎｃｅ）オブジェクトと共に提供される。この証拠オブジェクトは、所与の問い合わせの信頼性、したがって、それがどの動作を実施することができ、またはできないかを示し、したがって、その問い合わせにあるセキュリティレベルを適用す

る。

【００４８】

証拠に基づくセキュリティによって、ユーザは、セキュリティ許可を介して、その問い合わせ自体をどの程度実行することができるかを正確に指定できるようになる。例えば、全くアクセスが許されない場合は、例えばデータベースへの接続など、どのようなデータアクセスを要求するどのような問い合わせも失敗するはずである。唯一成功すると思われる問い合わせは、静的結果（例えば、「ｎｕｍｂｅｒ（３＋５）」）を返すものであるが、その理由は、データベースからのデータにアクセスしないからである。読取り専用アクセスが許される場合は、更新は実行されないはずである。データソースは、全く変更されないことが保証される。これは、ＸｐａｔｈナビゲータＡＰＩの諸機能に対応する。付加アクセスが許される場合は、ＩＮＳＥＲＴ（挿入）ステートメントだけが実行されるはずである。これは、元のデータが全く変更されず、付加だけが行わることを保証する。無制限のアクセスが許される場合には、どのような更新も実行できるはずである。これは、データソースのいずれかに含まれるデータの完全削除を含むものである。これは、ＸｐａｔｈエディタＡＰＩの諸機能に対応する。

【００４９】

【表１－１】

Ａ.	ＸＰａｔｈコンパイラクラス	
１.	コマンド：	XPathComplier(); XPathCompiler(XmlCompilerEnviornmentBase XmlCompilerEnvironment);
	説明：	このクラスのインスタンスを構築するメソッドコマンド。XmlCompilerEnviornmentBaseが提供される場合は、それがそのインスタンスのXmlCompilerEnviornmentとして設定される。
	引数：	XmlCompilerEnviornment; このクラスのインスタンス上でXmlCompilerEnviornmentとして設定するためのXmlCompilerEnviornmentBase
２.	コマンド：	OnCompilationEvent;
	説明：	コンパイル時にメッセージを受け取るために登録することのできるイベント。コンパイラは、ユーザがそれに基づいて処置を取ることのできる警告およびその他のメッセージを報告することができる。このイベントは、コンパイルを妨げない。

【００５０】

【表１－２】

３．　コマンド：　XmlCompilerEnvironmentBase XmlCompilerEnvironment { get; set; }
　　説明：　ＸＭＬコンパイラ環境特性。ユーザが使用するXmlCompilerEnvironmentBaseを取得または設定できるようにする。このコンパイラ環境は、その問い合わせ自体では伝達されない情報を問い合わせコンパイラに提供する。

４．　コマンド：　XmlExpression Compile(string queryText);
　　説明：　コンパイルメソッド。この関数は、所与の問い合わせ、すなわちＸＰａｔｈ問い合わせをコンパイルし、所与の問い合わせのＱＩＬ表現であるXmlExpressionを返す。所与の問い合わせの信頼性は、望ましくは、呼び出し元コードの信頼性と同一に設定される。
　　引数：　query;
　　XmlExpressionにコンパイルされる問い合わせ。コンパイル済み問い合わせであるＸＭＬ式を返す。

Ｂ．　ＸＳＬＴコンパイラクラス－XsltCompilerはＸＳＬＴスタイルシートをXmlExpressionsにコンパイルするように設計される。

１．　コマンド：　XsltCompiler();
　　XsltCompiler(XmlCompilerEnvironmentBase XmlCompilerEnvironment);
　　説明：　コンパイラメソッド。このクラスのインスタンスを構築する。XmlCompilerEnvironmentBaseが提供される場合には、それがそのインスタンスに対するXmlCompilerEnvironmentとして設定される。
　　引数：　XmlCompilerEnvironment;
　　このクラスのインスタンス上でXmlCompilerEnvironmentとして設定するためのXmlCompilerEnvironmentBase

２．　コマンド：　OnCompilationEvent;
　　説明：　コンパイル時にメッセージを受け取るために登録することのできるイベント。コンパイラは、ユーザがそれに基づいて処置を取ることのできる警告およびその他のメッセージを報告することができる。このイベントはコンパイルを妨げない。

３．　コマンド：　XmlCompilerEnvironmentBase XmlCompilerEnvironment { get;set; }
　　説明：　ユーザが、使用するXmlCompilerEnvironmentBaseを取得または設定できるようにする特性。このコンパイラ環境は、コンパイルされる実際の問い合わせに加えて情報も問い合わせコンパイラに提供する。XmlCompilerEnvironmentのデフォルトのインスタンス。

【００５１】

【表１－３】

４．	コマンド：	XmlExpression Compile(string queryUri, XmlResolver resolver); XmlExpression Compile(string queryUri, XmlResolver resolver, Evidence evidence);
	説明：	コンパイルメソッド。queryUriはXmlResolverパラメータを介して解決され、ＸＳＬＴスタイルシートとしてコンパイルされる。XmlResolverパラメータは、元のスタイルシートをコンパイルするのに使用される任意のxsl:includeおよびxsl:importステートメントの解決にも使用される。渡された証拠は、所与の問い合わせの信頼性、およびその後にそれがどの動作を実行できるかを決定するために使用される。証拠なしの多重定義（ｏｖｅｒｌｏａｄ）が使用される場合は、所与のＵＲＩを使用してその問い合わせに適切な証拠を生成する。コンパイル済み問い合わせであるXMLExpressionを返す。
	引数：	queryUri; コンパイルするＸＳＬＴスタイルシートを取り出すために解決するＵＲＩ。 resolver; XmlResolverは、ＸＳＬＴスタイルシート、およびそのスタイルシート中の任意のxsl:importおよびxsl:includeステートメントを解決するのに使用される。 evidence; 所与の問い合わせの信頼性を決定するセキュリティ証拠。
５．	コマンド：	XmlExpression Compile(XmlReader queryText, XmlResolver resolver); XmlExpression Compile(XmlReader queryText, XmlResolver resolver, Evidence evidence);
	説明：	コンパイルメソッド。この関数は、XmlReaderを介して渡されたＸＳＬＴ問い合わせを表すXmlExpressionを返す。渡された証拠は、所与の問い合わせの信頼性、およびその後にそれがどの動作を実行できるかを決定するのに使用される。証拠を取らない多重定義が使用される場合は、コンパイラは、IDataEvidenceインターフェイスを介して適切な証拠を得ようとする。このインターフェイスが実装されない場合は、適切な例外となる。コンパイル済み問い合わせであるXmlExpressionを返す。
	引数：	query; XmlExpressionにコンパイルされるＸＳＬＴを含むXmlReader。 resolver; このXmlResolverを使用して、所与のＸＳＬＴ中のxsl:importおよびxsl:includeステートメントを解決する。 evidence; 所与の問い合わせの信頼性を決定するセキュリティ証拠。

【００５２】

【表１－４】

Ｃ．	ＸＱｕｅｒｙコンパイラクラス	
１．	コマンド：	XQueryCompiler(); XQueryCompiler(XmlCompilerEnvironmentBase XmlCompilerEnvironment);
	説明：	コンパイラメソッド。このクラスのインスタンスを構築する。XmlCompilerEnvironmentBaseが提供される場合には、それがそのインスタンスのXmlCompilerEnvironmentとして設定される。
	引数：	XmlCompilerEnvironment; このクラスのインスタンス上でXmlCompilerEnvironmentとして設定するためのXmlCompilerEnvironmentBase。
２．	コマンド：	OnCompilationEvent;
	説明：	コンパイル時にメッセージを受け取るために登録することのできるイベント。コンパイラは、ユーザがそれに基づいて処置を取ることのできる警告およびその他のメッセージを報告することができる。このイベントはコンパイルを妨げない。
３．	コマンド：	XmlCompilerEnvironmentBase XmlCompilerEnvironment { get; set; }
	説明：	ＸＭＬコンパイラ環境特性。ユーザが使用するXmlCompilerEnvironmentBaseを取得または設定できるようにする。このコンパイラ環境は、コンパイルされる実際の問い合わせに加えて情報も問い合わせコンパイラに提供する。これはXmlCompilerEnvironmentのデフォルトのインスタンスである。
４．	コマンド：	XmlExpression Compile(TextReader queryText);
	説明：	コンパイルメソッド。この関数は、渡されたＸＱｕｅｒｙ問い合わせを表すXmlExpressionを返す。渡された証拠は、その問い合わせに何を実行させるべきか決定するのに使用される。
	引数：	query; XmlExpressionにコンパイルされる問い合わせ。 evidence; 所与の問い合わせの信頼性を決定するセキュリティ証拠。

【００５３】

【表１－５】

５．	コマンド：	XmlExpression Compile(string queryText);
	説明：	この関数は、所与の問い合わせのXmlExpressionをコンパイルし、それを返す。提供される証拠は、所与の問い合わせの信頼性を示す。
	引数：	query; XmlExpressionにコンパイルされる問い合わせを含む文字列。 evidence; 所与の問い合わせの信頼性を決定するセキュリティ証拠。
６．	コマンド：	XmlExpression Compile(string queryUri, XmlResolver resolver);
	説明：	queryUriがXmlResolverパラメータを介して解決され、ＸＱｕｅｒｙとしてコンパイルされる。コンパイル済み問い合わせであるXmlExpressionを返す。
	引数：	queryURI; コンパイルするＸＱｕｅｒｙを取り出すために解決するＵＲＩ。 resolver; コンパイルするＸＱｕｅｒｙを解決するのに使用されるXmlResolver。

【００５４】

ＩＩ．コンパイラ環境コマンド

一般に、コンパイラ環境は、問い合わせコンパイラが実際の問い合わせをコンパイルするのを支援するための補足的情報を提供するために使用される。具体的には、このインターフェイスを使用して、問い合わせをコンパイルするのに必要とされ得る外部関数、変数、およびデフォルト文書を解決する。すべての解決がXmlExpressionを返す必要があることに留意することは重要である。

【００５５】

関数解決の一例が、XmlViewSchemaおよびmap:view（）関数である。問い合わせ言語は、組み込みマッピングという概念を持たず、したがって、これを可能にする拡張を必要とする。XmlViewSchemaは、map:view()関数をXmlExpressionに解決することができ、したがって、個々のコンパイラで使用することができる。変数解決もほぼ同様に動作する。コンパイラは、その問い合わせ構文中で宣言されていない変数をまたいで読み取ることもできる。これは、ＸＰａｔｈがそのコンパイラ環境を必要とする別の領域である。コンテキスト文書解決では、あらゆるＸＭＬ問い合わせ言語は、デフォルト文書またはコンテキスト文書に問い合わせを行う必要がある。

【００５６】

XmlCompilerEnvironmentBaseは抽象クラスであり、完全なコンパイラ環境である。このクラスだけが、所与のコンパイラが必要とする様々な項目（ｉｔｅｍ）を解決することができ、各実装による動的演算の実施を可能にする。XmlCompilerEnvironmentBaseは問い合わせコンパイラが利用する環境である。これは、任意の所与のコンパイラが必要とする可能性のある大部分のオプションの解決を提供することができる。これは、IXmlCompilerIncludeおよびIXmlNamespace上のメソッドすべてを実装することができる。XmlCompilerEnvironmentBaseは、IXmlCompilerIncludeの実装に加えて、スキーマ、ネーム空間、その問い合わせをコンパイルすべき方法など、他の項目の解決も可能にする。

【００５７】

ＸＱｕｅｒｙは、問い合わせのコンパイル時に、Ｗ３Ｃ（登録商標）ＸＭＬスキーマ（ＸＭＬスキーマ定義－ＸＳＤ）を利用して静的分析を実施することができる。このコンパイラ環境はそれらのスキーマの解決を提供する。そのネーム空間／接頭部解決を利用すれ

ば、ユーザは、その都度問い合わせ自体の中で指定することを必要とせずにコンパイル間でよく使用される接頭部－ネーム空間結合を再利用することができる。この機能は、IXmlNamespaceResolverを実装することによって提供することができる。ユーザが一層効率良く問題を見つけられるように、ビルド型ＡＰＩを提供することもできる。

【００５８】

XmlEtivironmentBaseのための解決を提供するのに加えて、それらの解決済み項目をその環境に付加する方法も提供される。このXmlCompilerEnvironmentは、コンパイラで使用するために、少なくとも３つの異なる方法を介して諸関数を付加することができる。XmlCompilerEnvironmentは、IXmlCompilerIncludeの実装を取り込み、それらを所与の関数の解決に使用することができる。基本的には、それらの付加された実装に「ResolveFunction」メソッド呼び出しが渡され、それによってそれらの実装がその解決を試みることができるようにする。また、このコンパイラ環境は、XmlExpressionの形態で、事前コンパイル済みの問い合わせライブラリを付加することもできる。問い合わせライブラリは、単一のXmlExpressionにコンパイルされるユーザ定義のＸＱｕｅｒｙ関数群として定義することができる。こうして、任意の問い合わせ言語が、外部関数を呼び出すことによって、ＸＱｕｅｒｙの能力を利用することができる。パラメータは、所与の関数の問い合わせライブラリに適切にマッピングさせることができる。このコンパイラ環境は、問い合わせライブラリではないXmlExpressionを付加することもできる。個々のコンパイル済み問い合わせ（ＸＰａｔｈやＸＳＬＴなど）を問い合わせ関数として付加することもできる。

【００５９】

XmlCompilerEnvironmentは、様々なメソッドを使用して変数を付加することもできる。XmlCompilerEnvironmentは、IXmlCompilerIncludeの諸実装を取り込み、それらを所与の変数の解決に使用することができる。基本的には、付加された実装に「ResolveVariable」メソッド呼び出しが渡され、それによってそれらの実装がその解決を試みることができるようにする。このコンパイラ環境は、既存のXmlExpressionを変数として利用させることもできる。これは、XmlExpressionを関数として付加するのと同様である。これは、ユーザが、特定の言語の諸機能を別の言語で利用できるようにする。汎用オブジェクトを変数として付加することにより、ユーザは、その問い合わせ内のオブジェクトの値にバインドすることができるようになる。それらのオブジェクトは、その問い合わせでの使用に適したＸＳＤ型および値に変換される。

【００６０】

XmlCompilerEnvironmentは、問い合わせコンパイラによって使用されるコンテキスト文書を設定することができる。XmlCompilerEnvironment上に適当なコンテキスト文書を設定する少なくとも３つの方法がある。IXmlCompilerIncludeを実装するオブジェクトを使用して問い合わせのコンテキスト文書を設定することもできる。XmlCompilerEnvironmentは、その文書の解決のために、単に、ResolveContextDocument()メソッド呼び出しをIXmlCompilerIncludeにパイプ接続することもできる。この一例がXmlViewSchemaである。これは、ユーザが、map:view()関数を使用せずにマッピングを利用できるようにする。XmlExpressionをコンテキスト文書として設定することもできる。この方式では、ユーザは、問い合わせを介して、マッピングサポートを処理するのとほとんど同じように仮想ＸＭＬビューを作成することができる。ユーザにとっては、極端に単純化したマッピングを公開し、別個の問い合わせ中に付加論理を有する方が、両者を密に結合させるより容易であることもある。ＸＳＬＴやＸＱｕｅｒｙなどの言語は、コンテキスト文書と名前付き文書の両方に同時に問い合わせをすることができる。ＵＲＩへのコンテキスト文書解決を設定することによって、そのコンテキスト文書を、それが暗黙的に提供されるのではなく、動的に解決されるように指定変更することができる。これは、ユーザが、コンテキスト文書とXmlResolverの組み合わせではなく、問い合わせ先の単一のXmlResolverを指定することができるので、プログラミングを容易にする。

【００６１】

変数または関数がXmlCompilerEnvironmentに付加される順序は、解決が正常に実施され

る順序に影響を与え得ることに留意すべきである。１つの処理規則は、明示的な名前付き関数／変数が、成功したマッチの有無に関してチェックされることを定める。重複した名前付き関数／変数は、首尾よく付加されないことがある。単一のXmlExpressionに含まれる問い合わせライブラリはこの範疇に含まれることに留意すべきである。別の処理規則は、IXmlCompilerIncludeが、それらが付加された順序に従ってマッチの有無を順次テストされることを定める。IXmlCompilerIncludeは、その他の項目として重複した関数／変数を含むことができる。これが明示的な名前付き関数／変数の複製を含む場合には、それは、明示的な名前付き関数／変数が優先して解決されることになるため、絶対に解決され得ない。別のIXmlCompilerIncludeが重複した関数／変数を含んでいた場合は、最初に付加されたIXmlCompilerIncludeだけが所与の項目を解決することになる。

【００６２】

例えば、XmlCompilerEnvironmentが生成され、関数解決のために２つのIXmlCompilerIncludeを付加した場合には、実際の機能を解決しようとするときに、第１のIXmlCompilerIncludeをチェックすることができる。この解決に失敗した場合は、正常な解決のために第２のIXmlCompilerIncludeがチェックされる。名前付き関数がXmlExpressionと共にこの環境に付加された場合には、それはIXmlCompilerInclude実装の前にチェックされるはずである。というのは、明示的な名前付き関数／変数はIXmlCompilerIncludeの実装の前にチェックすることができるからである。

【００６３】

以下に、コンパイラ環境コマンドの例を示す。

【００６４】

【表２－１】

１．	コマンド：	XmlExpression ResolveFunction(XmlQualifiedName functionName, XmlExpression[] functionParameters);
	説明：	解決関数メソッド。このメソッドは、コンパイラがコンパイル時に外部関数を解決できるようにする。渡される名前および引数は、実装の解決のために使用される。解決に失敗した場合は、関数はヌルを返す。関数の結果を表すXmlExpressionを返す。解決が不成功だった場合は、ヌルが返される。
	引数：	functionName; ルックアップされている関数の完全修飾名。 functionParameters; それを用いて関数が呼び出されているパラメータを表すXmlExpressionの配列。ＸＭＬコンパイラ環境の実装は、解決時にこの情報を利用してパラメータおよび最適化の型を確認する。
２．	コマンド：	XmlExpression ResolveVariable(XmlQualifiedName variableName);
	説明：	このメソッドは、コンパイラがコンパイル時に外部変数を解決できるようにする。変数の完全修飾名が解決に使用される。解決に失敗した場合は、この関数はヌルを返す。変数の値を表すXmlExpressionを返す。解決が不成功だった場合は、ヌルが返される。
	引数：	variableName; ルックアップされている変数の完全修飾名。
３．	コマンド：	XmlExpression ResolveContextDocument();
	説明：	このメソッドは、コンパイラがコンテキスト文書をどのように取り出すかを動的に解決できるようにする。解決に失敗した場合は、このメソッドはヌルを返す。コンテキスト文書をどのように取り出すかを表すXmlExpressionを返す。解決に失敗した場合は、ヌルが返される。
４．	コマンド：	XmlSchema ResolveSchema(string namespace);
	説明：	コンパイラが所与のネーム空間のスキーマを取り出せるようにする。このメソッドは、スキーマを解決できなかった場合には、ヌルを返す。所与のネーム空間のコンパイル済みスキーマであるXmlSchemaオブジェクトを返す。スキーマを解決できなかった場合は、ヌルが返される。
	引数：	namespace; スキーマを解決する対象のネーム空間。

【００６５】

【表２－２】

５．	コマンド：	bool IsDebug { get; };
	説明：	コンパイラがデバッグ特性と共に問い合わせをコンパイルすべきか否かを決定できるようにする特性。
６．	コマンド：	XmlCompilerEnvironment(); XmlCompilerEnvironment(bool IsDebug);
	説明：	新規のXmlCompilerEnvironmentをインスタンス化するメソッド。IsDebugの値を提供する多重定義が提供された場合は、この値は、基準のXmlCompilerEnvironmentクラス上で呼び出されたIsDebugアクセス機構（accessor）上で返される。デフォルトのコンストラクタ（constructor）が使用された場合は、この値はデフォルトで偽になる。
７．	コマンド：	XmlNamespaceManager XmlNamespaceManager { get; set; }
	説明：	XmlNamespaceManager特性は、XmlCompilerEnvironmentBase.ResolvePrefixメソッドおよびXmlCompilerEnvironmentBase.ResolveNamespaceメソッドによる解決に使用される。
８．	コマンド：	XmlSchemaSet XmlSchemaSet { get; set; }
	説明：	このXmlSchemaSet特性は、ＸＭＬスキーマ型情報を取得、設定するためにXmlCompilerEnvironmentBase.ResolveSchemaメソッドによる解決に使用される。
９．	コマンド：	void SetContextDocumentResolution(IXmlCompilerInclude documentResolver);
	説明：	このコンテキスト文書メソッドは、ユーザが、そのコンテキスト文書の解決がIXmlCompilerIncludeの実装から行われるように設定することを可能にする。
	引数：	documentResolver; そのコンテキスト文書を解決することのできるIXmlCompilerIncludeの実装。
１０．	コマンド：	void SetContextDocumentResolution(XmlExpression documentExpression);
	説明：	この文書コンテキストメソッドは、ユーザが、デフォルト文書をXmlExpressionとして設定できるようにする。これは、別の問い合わせまたはマッピングでのＸＳＬＴまたはＸＰａｔｈの合成に際して有用となるはずである。
	引数：	documentExpression; 現在の環境を使用してコンパイルされる問い合わせでデフォルトの文書解決の代わりに合成されるXmlExpression。
１１．	コマンド：	void SetContextDocumentResolution(string contextDocumentUri);
	説明：	この文書コンテキストメソッドは、ユーザが、実行時のその文書の解決時に使用されるデフォルトの文書名を設定できるようにする。これは、１つのデフォルト文書またはデフォルト文書と名前付き文書の１つの組み合わせしかないＸＰａｔｈおよびＸＳＬＴに有用である。
	引数：	contextDocumentUri; 提供されたXmlResolverを介して実行時に解決される名前。

【００６６】

【表２－３】

１２．	コマンド：	void AddFunctions(IXmlCompilerInclude functionResolver)；
	説明：	この関数メソッドは、コンパイラ環境にIXmlCompilerIncludeを付加し、IXmlCompilerIncludeをコンパイル時の関数解決に使用できるようにする。
	引数：	function resolver；
		関数を解決することのできるIXmlCompilerIncludeの実装。
１３．	コマンド：	void AddFunctions(XmlExpression library)；
	説明：	この関数メソッドを使用して既存のXmlExpressionを関数ライブラリとして付加する。例：ユーザは複数のＸＱｕｅｒｙ関数を有し、次いでそれがXmlExpressionにコンパイルされる。次いでユーザはこのXmlExpressionを関数ライブラリとして付加し、それを所与のネーム空間に関連付ける。ユーザは、その後、これらの関数を（ＸＱｕｅｒｙに限らず）他の問い合わせ内で使用することもできる
	引数：	library；
		以前にコンパイル済みの関数を含むコンパイル済みXmlExpression。
１４．	コマンド：	void AddFunction(XmlQualifiedName name, XmlExpression function)；
	説明：	この関数メソッドは、ユーザがXmlExpressionを、他の問い合わせのコンパイルで使用するための関数としてバインドできるようにする。例：ＸＱｕｅｒｙでは、ＸＱｕｅｒｙ全体が単一関数とみなされる。ＸＳＬＴでは、スタイルシート全体が単一関数とみなされる。ＸＰａｔｈでは、ＸＰａｔｈ式全体が単一関数とみなされる。
	引数：	name；
		問い合わせ中でその下で関数が参照される完全修飾名。
		Function；
		バインドされたＸＭＬ式。
１５．	コマンド：	void AddVariables(IXmlCompilerInclude variableResolver)；
	説明：	この変数解決のメソッドは、IXmlCompilerIncludeをコンパイラ環境に付加し、コンパイル時にIXmlCompilerIncludeを使用して変数を解決できるようにする。
	引数：	variableResolver；
		変数解決を提供することのできるIXmlCompilerIncludeの実装。

【００６７】

【表２－４】

１６．	コマンド：	void AddVariable(XmlQualifiedName name, XmlExpression variable); 説明： この変数付加のメソッドは、外部パラメータを特定の定義にバインドする。ＸＱｕｅｒｙに有効な関数は、ＸＱｕｅｒｙ関数内に含まれないものである。ＸＳＬＴスタイルシート全体が変数とみなされる。ＸＰａｔｈ式全体が変数とみなされる。 引数： name; バインドされる外部パラメータの名前。 variable; 変数を定義するXmlExpression。
１７．	コマンド：	void AddVariable(XmlQualifiedName name, object variable); 説明： 共通言語実行時オブジェクトを変数にバインドする変数付加のメソッド。このオブジェクトは、ＸＳＤ等価物に変換され、その値が使用される。 引数： Name; バインドされる外部パラメータの名前。 Variable; 変数を定義する共通言語実行時オブジェクト。
１８．	コマンド	void AddVariableDeclaration(XmlQualifiedName name, XmlSchemaType type, XmlExpression defaultValue); 説明： 変数をまだ定義せずに変数を宣言する変数宣言を付加するメソッド。定義は、XmlArgumentListを介して実行時に提供される。XmlArgumentListに適切な変数が見つからなかった場合は、提供されているdefaultValueが代わりに使用される。 引数： name; 宣言する変数の完全修飾名。 type; 宣言される変数のXmlSchemaType。 defaultValue; 変数のデフォルト値。

【００６８】

ＩＩＩ．ＸＭＬ式コマンド

XmlExpressionは、所与のＸＭＬ問い合わせのコンパイル済みの書式である。これは、基本的に、ＱＩＬをカプセル化したものである。XmlExpressionは合成可能なオブジェクトである。すなわち、XmlExpressionが作成された後で、それを、別の問い合わせのコンパイル時に再利用して、問い合わせライブラリや仮想ビューなどその他の機能を使用可能にすることができる。

【００６９】

XmlExpression上では少なくとも２つのメソッドを使用することができる。すなわち、値は、それが静的値である場合には、所与のXmlExpressionの値を取り出すのに使用することのできるIXmlValueReaderを返し、型は、そのXmlExpressionのＸＳＤ型を取り出すのに使用することのできるXmlSchemaTypeオブジェクトを返す。

【００７０】

【表３】

１．	コマンド：	IXmlValueReader Value{ get; }
	説明：	そのXmlExpressionの値であるIXmlValueReaderを返す。また、その値が定数である場合には、IXmlValueReaderを実装するオブジェクトも返す。
２．	コマンド：	XmlSchemaType Type{ get; }
	説明：	この特性は、そのXmlExpressionの型をＸＭＬスキーマ（ＸＳＤ）によって表されるものとして表すXmlSchemaTypeを返すことができる。また、その値が定数である場合には、XmlSchemaTypeも返す。

【００７１】

ＩＶ．バックエンドターゲット問い合わせ生成プログラムコマンド

ターゲット問い合わせ生成プログラム（またはターゲット言語コンパイラ）は、特定のデータソースに所与の問い合わせを実行することのできるオブジェクトを生成するのに使用される。ここでは、２つの例示的書式を説明する。XmlILGeneratorエンジンは、RXPathNavigatorまたはXPathEditor実装への問い合わせを実行することができる。これは、XmlCacheなど、RXPathNavigatorを顕在化できる任意のデータソースが、問い合わせ機能を「組み込む」ことができるようにする。SQLGeneratorエンジンは、XmlExpressionを直接処理することができ、したがって、ＳＱＬステートメントを実行のために最適化することができる。

【００７２】

【表４】

Ａ．	ＸＭＬターゲットコマンド	
１．	コマンド：	XmlILGenerator();
	説明：	このメソッドコマンドは新規のXmlILGeneratorを構築する。
２．	コマンド：	XmlCommand Generate(XmlExpression query);
	説明：	このメソッドコマンドは、所与のXmlExpressionのためのXmlCommandを生成する。次いでそのXmlCommandを実行して問い合わせの結果を取り出すことができる。問い合わせの実行可能な書式であるXmlCommandを返す。
	引数：	query; XmlCommandがそこから生成されるXmlExpression。
３．	コマンド：	OnGenerateEvent;
	説明：	生成時にメッセージを受け取るために登録することのできるイベント。生成プログラムは、ユーザがそれに基づいて処置を取ることのできる警告またはその他のメッセージを報告することができる。このイベントは、XMLCommand生成を妨げない。
Ｂ．	ＳＱＬターゲットコマンド	
１．	コマンド：	SqlGenerator();
	説明：	この生成プログラムはＱＩＬを取り込み、データベースに対して問い合わせを行うことができる。このメソッドは新規のSqlGeneratorを構築する。
２．	コマンド：	XmlCommand Generate(XmlExpression query);
	説明：	このメソッドコマンドは、所与のXmlExpressionのためのXmlCommandを生成する。次いでそのXmlCommandを実行して問い合わせの結果を取り出すことができる。問い合わせの実行可能な書式であるXmlCommandを返す。
	引数：	Query; XmlCommandがそこから生成されるXmlExpression。
３．	コマンド：	OnGenerateEvent;
	説明：	生成中にメッセージを受け取るために登録することのできるイベント。生成プログラムは、ユーザがそれに基づいて処置を取ることのできる警告またはその他のメッセージを報告することができる。このイベントはXMLCommand生成を妨げない。

【００７３】

Ｖ．ＸＭＬコマンド

XmlCommandは、問い合わせシステム実行時に結果として生じる出力オブジェクトの１つとすることができる。それは、ユーザまたはその他のプログラムによって実行されうる物理的問い合わせ計画である。XmlCommandは、その問い合わせの実行に使用できるデータソースおよび実行時パラメータの解決と共に供給される。

【００７４】

【表５－１】

１．	コマンド：	OnExecutionEvent;
	説明：	このイベントコマンドを問い合わせ実行時に使用して、結果セットに含まれない情報をユーザに戻すことができる。例えば、いくつかのＷ３Ｃ（登録商標）問い合わせ標準は、処理中の実装依存の挙動を可能にする。実行ランタイムはこの挙動をユーザに報告することができる。
２．	コマンド：	void Execute(RXPathNavigator contextDocument, XmlArgumentList argList, XmlWriter results); void Execute(RXPathNavigator contextDocument, XmlResolver dataSources, XmlArgumentList argList, XmlWriter results);
	説明：	この実行メソッドは、所与のXmlArgumentListを実行時パラメータとして用いて、提供されたRXPathNavigatorへの問い合わせを実行する。結果は提供されたXmlWriterに出力される。
	引数：	contextDocument; 問い合わせ先のデフォルト文書。 argumentList; XmlArgumentListは、その問い合わせの実行に使用できる任意の実行時パラメータを含む。 results; この問い合わせの結果が書き込まれるXmlWriter
３．	コマンド：	void Execute(XPathEditor contextDocument, XmlArcumentList argList, XmlWriter results);
	説明：	この実行メソッドは、提供されたIXmlInforsetEditorへの問い合わせを実行時パラメータとして実行する。結果は、提供されたXmlWriterに出力される。この多重定義は、RXPathNavigatorではなくXPathEditorを利用するので、更新を実行させることができる。
	引数：	defaultDocument; 問い合わせ先のデフォルト文書 argumentList; XmlArgumentListは、その問い合わせの実行に使用できる任意の実行時パラメータを含む。 results; この問い合わせが書き込まれるXmlWriter。
４．	コマンド：	void Execute(XmlResolver dataSources, XmlArgumentList argList, XmlWriter results);
	説明：	この実行メソッドは、XmlResolverを介してデータソースにアクセスし、XmlArgumentListによって提供された実行時パラメータを使用することによって問い合わせを実行する。結果は提供されたXmlWriterに出力される。
	引数：	datasources; 問い合わせの実行に使用できる名前付きデータソースを取り出すのに使用されるXmlResolver。

【００７５】

【表５－２】

		argumentList; XmlArgumentListは、その問い合わせの実行に必要とされる可能性のある任意の実行時パラメータを含む。 results; この問い合わせの結果が書き込まれるXmlWriter。

５．	コマンド：	void Execute(string contextDocumentUri, XmlResolver dataSources, XmlArgumentList argList, XmlWriter results);
		void Execute(string contextDocumentUri, XmlResolver dataSources, XmlArgumentList argList, TextWriter results);
		void Execute(string contextDocumentUri, XmlResolver dataSources, XmlArgumentList argList, Stream results);

	説明：	この実行メソッドは、XmlResolverを介してデータソースにアクセスし、XmlArgumentListによって提供される実行時パラメータを使用することによって問い合わせを実行する。デフォルト文書は、提供された名前と共にXmlResolverにマッピングされる。結果は提供されたXmlWriterに出力される。
	引数：	contextDocumentUri; XmlResolverパラメータを介して解決されるコンテキスト文書のＵＲＩ。 dataSources; 問い合わせの実行に使用できるコンテキストデータソースおよび名前付きデータソースを取り出すのに使用されるXmlResolver。 argumentList; XmlArgumentListは、その問い合わせの実行に必要とされる可能性のある任意の実行時パラメータを含む。 Results; この問い合わせの結果が書き込まれるXmlWriter、TextWriter、またはStream。
６．	コマンド：	XmlReader Execute(RXPathNavigator defaultDocument, XmlArgumentList argList;
	説明：	このメソッドは、提供されたRXPathNavigatorへの問い合わせを所与のXmlArtumentListを実行時パラメータとして用いて実行する。結果は、RXmlReaderによって返される。問い合わせの結果を返すXmlReaderを返す。
	引数：	defaultDocument; リゾルバによって解決されるデフォルト文書名。 argumentList; XmlArgumentListは、その問い合わせの実行に使用できる任意の実行時パラメータを含む。

【００７６】

【表５－３】

７．	コマンド：	XmlReader Execute(XPathEditor defaultDocument, XmlArgumentList argList;
	説明：	このメソッドは、所与のXmlArgumentListを実行時パラメータとして用いて提供されたXPathEditorへの問い合わせを実行する。結果は、RXmlReaderによって返される。この多重定義は、XmlReaderに対してXPathEditorを取り込むので、更新を実行させることができる。問い合わせの結果を返すXmlReaderを返す。
	引数：	defaultDocument; リゾルバによって解決されるデフォルト文書名。 argumentList; XmlArgumentListは、その問い合わせの実行に使用できる任意の実行時パラメータを含む。
８．	コマンド：	XmlReader Execute(XmlResolver dataSources, XmlArgumentList, argList;
	説明：	このメソッドは、XmlResolverを介してdataSourcesにアクセスし、XmlArgumentListによって提供された実行時パラメータを使用することによって、問い合わせを実行する。結果はRXmlReaderによって返される。
	引数：	dataSources; 問い合わせの実行に使用できる名前付きデータソースを取り出すのに使用されるXmlResolver。 argumentList; XmlArgumentListは、その問い合わせの実行に必要とされる可能性のある任意の実行時パラメータを含む。
９．	コマンド：	XmlReader Execute(string contextDocumentUri, XmlResolver, dataSources, XmlArgumentList argList;
	説明：	このメソッドはXmlResolverを介してdataSourcesにアクセスし、XmlArgumentListによって提供された実行時パラメータを使用することによって問い合わせを実行する。デフォルト文書は、提供された名前と共にXmlResolverにマッピングすることができる。結果は、XmlReaderによって返される。
	引数：	contextDocument Uri; リゾルバによって解決されるコンテキスト文書のＵＲＩ。 dataSources; 問い合わせの実行に使用できる名前付きデータソースを取り出すのに使用されるXmlResolver。 argumentList; XmlArgumentListは、その問い合わせの実行に必要とされる可能性のある任意の実行時パラメータを含む。

【００７７】

【表５－４】

１０．	コマンド：	void Execute(XmlResolver, inputSrc, XmlArgumentList argList XmlResolver, outputSrc;
	説明：	このメソッドは、所与のXmlArgumentListを実行時パラメータとして、XmlResolverを所望の入力元として用いて問い合わせを実行する。結果は出力される。
	引数：	argList; XmlArgumentListは、その問い合わせの実行に使用できる任意の実行時パラメータを含む。 inputSrc; 所望の入力でのデータソースを返すXmlResolver。 outputSrc; 所望の出力でのデータソースを返すXmlResolver。

【００７８】

ＶＩ．問い合わせイベントメッセージ

【００７９】

【表６】

１．	コマンド：	string Message { get; };
	説明：	帯域外情報をユーザに伝達することのできる実行ランタイムからのメッセージ。

【００８０】

前述のように、本発明の例示的実施形態を、様々な計算処理装置およびネットワークアーキテクチャと一緒に説明してきたが、その基底をなす概念は、問い合わせシステムおよびアプリケーションプログラミングインターフェイスを実装することが望ましい任意の計算処理装置またはシステムに適用することができる。したがって、本発明の方法およびシステムは、様々なアプリケーションおよび装置に適用することができる。本明細書では、様々な選択を表すものとして例示的プログラミング言語、名称および実例が選択されているが、それらの言語、名称および実例は、限定のためのものではない。本発明によって実現されるものと同じ、類似の、または等価のシステムおよび方法を実現するオブジェクトコードを提供する多数の方法があることを、当分野の技術者は理解するであろう。

【００８１】

本明細書で述べた様々な技法は、ハードウェアまたはソフトウェアと共に、または適切な場合には、両者の組み合わせと共に実装することができる。したがって、本発明の方法および装置、またはそのある種の態様または部分は、フロッピー（登録商標）ディスケット、ＣＤ－ＲＯＭ、ハードドライブまたはその他任意の機械可読記憶媒体などの有形の媒体で実施されたプログラムコード（すなわち命令）の形を取ることができ、そのプログラムコードが、コンピュータなどのマシンにロードされ、それによって実行されると、そのマシンは本発明を実施する装置となる。プログラム可能コンピュータ上でのプログラムコード実行の場合には、その計算処理装置は、一般に、プロセッサ、プロセッサによる読取りが可能な（揮発性および不揮発性メモリおよび／または記憶素子を含む）記憶媒体、少なくとも１つの入力装置、および少なくとも１つの出力装置を含む。本発明の信号処理サービスを、例えば、データ処理ＡＰＩなどの使用によって利用することのできる１つまたは複数のプログラムは、好ましくは、コンピュータとやりとりするために、高水準手続き型言語またはオブジェクト指向プログラミング言語で実装される。ただし、そのまたはそれらのプログラムは、必要に応じて、アセンブリ言語または機械語で実装することもでき

る。いずれにしても、その言語は、コンパイル済みまたは解釈済み言語とすることができ、ハードウェア実装形態と組み合せることができる。

【００８２】

本発明の方法および装置は、電気配線またはケーブル、光ファイバ、その他任意の伝送形態など、何らかの伝送媒体を介して伝送されるプログラムコードの形態で実施された通信を介して実施することもでき、そのプログラムコードが、ＥＰＲＯＭ、ゲートアレイ、プログラム可能論理デバイス（ＰＬＤ）、クライアントコンピュータ、ビデオ記録装置などのマシンに受け取られ、それにロードされ、それによって実行されると、上記の例示的実施形態で述べたような信号処理機能を有する受け取り側マシンは、本発明を実施する装置となる。汎用プロセッサ上で実装されると、このプログラムコードはプロセッサと相まって、本発明の機能を呼び出すように動作する独自の装置を提供する。さらに、本発明と一緒に使用されるどのような記憶技法も、必ず、ハードウェアとソフトウェアの組み合わせとなり得るものである。

【００８３】

以上、本発明を様々な形態の好ましい実施形態との関連で説明してきたが、本発明から逸脱することなく本発明と同じ機能を実施するために、他の類似の実施形態を使用することもでき、あるいは、前述の実施形態に改変および付加を行うこともできることを理解すべきである。さらに、特に無線ネットワーク接続装置の数が急増し続けているため、ハンドヘルド装置オペレーティングシステムやその他のアプリケーション別オペレーティングシステムを含めて、様々なコンピュータプラットフォームが企図されていることを強調しておく必要がある。したがって、本発明は、どのような単一の実施形態にも限定すべきものではなく、添付の特許請求の範囲による幅および範囲で解釈すべきものである。

【図面の簡単な説明】

【００８４】

【図１】本発明の諸態様を実装することのできる計算処理環境の一例を示す構成図である。

【図２】本発明の諸態様を実装することのできるアーキテクチャを示す構成図である。

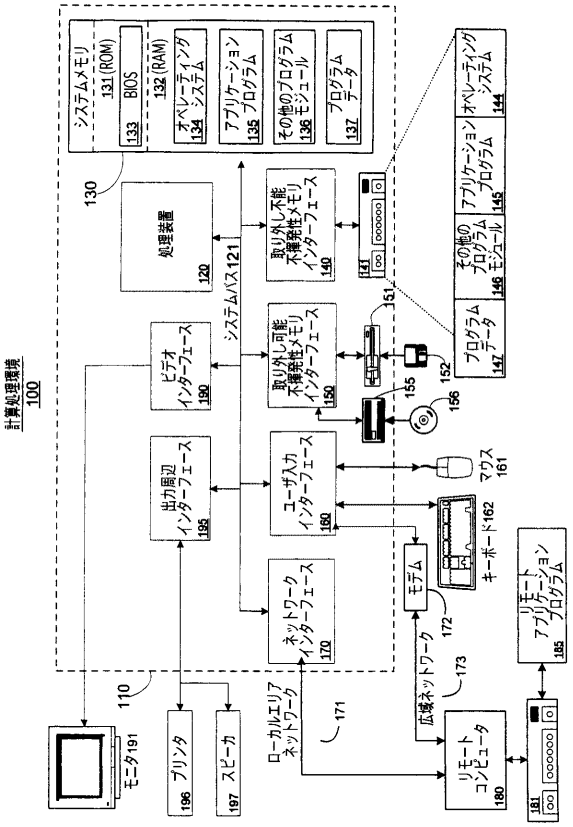
【図３】本発明の諸態様に適用可能な流れを示す流れ図である。

【符号の説明】

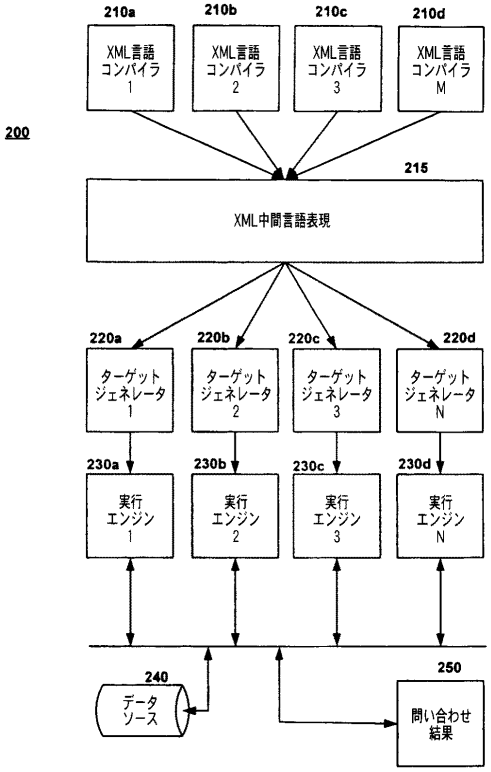
【００８５】

２００　問い合わせ実行時アーキテクチャ
２１０ａ　ＸＭＬ言語コンパイラ１
２１０ｂ　ＸＭＬ言語コンパイラ２
２１０ｃ　ＸＭＬ言語コンパイラ３
２１０ｄ　ＸＭＬ言語コンパイラＭ
２１５　ＸＭＬ中間言語表現
２２０ａ　ターゲットジェネレータ１
２２０ｂ　ターゲットジェネレータ２
２２０ｃ　ターゲットジェネレータ３
２２０ｄ　ターゲットジェネレータＮ
２３０ａ　実行エンジン１
２３０ｂ　実行エンジン２
２３０ｃ　実行エンジン３
２３０ｄ　実行エンジンＮ
２４０　データソース
２５０　問い合わせ結果

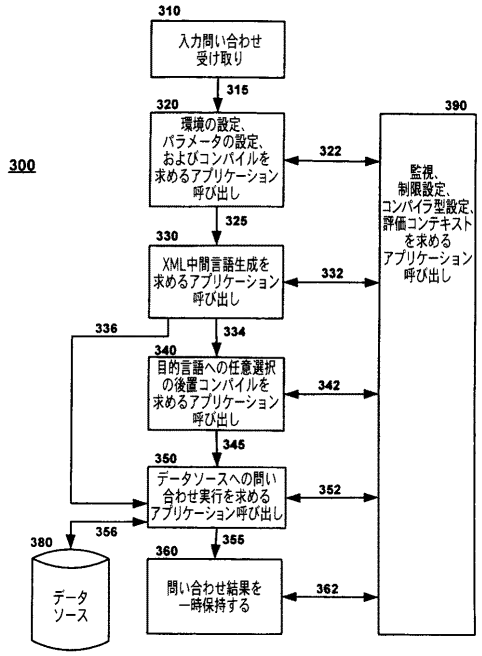
【図１】


計算処理環境
100
システムメモリ
131 (ROM)
133 BIOS
132 (RAM)
オペレーティングシステム
134
アプリケーションプログラム
135
その他のプログラムモジュール
136
プログラムデータ
137
130
処理装置
120
システムバス121
取り外し不能不揮発性メモリインターフェース
140
取り外し可能不揮発性メモリインターフェース
150
ビデオインターフェース
190
出力周辺インターフェース
195
ユーザ入力インターフェース
160
ネットワークインターフェース
170
141
151
152
155
156
オペレーティングシステム
144
アプリケーションプログラム
145
その他のプログラムモジュール
146
プログラムデータ
147
マウス
161
キーボード162
モデム
172
リモートアプリケーションプログラム
185
広域ネットワーク
173
ローカルエリアネットワーク
171
110
モニタ191
プリンタ
196
スピーカ
197
リモートコンピュータ
180
181

【図２】


200
210a
XML言語コンパイラ1
210b
XML言語コンパイラ2
210c
XML言語コンパイラ3
210d
XML言語コンパイラM
215
XML中間言語表現
220a
ターゲットジェネレータ1
220b
ターゲットジェネレータ2
220c
ターゲットジェネレータ3
220d
ターゲットジェネレータN
230a
実行エンジン1
230b
実行エンジン2
230c
実行エンジン3
230d
実行エンジンN
240
データソース
250
問い合わせ結果

【図３】


300
310
入力問い合わせ受け取り
315
320
環境の設定、パラメータの設定、およびコンパイルを求めるアプリケーション呼び出し
322
325
330
XML中間言語生成を求めるアプリケーション呼び出し
332
334
336
340
目的言語への任意選択の後置コンパイルを求めるアプリケーション呼び出し
342
345
350
データソースへの問い合わせ実行を求めるアプリケーション呼び出し
352
355
356
360
問い合わせ結果を一時保持する
362
380
データソース
390
監視、制限設定、コンパイラ型設定、評価コンテキストを求めるアプリケーション呼び出し

フロントページの続き

(72)発明者 マーク ダブリュ．ファッセル
アメリカ合衆国 ９８０７４ ワシントン州 サマミッシュ ノースイースト ３３ コート ２０５１７
(72)発明者 アンドリュー イー．キンボール
アメリカ合衆国 ９８０７４ ワシントン州 サマミッシュ ２３７ アベニュー サウスイースト ５３７
(72)発明者 マイケル エル．ブランデージ
アメリカ合衆国 ９８０３３ ワシントン州 カークランド １２２ アベニュー ノースイースト ８２２２
(72)発明者 セルゲイ デュビネッツ
アメリカ合衆国 ９８００７ ワシントン州 ベルビュー １５１ アベニュー サウスイースト １２２５
(72)発明者 トッド エフ．プフライガー
アメリカ合衆国 ９８０５２ ワシントン州 レッドモンド ノースイースト ５８ コート １８５１０

Ｆターム(参考) 5B081 AA10 CC01
5B082 GA08