(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第6420311号
(P6420311)

(45)発行日　平成30年11月7日(2018.11.7)　(24)登録日　平成30年10月19日(2018.10.19)

(51)Int.Cl.　F I
G06F 12/00 (2006.01)　G06F 12/00 520E
G06N 5/04 (2006.01)　G06N 5/04

請求項の数 30　(全 52 頁)

(21)出願番号　特願2016-502646(P2016-502646)
(86)(22)出願日　平成26年3月14日(2014.3.14)
(65)公表番号　特表2016-521398(P2016-521398A)
(43)公表日　平成28年7月21日(2016.7.21)
(86)国際出願番号　PCT/US2014/027854
(87)国際公開番号　WO2014/143755
(87)国際公開日　平成26年9月18日(2014.9.18)
審査請求日　平成27年11月17日(2015.11.17)
(31)優先権主張番号　61/787,177
(32)優先日　平成25年3月15日(2013.3.15)
(33)優先権主張国　米国(US)

(73)特許権者　515256257
ベウラワークス，エルエルシー.
アメリカ合衆国，４６３８５，インディアナ州，ヴァルパレーゾ，ノース　３６０　ウエスト，８０８番地
(74)代理人　110000811
特許業務法人貴和特許事務所
(72)発明者　グオ，サイト
アメリカ合衆国，４６３２４，インディアナ州，ハモンド，マジソン　アベニュー，７１３２番地

最終頁に続く

(54)【発明の名称】データ取り込みおよび該データへのユーザアクセス促進システムおよび方法

(57)【特許請求の範囲】
【請求項１】
データを取り込み、かつ、該データへのユーザのアクセスを促進させて、パフォーマンスを向上させることができる、システムであって、
少なくとも１つの処理装置により実行され、前記データが、少なくとも１つのデータ特徴および少なくとも１つの使用特徴を含む、１つ以上のデータ属性を備えた１つ以上のオブジェクトとして保存される、ストレージ装置と、
前記少なくとも１つの処理装置により実行され、ユーザが、前記ストレージ装置にデータを入力すること、および、ストレージ装置に保存されたデータにアクセスすることを可能にする、１つ以上の入力メカニズムを提供する、少なくとも２つのユーザインターフェースであって、該少なくとも２つのユーザインターフェースのうちの第１ユーザインターフェースの入力メカニズムが、該少なくとも２つのインターフェースのうちの第２ユーザインターフェースの入力メカニズムよりも高い抽象化レベルで機能し、第１および第２ユーザインターフェースは、異なるアプリケーション開発技術を有するユーザによる使用に適応する、少なくとも２つのユーザインターフェースと、
前記少なくとも１つの処理装置により実行され、前記ストレージ装置および前記少なくとも２つのユーザインターフェースに動作的に接続される、コントローラであって、前記少なくとも２つのユーザインターフェースから前記データを受領し、前記データ属性に基づく前記オブジェクトとして、前記データを、前記ストレージ装置の少なくとも第１のデータベースに第１のストレージフォーマットで保存するとともに、前記オブジェクト間の

関係を追加オブジェクトとして保存して、前記データが前記オブジェクトおよび追加オブジェクトを含むようにし、かつ、必要に応じて、第１のデータベースに保存された前記データを、前記オブジェクトの前記データ属性に基づいて、第１のストレージフォーマットとは異なる、少なくとも第２のストレージフォーマットに変換し、第２のストレージフォーマットに変換された前記データを、前記ストレージ装置の第２のデータベースに保存して、該システム全体のパフォーマンスを向上させる、コントローラと、
を備える、データ取り込みおよび該データへのユーザアクセス促進システム。
【請求項２】
第１データベースはトリプルストアデータベースを備える、請求項１に記載のシステム。
【請求項３】
第２データベースは、関係データベース、カラム型データベース、グラフデータベース、キーバリューデータベース、ドキュメントデータベース、およびファイルストレージデータベースのうち１つ以上を備える、請求項２に記載のシステム。
【請求項４】
前記オブジェクトはそれぞれ、該オブジェクトを一意的に識別するオブジェクトＩＤ、該オブジェクトの現在の状態を示す状態インジケータ、該オブジェクトの訂正の順序に関する現在の訂正の状態を示すリビジョン番号、および、該オブジェクトの特定の訂正がいつ行われたかを示すタイムスタンプを含み、少なくとも前記状態インジケータおよび前記オブジェクト間の関係は、ネットワーク発見技術を使用して、前記オブジェクトのネットワーク表示および前記オブジェクトの前記関係を構築するために使用され、かつ、前記ネットワーク表示は、存在していなかった前記オブジェクト間の関連づけの生成、更新および削除のために使用される、請求項１に記載のシステム。
【請求項５】
前記入力データの少なくとも１つのデータ特徴は、データサイズおよびデータスキーマのうち１つ以上を備える、請求項４に記載のシステム。
【請求項６】
前記データの少なくとも１つのデータ特徴は、前記データがアクセスされるたびに変更されたかどうか、およびデータごとに要求される保持時間の１つ以上を備える、請求項４に記載のシステム。
【請求項７】
前記データの少なくとも１つの使用特徴は、データ書き込み頻度、データ更新頻度、データ読み取り頻度、データ読み取り要求タイプ、およびユーザの同時要求数のうち１つ以上を備える、請求項１に記載のシステム。
【請求項８】
前記コントローラは、第１および第２のユーザインターフェースのユーザに、任意のデータベースストレージフォーマットに依存しない１つのフォーマットで表される共有操作をサポートする単一のインターフェースを提供することにより、該システムのユーザとの通信を促進させる、請求項１に記載のシステム。
【請求項９】
前記少なくとも２つのユーザインターフェースは、少なくとも１つのメジャーデベロッパインターフェース、および、少なくとも１つのマイナーデベロッパインターフェースを備える、請求項１に記載のシステム。
【請求項１０】
前記少なくとも１つのメジャーデベロッパインターフェースは、比較的熟練したユーザがソフトウェアを開発できる統合開発環境を備える第１メジャーデベロッパインターフェースを含む、請求項９に記載のシステム。
【請求項１１】
前記少なくとも１つのメジャーデベロッパインターフェースは、グラフィカルユーザインターフェース（ＧＵＩ）アプリケーションおよびソフトウェアウィジェットの１つ以上

の開発をサポートする、ＧＵＩベースのプラットフォームを備える第２メジャーデベロッパインターフェースを含む、請求項９に記載のシステム。

【請求項１２】

前記マイナーデベロッパインターフェースは、ソフトウェア開発スキルを欠くユーザが、より高次の機能構築ブロックに基づいてアプリケーションを構築できるグラフィカルウェブアプリケーションビルダを備える、請求項９に記載のシステム。

【請求項１３】

前記アプリケーションメタファイルに基づいて実行可能なアプリケーションを生成する、前記コントローラおよび前記マイナーデベロッパインターフェースに動作的に接続されたコードジェネレータをさらに備える、請求項１２に記載のシステム。

【請求項１４】

前記コントローラは、人間に読み取り可能なユーザデータクエリに基づいて、機械に読み取り可能なユーザデータクエリを生成し、人間に読み取り可能な自然言語テキストに基づいた中間コマンドを生成するための自然言語処理コンポーネントをさらに備え、前記中間コマンドは、自動的に生成されるコードを提供するために解析される、請求項１に記載のシステム。

【請求項１５】

前記コントローラは、自然言語の質問を、自動データマイニングコンポーネントにより実行されるテクニカルクエリ言語に変換する、自然言語処理エンジンを使用した前記ストレージ装置から前記データを回収するデータマイニングコンポーネントをさらに備える、請求項１に記載のシステム。

【請求項１６】

システム内にデータを取り込み、かつ、該データへのユーザのアクセスを促進させて、前記システムのパフォーマンスを向上させるための方法であって、

少なくとも１つの処理装置により実行されるコントローラにおいて、前記少なくとも１つの処理装置により実行される、少なくとも２つのユーザインターフェースであって、該少なくとも２つのユーザインターフェースのうちの第１ユーザインターフェースの入力メカニズムが、該少なくとも２つのインターフェースのうちの第２ユーザインターフェースの入力メカニズムよりも高い抽象化レベルで機能し、第１および第２ユーザインターフェースは、異なるアプリケーション開発技術を有するユーザによる使用に適応する、少なくとも２つのユーザインターフェースのうちの、少なくとも第１ユーザインターフェースから受領したデータを、前記コントローラにより、少なくとも１つのデータ特徴および少なくとも１つの使用特徴を含む、１つ以上のデータ属性を備えた１つ以上のオブジェクト、および、該オブジェクト間の関係を備えた追加オブジェクトとして、ストレージ装置の少なくとも第１データベースに、第１ストレージフォーマットで保存し、

前記コントローラにより、必要に応じて、第１データベースに保存された前記データを、前記オブジェクトの前記データ属性に基づいて、第１ストレージフォーマットとは異なる、少なくとも第２ストレージフォーマットに変換し、変換された前記データを、前記ストレージ装置の第２データベースに保存し、該システム全体のパフォーマンスを向上させ、および、

前記コントローラにより、第１データベースおよび第２データベース内の前記データにアクセスするために、第１ユーザインターフェースまたは少なくとも１つの第２ユーザインターフェースから受信したクエリを処理する、

工程からなる、データ取り込みおよび該データへのユーザアクセス促進方法。

【請求項１７】

前記オブジェクト間の前記関係を使用し、かつ、該オブジェクト間のネットワーク発見技術を使用する工程をさらに含み、前記オブジェクト間の関係のソーシャルネットワークによる表示を生成する、請求項１６に記載の方法。

【請求項１８】

クエリパターンおよび結果として生じるデータパターンを監視し、前記クエリパターン

またはデータパターンが、少なくとも１つのあらかじめ定義された変換始動ルールを満たす場合に、前記変換ステップを開始する、請求項１６に記載の方法。
【請求項１９】
第１データベースおよび第２データベースにおける前記入力データを、前記コントローラによって更新する、請求項１６に記載の方法。
【請求項２０】
第１データベースおよび第２データベースの少なくとも１つは、トリプルストアデータベースを備える、請求項１６に記載の方法。
【請求項２１】
第１データベースおよび第２データベースの少なくとも１つは、関係データベース、カラム型データベース、グラフデータベース、キーバリューデータベース、ドキュメントデータベース、およびファイルストレージデータベースのうちの１つ以上を備える、請求項２０に記載の方法。
【請求項２２】
前記オブジェクトはそれぞれ、該オブジェクトを一意的に識別するオブジェクトＩＤ、該オブジェクトの現在の状態を示す状態インジケータ、該オブジェクトの訂正の順序に関する現在の訂正の状態を示すリビジョン番号、および、前記オブジェクトの特定の訂正がいつ行われたかを示すタイムスタンプを含み、少なくとも前記状態インジケータおよび前記オブジェクト間の関係は、ネットワーク発見技術を使用して、前記オブジェクトのネットワーク表示および前記オブジェクトの前記関係を構築するために使用され、かつ、前記ネットワーク表示は、存在していなかった前記オブジェクト間の関連づけの生成、更新および削除のために使用される、請求項１６に記載の方法。
【請求項２３】
前記データの少なくとも１つのデータの特徴は、データサイズおよびデータスキーマのうち１つ以上を備える、請求項１６に記載の方法。
【請求項２４】
前記入力データの少なくとも１つのデータの特徴は、前記データがアクセスされるたびに変更されたかどうか、およびデータごとに要求される保持時間の１つ以上を備える、請求項１６に記載の方法。
【請求項２５】
前記入力データの少なくとも１つの使用特徴は、データ書き込み頻度、データ更新頻度、データ読み取り頻度、データ読み取り要求タイプ、およびユーザの同時要求数のうち１つ以上を備える、請求項１６に記載の方法。
【請求項２６】
前記コントローラより、第１および第２のユーザインターフェースのユーザに、任意のデータベースストレージフォーマットに依存しない１つのフォーマットで表される共有操作をサポートする単一のインターフェースを提供することにより、該システムのユーザとの通信を促進させる、請求項１６に記載の方法。
【請求項２７】
第１ユーザインターフェースは、少なくとも１つのメジャーデベロッパインターフェースを備え、第２ユーザインターフェースは、マイナーデベロッパインターフェースを備え、
前記メジャーデベロッパインターフェースは、比較的熟練したユーザがソフトウェアを開発できる統合開発環境を備え、前記マイナーデベロッパインターフェースは、ソフトウェア開発スキルを欠くユーザが、より高次の機能構築ブロックに基づいてアプリケーションを構築できるグラフィカルウェブアプリケーションビルダを備える、請求項１６に記載の方法。
【請求項２８】
前記コントローラにユーザがデータマイニングリクエストを行うことによりデータマイニングタスクを実行する工程であって、前記リクエストを質問駆動型データマイニングコ

ンポーネントに直接送信する、あるいは、前記リクエストが自然言語により表されている場合には、前記リクエストを、自然言語処理エンジンにより、前記質問駆動型データマイニングコンポーネントにより使用可能な指示へ変換して送信する工程をさらに含む、請求項１６に記載の方法。

【請求項２９】

前記コントローラによって、機械読取可能ユーザデータクエリに基づいて、自然言語の質問を、自動データマイニングコンポーネントにより実行されるテクニカルクエリ言語に変換する、自然言語処理エンジンを使用した前記ストレージ装置から前記データを回収するデータマイニングを行う工程をさらに備える、請求項１６に記載の方法。

【請求項３０】

前記コントローラによって、前記クエリおよび該クエリから生じたデータパターンをモニタリングする工程をさらに備える、請求項１６に記載の方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、主に企業の情報管理、特に、企業の情報管理の一部あるいは補助としてのナリッジ（知識）取込および発見システムに関する。

【背景技術】

【０００２】

法人やその他の組織を含む、さまざまなタイプの事業主体（ここでは集合的に「企業」という）は、通常、関連する顧客に商品および／あるいは役務を提供するといった特定の目標を達成するために、設立され、組織され、および運営されている。さまざまな規模の企業のいずれにおいても、これらの目標を達成するために、無数のプロセスを導入し、そのようなプロセスを実行する際に、関連する相当な量のデータを入手することが必要とされる。企業が徐々に規模を拡大し、および／または、より困難かつ複雑な目標を達成しようと試みるに従って、たいていの場合、それに伴うプロセスを真に理解し、そのプロセスを適切に管理し、そのプロセスを実行するために必要なリソースを管理することが非常に困難となる。すなわち、消費者インサイトなどを探る際に必要となる調査データが相当な量となる一方で、そのようなデータの量の膨大さ、複雑さ、および流動性により、潜在的なリソースを活用することが困難となっている。

【０００３】

この問題を部分的に解消する、さまざまな技術が存在している。たとえば、データを効率的に保存しアクセスできるようにするために、過去４０年間において数多くのデータベース技術が開発されてきているが、いずれにも一長一短がある。さらに、このような技術が存在する場合でも、決定者がデータにアクセスできるようにするためには、特に訓練された技術者、たとえば、ソフトウェア開発および／またはデータベース管理のエキスパートのサポートが必要となる。しかしながら、これにより実質的に費用が発生する一方、データ利用者のニーズが満たされない可能性が十分にある。さらに、保存された調査データから調査レポートを届ける技術が知られているが、消費者インサイトを探り、かつ、データが示すプロセスについて理解することは、いまだ困難な課題である。

【０００４】

したがって、企業データを取り込み、かつ、企業データへのアクセスを容易にして、企業データ管理についての経験が不十分な場合であっても、消費者インサイトを明らかにするという、従前は不可能ではないとしても、非常にコストがかかって困難であったことを可能とするシステムを提供することは有益である。

【関連文献の相互参照】

【０００５】

本願は、本明細書に参照として組み込まれる、２０１３年３月１５日に出願された米国特許仮出願第６１／７８７，１７７号（発明の名称：「企業レベルのアプリケーション開発システム」）に基づく優先権の利益を主張する。

【発明の概要】
【０００６】
本発明は、従来技術によるソリューションの不都合な点を解消する、ナレッジ取込および発見システムに関する。具体的には、該システムは、入力されたすべてのデータがオブジェクトとして保管されるナリッジリポジトリと、オブジェクト間のリレーションシップとからなる。追加的に、前記入力データは、１つ以上の保管フォーマットに従って保管される。該システム内では、少なくとも２つの階層型ユーザインターフェースが、前記入力されたデータを得るための入力メカニズム、前記入力データに関するオブジェクト情報、および前記入力データに関する関係情報を提供することにより、エンドユーザのアプリケーション開発を可能にする。追加的に、コントローラは、前記ナリッジリポジトリおよび前記少なくとも２つの階層型ユーザインターフェースと作動可能に接続され、前記少なくとも２つの階層型ユーザインターフェースから前記入力データ、前記オブジェクト情報、および前記関係情報を受領し、前記オブジェクト情報および前記関係情報に基づいて、前記入力データをオブジェクトとしてナリッジリポジトリに保管する。
【図面の簡単な説明】
【０００７】
【図１】図１は、本発明のさまざまな態様を実行するために用いられる典型的な処理装置を示すブロック図である。
【図２】図２は、本発明を実行するために用いられる、さまざまなネットワーク化されたハードウェアコンポーネントを示すブロック図である。
【図３】図３は、本発明のさまざまな実施態様に基づいたファンクショナルコンポーネントを示すブロック図である。
【図４】図４は、ＲＤＦデータおよびリレーショナルデータに基づくデータ変換処理の典型的な実行例を示すブロック図である。
【発明を実施するための形態】
【０００８】
本発明は、添付の請求の範囲において具体的に定義される。本発明の特徴は、添付の図面とともに、下記の詳細な説明によって明らかとされる。１つ以上の実施態様が例示としてのみ記載される。添付の図面において、同一の参照符号は同一の要素を示す。
【０００９】
図１は、本発明を実行するために使用される典型的な処理装置１００を示す。処理装置１００は、たとえば、下記に詳述するシステム２００の１つ以上のコンポーネントを作動させるために使用される。たとえば、処理装置１００は、ワークステーションコンピュータあるいはサーバコンピュータを備える。あるいは、処理装置１００は、ストレージコンポーネント１０４に連結されたプロセッサ１０２を備える。ストレージコンポーネント１０４は、順番に、保存された実行可能命令１１６およびデータ１１８を備える。
【００１０】
一実施形態では、プロセッサ１０２は、１つ以上のマイクロプロセッサ、マイクロコントローラ、デジタル信号プロセッサ、コプロセッサなど、あるいは、これらの組み合わせを備え、保存された実行可能命令１１６の実行を可能とし、かつ、保存されたデータ１１８を操作する。同様に、ストレージコンポーネント１０４は、ランダムアクセスメモリ（ＲＡＭ）、読取専用メモリ（ＲＯＭ）、その他の非一時的な機械読取可能装置などの、１つ以上の揮発性あるいは不揮発性メモリを備える。さらに、ストレージコンポーネント１０４は、ハードドライブ、光学ディスクドライブ、フロッピーディスクドライブなどのさまざまな形態を採ることができる。
【００１１】
図１に示したタイプのプロセッサおよびストレージの配置は、当業者には周知である。一実施態様において、本明細書に開示されたプロセッシング技術は、１つ以上の処理装置１００のストレージコンポーネント１０４内において、実行可能命令とデータとの組み合わせとして実行される。

【００１２】
図示のように、処理装置１００は、１つ以上のユーザ入力装置１０６、ディスプレイ１０８、周辺インターフェース１１０、その他の出力装置１１２、およびプロセッサ１０２と通信するネットワークインターフェース１１４を備える。プロセッサ１０２は、図示のように、これらの装置／ディスプレイ／インターフェース１０６～１１４と個別かつ直接に接続可能であるが、処理装置１００のコンポーネント同士を意図されたように通信させるメカニズムとして、１つ以上のバスサブシステム（図示せず）を用いることも、当業者には周知である。
【００１３】
ユーザ入力装置１０６は、プロセッサ１０２にユーザが入力するためのメカニズムを備える。たとえば、ユーザ入力装置１０６は、装置１００のユーザがプロセッサ１０２にデータを入力するためのキーボード、マウス、タッチスクリーン、マイクロフォン、適切な音声認識アプリケーション、その他の手段を備えることができる。ディスプレイ１０８は、ＣＲＴディスプレイ、フラットパネルディスプレイ、その他の公知のディスプレイを備える。一実施態様においては、ディスプレイ１０８は、プロセッサ１０２によって実行される、適切に保存された命令１１６と関連して、下記のグラフィカルユーザインターフェースを実行するために用いることができる。グラフィカルユーザインターフェースをこのように実行することは、当業者には周知である。
【００１４】
周辺インターフェース１１０は、本発明に関連して用いられる、磁気ディスクあるいは光学ディスク装置などのメディア装置を含む周辺機器、他の処理装置、あるいは他の入力ソースと通信するために必要とされる、ハードウェア、ファームウェア、および／またはソフトウェアを有する。同様に、その他の出力装置１１２は、同様なメディアドライブメカニズム、他の処理装置、あるいは、スピーカー、ＬＥＤ、プリンタ、ファクシミリ装置、触覚出力装置など、装置１００のユーザに情報を提供するための出力先を任意に備える。
【００１５】
また、ネットワークインターフェース１１４は、プロセッサ１０２が、ワイヤネットワークあるいはワイヤレスネットワーク、ローカルネットワークあるいはワイドエリアネットワーク、プライベートネットワークあるいはパブリックネットワークを経由して、他の装置と通信することを可能にする公知のハードウェア、ファームウェア、および／またはソフトウェアを備える。たとえば、そのようなネットワークには、ワールドワイドウェブ、インターネット、あるいは企業内ネットワークなどの公知のネットワークが含まれる。
【００１６】
装置１００は、本発明を実行するための技術の一例であって、機能的に同等である他の技術も採用可能である。たとえば、既知ではあるが、１つ以上のプロセッサによって履行された実行可能命令により実行される機能の一部または全部を、ファームウェア、および／または、特定用途向け集積回路（ＡＳＩＣ）、プログラム可能な論理アレイ、ステートマシンなどのハードウェア装置を用いて実行することもできる。さらに、装置１００を、図示の実施態様と比較して、より多くのコンポーネントにより、あるいは、より少ないコンポーネントにより構成することもできる。このように、本発明の多くの変形例も使用可能である。さらに、図１には、単一の処理装置１００が示されているが、本発明を実施するにあたって、このような処理装置の組み合わせを、公知のネットワーク技術と組み合わせて作動させるように構成することも可能である。処理装置およびネットワークは常に変化し続けるため、図１に図示した処理装置１００は、当業者に公知である多数の処理装置についての具体的な一例として示されているにすぎない。
【００１７】
図２は、本発明を実施するために用いられる、複数のハードウェアコンポーネントを含むシステム２００を図示する。図示のように、システム２００は、１つ以上のサーバコンピュータからなるコントローラ２０２を備える。コントローラ２０２は、他のさまざまな

コンポーネントと、直接的にあるいは１つ以上のネットワーク２０４を介して、通信することができる。ネットワーク２０４は、ローカルネットワークあるいはワイドエリアネットワーク、プライベートネットワークあるいはパブリックネットワーク、ワイヤネットワークあるいはワイヤレスネットワークからの所望の組み合わせを選択して、構成することができる。上述のように、このようなネットワークには、公知のワールドワイドウェブ、インターネット、あるいは企業内ネットワークが含まれる。

【００１８】

ワークステーション２０６は、デスクトップコンピュータ、ラップトップコンピュータ、あるいはモバイルコンピュータなどの処理装置を備え、ネットワーク２０４を介して、コントローラ２０２と通信可能である。一実施態様においては、ワークステーション２０６は、公知のように、グラフィカルユーザインターフェースを提供可能なウェブブラウザアプリケーションあるいは他のアプリケーションを実行する。このようなアプリケーションを用いて、ワークステーション２０６は、下記に詳述するように、さまざまな階層型ユーザインターフェース２０８のうちの１つをさらに実行可能である。さらに、ワークステーション２０６は、このような階層型ユーザインターフェース２０８に基づいて開発された１つ以上のエンドユーザアプリケーションを受け入れて、実行することも可能である。

【００１９】

図示のように、１つ以上の階層型ユーザインターフェースサーバ２０８は、コントローラ２０２と通信するとともに、ネットワーク２０４を介して、ワークステーション２０６と通信することが可能である。公知であるが、１つ以上の階層型ユーザインターフェースサーバ２０８は、アプリケーションおよびウェブサーバの組み合わせにより構成することができる。この場合、ウェブサーバは、ユーザからの要求により、ウェブサーバと接続しているアプリケーションサーバにより提供されたアプリケーションリソースを用いたアクションを実行する。具体的には、ウェブサーバリレーは、このような要求をアプリケーションサーバに送り、アプリケーションサーバは、特定のアクションを実行し、このアクションの結果をウェブサーバに戻し、さらに、ウェブサーバは、その結果をユーザワークステーション２０６に送る。このようなウェブサーバは、ハードウェアコンポーネントにより構成されるが、本明細書中に記載されたいずれのサーバと同様に、コンピュータシステム上で作動するソフトウェアモジュールによっても構成可能である。

【００２０】

いずれにせよ、このような技術によれば、階層型ユーザインターフェースサーバ２０８は、下記に詳述する、少なくとも１つのメジャーデベロッパインターフェースおよび／またはマイナーデベロッパインターフェースを提供する。たとえば、階層型ユーザインターフェースサーバ２０８は、ワークステーション２０６上に表示されたウェブページなどを実現することにより、１つ以上の階層型ユーザインターフェースを提供する。これらの階層型インターフェースは、一実施形態において、最終的にアプリケーションメタファイルを開発するために使用される。ここで、アプリケーションメタファイルは、実行可能なソースコードを生成するのに十分な、公知かつ下記に詳述するユーザインターフェースマークアップあるいはファンクショナルマークアップなどの情報からなる。エンドユーザアプリケーションサーバ２１２は、ウェブサーバおよびアプリケーションサーバを備え、上述したように、要求するユーザに対して、コード生成サーバ２１０によって生成されたエンドユーザアプリケーションを提供する機能を備える。

【００２１】

図２にさらに図示するように、コントローラ２０２は、一括してデータベースコンプレックス２１９を構成する複数のデータベースサーバ２１４～２１８と通信する。ここで、データベースは、本明細書中に記載されているデータベースストレージフォーマットを含む、公知のデータベースストレージフォーマットを実行する適切なストレージ装置を備える。たとえば、１つ以上の第１データベースサーバ２１４は、第１ストレージフォーマットあるいはスキーマを実行し、１つ以上の第２データベースサーバ２１６は、第２ストレージフォーマットあるいはスキーマを実行し、さらに第Ｎ番目までのデータベースサーバ

を設けることができ、１つ以上の第Ｎ番目のデータベースサーバ２１８は、第Ｎ番目のストレージフォーマットあるいはスキーマを実行するよう設けられる。たとえば、一実施形態では、第１データベースサーバ２１４は、いわゆるトリプルストアデータベースを実装し、第２データベースサーバ２１６は、リレーショナルデータベースを実装し、第Ｎ番目のデータベースサーバ２１８は、カラム型データベース、グラフデータベース、キーバリュー型データベース、ドキュメントデータベース、およびファイルストレージデータベースなどのさらに他のデータベースストレージフォーマットを実行する。さらに他のデータベースストレージフォーマットの使用が可能であることは、当業者には自明であり、したがって、本発明はデータベースストレージフォーマットの種類によっては限定されることはない。

【００２２】

このような構成により、それぞれのデータベースストレージフォーマットを利用することが可能となるが、下記に詳述するように、コントローラ２０２は、エンドユーザがそれぞれのデータベースストレージフォーマットの複雑さを習得することを不要とする、抽象化レイヤーとして機能する。一実施形態では、下記に記載するように、コントローラ２０２は、全体のパフォーマンスを向上させるために、必要に応じて、１つのストレージフォーマットから他のストレージフォーマットへのデータの変換を開始する。別の実施形態では、マルチデータベースストレージフォーマットを備えることにより、ユーザが、データの変換の条件を具体的に定義することが可能となる。たとえば、いわゆるＣＡＰ（一貫性、可用性、分断耐性）定理によれば、分散データベースを用いても、３つの属性である、一貫性（すべてのノードは最新かつ同一の情報を有する）、可用性（可用時間／要求受け入れ）、分断耐性（分断状態を扱う）からたった２つの属性のみを有することができる。この帰結に基づき、ユーザは、これら属性のそれぞれあるいは組み合わせを最適化するために、さまざまなデータベース間におけるデータ変換のための要求を指定することができる。

【００２３】

さらに図示されているように、コントローラ２０２は、ネットワーク２０４を介して、１つ以上の自然言語処理（ＮＬＰ）サーバ２２０および１つ以上のデータマイニングサーバ２２２と通信する。下記にさらに詳述するように、ＮＬＰサーバ２２０は、データベースコンプレックス２１９内におけるデータへのアクセス時だけでなく、エンドユーザアプリケーションの開発時においても、自然言語クエリの使用を促進させる。ＮＬＰサーバ２２０とともに働くデータマイニングサーバ２２２は、原因解明、分類、クラスタリング、結合規則発見、および／または、データベースコンプレックス２１９に保存されたデータに基づいた回帰分析などのさまざまなデータマイニングタスクを実行する。

【００２４】

図３は、システム３００と、システム３００に設けられたさまざまな機能を図示している。図３に示されたそれぞれのコンポーネントは、上述したように、図示した機能を実行する１つ以上の処理装置を用いて実行される。システム３００内では、コントローラ３０２は、図示の例においては、リレーショナルデータベース３０４、カラム型データベース３０６、およびトリプルストアデータベース３０８を含む複数のデータベース３０４～３０８と通信する。公知であるが、それぞれのデータベース３０４～３０８は、通常、やりとりを促進するためのそれぞれのデータベース管理システム（ＤＢＭＳ）を備える。図示のように、コントローラ３０２は、対応するＤＢＭＳによって実行されるアプリケーション・プログラミング・インターフェース（ＡＰＩ）３０４ａ～３０８ａを通じて、これらのデータベース３０４～３０８と通信する。このようなＡＰＩは、メーカー独自のドライバあるいは独自のＲＥＳＴ（レプリゼンテーショナル・ステート・トランスファー）インターフェースによって実施される。

【００２５】

一実施態様においては、システム２００および３００によって扱われるすべてのデータは、オブジェクトとして扱われる。したがって、それぞれのデータには、オブジェクトを

一意的に識別するオブジェクトＩＤ、オブジェクトの現在の状況を示す状態インジケータ、オブジェクトの訂正の順序に関する訂正の現在の状態を示すリビジョン番号、および、ある特定の訂正がいつ行われたかを示すタイムスタンプが付される。オブジェクトは、システム内において決して物理的に削除されることはない。ユーザによってオブジェクトが修正あるいは「削除」された場合には、システムはオブジェクトの訂正を行い、この訂正をオブジェクトの現在の状況に反映させる。従前の訂正は履歴として保存される。オブジェクトの例として、いくつかのネームバリューペアに基づいて記述された、周知のジェイソン（ＪＳＯＮ）フォーマットを使用した、グラフィカルユーザインターフェースで見られるタイプの送信ボタンを表１に示す。

【００２６】

【表１】

```
{
        “id”: “jk234hjk34h2i3o4u89ghkjnhk”,
        “objectType”: “widget” ,
        “widgetType”: “button”,
        “title”: “submit”,
        “history”: {
                “rev”: “12”,
                “state”: “active”,
                “timestamp”:”1394654029”
        },
        “widgetProperties”: {
                “width”: “20px” ,
                “height”: “15px”,
                “x”: “100px”,
                “y”: “150px” ,
                “float”: “left”
        },
        “behavior”: [
                {
                        “event”: “single click”,
                        “action”: “asdfjk314j2hjwdflj234”
                }
        ]
}
```

Table 1.

【００２７】

本例では、オブジェクトは「ウィジェット」タイプであり、より具体的には、「送信(submit)」のタイトルが付されたウィジェットの「ボタン(button)」タイプである。このオブジェクトは、現在「アクティブ(active)」であり、訂正１２回目のものである。このオブジェクトは、さらに、「シングルクリック(single click)」の場合にどのような「アクション(action)」を採るかという行動定義を含む。公知であるが、ＪＳＯＮは、人間にとって理解可能であるだけでなく、機械による解析も可能である。当業者には自明であるが、多種多様なオブジェクトタイプおよびサブタイプが、いかなるデータをも実質的にオブジェクトとして取り扱うために、使用可能である。たとえば、システム２００および３００に備えられた自然言語クエリは、一続きの「ワード」オブジェクトとしてみなされ、クエリ自体は、そのような「ワード」オブジェクトの集合体からなるオブジェクトとして扱われる。別の例においては、ソフトウェアソースコードのセグメントは、いくつかの「ステートメント」、「オペレータ」、「変数」、「変数名」などのオブジェクトからなる第１オブジェクトとして扱われる。

【００２８】

システム内のすべてのデータをオブジェクトとして扱うことの利点は、本明細書中にお

いては、オブジェクト間の関係についてステートメントが作成される「トリプル」データ表現コンセプトに対応していることである。たとえば、いわゆるリソースデータフレームワーク（ＲＤＦ）スペシフィケーションは、ウェブリソースなどの「リソース」に関するステートメントを作成するために、主語－述語－目的語の表現（トリプル）を確立する。なお、本明細書においては、コンセプトは、オブジェクトに容易に適用可能であるものを意味する。簡単な例として、上述の例を元にした、トリプルに従って記述可能であるウェブホームで使用されるボタンウィジェットの例を表２に示す。

【００２９】

【表２】

x:button	y:is_in	z:form	c:91fbc220-aacd-11e3-a5e2-0800200c9a66

Table 2.

【００３０】

この例では、主語「ボタン(button)」は、目的語「フォーム(form)」に、関係を表す述語「is_in」によって、関係づけられる。公知であるが、ＲＤＦにおいては、接頭辞ｘ、ｙ、ｚは、本例では「button」、「form」、「is_in」からなる、一意的なネーミングエンティティの情報を提供するユニフォーム・リソース・アイデンティファイア（ＵＲＩ）の一般的な簡略表記である。好ましい実施形態においては、このトリプルフォームは、コンテクストに追加的な（接頭辞ｃを有する）フィールドを提供する、いわゆる「ｎｑｕａｄ」フォーマットに拡張される。したがって、表２の例では、このコンテクストフィールドは、オブジェクトデータ同士をリンクするユニバーサル・ユニーク・アイデンティファイア（ＵＵＩＤ）値を有するように使用される。すなわち、本実施形態においては、コンテクスト・クワド・フィールドは、さまざまなデータ同士を単一のオブジェクトにおいてつなぎ、実際には、多数のトリプル／クワド値を含み得る。下記に詳述するように、ＲＤＦのような規格は、情報を体系づけることによって知識表現を提供するために用いられる構造的なフレームワークを記述した情報などのオントロジー情報を伝達するステートメントも提供する。なお、オントロジー情報は、１つのストレージフォーマットから他のストレージフォーマットへのデータの変換をアシストするために使用される。

【００３１】

一実施態様においては、コントローラ３０２を介して、すべてのデータが、データベース３０４～３０８に追加され、データベース３０４～３０８において変更され、データベース３０４～３０８から読み取られあるいは削除される。上述したように、これにより、すべてのデータベース固有プロトコルを終了させて、コントローラ３０２のユーザには、単一のインターフェースのみ示される。具体的には、前記単一のインターフェースは、いかなるデータベースストレージフォーマットにも依存しない１つのフォーマットで表される共有操作をサポートする。たとえば、コントローラ３０２は、ＪＳＯＮベースのシークェル（ＳＱＬ）のようなＡＰＩを使用するデータを管理するためにエンドユーザに統一されたＡＰＩを提供する。ＳＱＬのようなＡＰＩは、システム３００の外部および内部ユーザとの通信を促進する。これにより、特に、厳格な関係データベース要求を、比較的自由かつ柔軟なＮｏＳＱＬデータベース要求に橋渡しができるので、従来からの開発者が、高度な学習をすることなく、ＮｏＳＱＬデータベースあるいはマルチデータベースの利点を享受することを可能にしている。完全を期するために、場合によっては、エンドユーザに、ＳＱＬのような統一されたＡＰＩに加えて、それぞれのデータベース３０４～３０８のＤＢＭＳへのアクセスを提供することが望ましい。ただし、基礎となるＡＰＩデータベースへのそのようなアクセスは、ＡＰＩに特有の知識を有しないエンドユーザにとっては好ましくない。いずれにせよ、この実施形態においては、ＳＱＬのような統一されたＡＰＩ手段は、すべてのデータベース管理システムにより一般的には提供される、生成、読み取り、更新、および削除といったクラッド（ＣＲＵＤ）操作を含む。このような生成、読み取り、更新、削除の操作についてのＪＳＯＮの例を、表３～表６にそれぞれ示す。

【００３２】

【表３】

```
{
        “collection”:”VideoRental”,
        “data”:{
                {“name”:”Customer”, “CustomerFirstName”:”Paul”, “CustomerId”: “9001”,},
                {“name”:”Rented”, “RentalDate”:”09/28/01”},
                {“name”:”Video”, “VideoId”:”14564”}
        }
}
```

Table 3. – JSON create

【００３３】
【表４】

```
{
        “collection”:”VideoRental”,
        “select”: “CustomerFirstName”,
        “where”:{
                “relation”: {“name”: “Rented”},
                “object”: {“VideoId”:”14564”}
        }
}
```

Table 4. – JSON read

【００３４】
【表５】

```
{
        “collection”:”VideoRental”,
        “update”: “CustomerFirstName”,
        “where”:{
                “relation”: {“name”: “Rented”},
                “object”: {“VideoId”:”14564”}
        }
        “value”:”Jane”
}
```

Table 5. – JSON update

【００３５】
【表６】

```
{
        “collection”:”VideoRental”,
        “where”:{
                “relation”: {“name”: “Rented”},
                “object”: {“VideoId”:”14564”}
        }
}
```

Table 6. – JSON delete

【００３６】
表３～表６が、ＳＱＬのような統一されたＡＰＩの例であるということは、当業者には自明であり、かつ、同様のＳＱＬのような統一されたＡＰＩは、ＸＭＬ（エクステンシブル・マークアップ・ランゲージ）のような他のフォーマットにおいて実行可能であることも自明である。そのような操作要求に基づき、コントローラ３０２は、上記の例においては、ＪＳＯＮ要求を必要なデータベース固有のクエリフォーマットに変換する。たとえば、上述の操作を元に、ユーザは、表４に示すように、読み取り要求をコントローラ３０２へ送る。トリプルストアデータベース３０８に対するクエリを実行する際に、コントローラ３０２は、表７に示したタイプのスパークル（ＳＰＡＲＱＬ）クエリを生成する。
【００３７】
【表７】

```
SELECT ?x
FROM VideoRental
WHERE
{ ?x ?y ?z
    WHERE
    {
       ?y name Rented.
       ?z has property ?h
            WHERE
            {
              ?h name VideoId.
              ?h value 14564.
            }
    }
}
```

Table 7.

【００３８】
本例においては、マッピングルールは、“collection”：“X” => FROM X; “select”：“X” => SELECT ?x; “relation”：{...} => WHERE {?x ?y ?z WHERE {?y...}}などである。このタイプのさらなるマッピングは、当業者であれば容易に導き出すことが可能である。
【００３９】
データ（上述のように、あるオブジェクトに関するデータ）が追加されると、まず、コントローラ３０２は、上述のトリプルの形式にデータを追加する。すなわち、データは、トリプルストアデータベース３０８において生成され、このようなデータに対するクエリは、少なくとも最初にトリプルストアデータベース３０８に適用される。一実施態様においては、トリプルストアデータベース３０８は、トリプルに第４の要素が追加された、いわゆるｎｑｕａｄフォーマットにされる。この場合、第４の要素は、上述のように、オブジェクトアイデンティファイアである。
【００４０】
ユーザがデータに問い合わせると、クエリパーサ、あるいはコントローラ３０２において実行される監視（monitory）は、クエリを監視し、データパターンを生ずる。このようなクエリパーサは、たとえば、ゾーホー社（Zoho CorporationPvt. Ltd.）のアプリケーションズマネージャ(http://www.manageengine.com/products/applications_manager/database-query-monitoring.htmlにおいて入手可能）にあるように、公知である。たとえば、すべてのクエリは、どのオブジェクトがアクセスされているか、データが書き込まれているのか読み取られているのか、当該データのサイズ、クエリの頻度（ロギングデータからの推定）、あるいはどの特定のタイプのレポート／ＳＥＬＥＣＴステートメントが実行されているかなどの、特定のキー・パフォーマンス・インジケータに関して監視される。結果として、前記クエリパーサは、既存のクエリパターンを、あらかじめ定義済みのデー

タ媒体変換トリガ規則にマッチさせることが可能である。その例を下記に挙げる。これらの規則は、データパターンが与えられた規則の条件を満たすと、１つのストレージフォーマットから他のストレージフォーマットへのデータ変換の必要性が、部分的あるいは全体的に、検出される。すなわち、あらかじめ定義された変換規則により、コントローラ３０２は、ある特定のデータを変換できるかについて決定し、変換できるのであれば、コントローラ３０２は、オリジナルのデータ（たとえば、第１データストレージフォーマットに保存されたデータ）を順に処理する変換プロセスを開始し、目的とするフォーマット、たとえば第２データストレージフォーマットで、新たなデータを作成する。同時に、オリジナルのデータはそのまま残されるため、ユーザは、前記変換処理の間でも、データに対して問合せを行うことは可能である。データが変換されると、前記クエリパーサは変換プロセスを通知され、前記クエリパーサは、この部分のデータに対する将来のクエリを解析する方法を変更することが可能となる。たとえば、一実施態様においては、前記クエリパーサは、前記ＳＱＬのような統一されたＡＰＩオペレーションを、特定の基礎的なデータベースのＡＰＩに置き換えて、将来のクエリが適切に扱われ、正しい回答が戻るようにする。

【００４１】

与えられた部分のデータにとって最適なデータベースストレージフォーマットが不明である状況が生じる場合がある。そのような場合には、利用可能なそれぞれのデータベースストレージフォーマットにオブジェクトを変換させ、模擬ロードテストを行うことが望ましい。このようなロードテストは、収集したログデータに基づいて、現実世界のユーザのアクションをまねることが可能である。そのようなロードテストを行う際には、さまざまなアクティビティのパフォーマンスが監視され、望ましい基準に対してどのデータベースストレージフォーマットが最も良好なパフォーマンスを示したかにということに基づいて、最適なデータベースストレージフォーマットが選択される。たとえば、もしその結果が顕著なパフォーマンスの向上を示した場合には、追加的な規則が作成され、データクエリにより関連するタイプのデータが関連付けられるようになる。代替的な実施形態においては、そのような新たな規則を推断するために、公知の機械学習技術が用いられる。たとえば、機械学習アルゴリズムは、既知の規則を用いて統計上のモデルを整えることができ、さらに、未知の規則を新たに推断することが可能である。このようにして、未知のデータに対するパフォーマンステスト（時間のかかるプロセスである）を省くことができ、代わりに、未知のデータは即座に推断された規則に基づいて直接変換される。その後は、必要に応じて、かつ、利用可能なリソースの存在を前提として、より正確な模擬ロードテストによって、推断された規則をさらに確証することが可能である。

【００４２】

上述したように、コントローラ３０２がいつデータ媒体交換を開始するかを決定するために、規則が利用される。一実施態様においては、そのような規則を確立するためにさまざまな要因が、どの要因が一般的にデータ要因あるいはデータ特徴、および使用要因あるいは使用特徴として分類されるかといった観点から、検討される。データ特徴は、最適なデータベースストレージフォーマットの決定に影響する基礎的なデータの特定の属性に関し、データサイズ、要求されるデータの鮮度、あるいは要求されるデータの保存などを含む。使用特徴は、データがどのように使用されたかということの属性に関し、データ書き込み頻度、データ更新頻度、データ読み取り頻度、データ読み取り要求タイプ、およびユーザの同時実行性などを含む。

【００４３】

データ特徴に関しては、データは、比較的短く、数バイトという簡単なテキストであったり、メガバイトのグラフィックであったり、あるいは、ギガバイトという動画であったりする。公知であるが、それぞれのグラフィックのサイズにより、保存のためにどのタイプのデータベースが最適であるかが決定される。その他の関連するデータ特徴は、データに要求される「新鮮さ」である。たとえば、公知であるが、データベース３０４～３０８のそれぞれは、何らかの形でデータのキャッシュを実行する。レポートデータの一時的な

キャッシュは、大幅なデータの改善を可能にするが、これは、データがアクセスされるごとに、レポート中のデータが変化しない場合に限り実行可能なオプションである。さらなるその他の関連するデータ特徴は、要求されるデータの保存である。データは、通常、所定期間ごとにのみ直接的に使用される。たとえば、秒刻みの生産ラインデータは、通常、その後の数週間あるいは数ヶ月間にわたって直接的に有用になることはない。このように、どのデータを、高価であるが高速のデータベースストレージメカニズムから、より遅いが低コストのストレージメカニズムに整理するかについて最適に選択をすることが望まれる。

【００４４】

使用特徴としては、データ読み取り、書き込み、および／または更新の頻度が採用される。たとえば、データには、年次報告書の作成に伴うデータのように、年に一度しか書き込まれないタイプのデータもあれば、生産ラインなどにおいて１秒間に複数回書き込まれるタイプのデータもある。また、一度書き込まれると二度と変更されないデータもあれば、頻繁に変更されるデータもある。もし低頻度のデータが複数のエリアで複製されると、これらのデータをつないで一続きにするために、その更新にかかる時間が次第に長くなる。さらに、多数のシステムにおいて、データ読み取りとデータ書き込みとの間にトレードオフが生ずる。すなわち、１つの操作が他の操作よりもリソースを消費する。さらに、公知であるが、データ読み取りが高い頻度で行われる場合であっても、与えられたレポートが同一のインデックス基準を用いているかによって大きな差が生じる。たとえば、競争的なトーナメントの高得点のリストを見る場合には、毎秒読み取りが行われる。しかしながら、トーナメントの高得点から特定の区分の高得点への変化は決して変わらないか、非常にまれである。レポートのシナリオについては、ユーザの同時性が最適なストレージフォーマットの決定について多大な影響を与える。たとえば、１人のユーザが、レポートを実行した後、レポートをキャッシュすると、メモリに常駐することになり、多大なパフォーマンスの向上は見込まれない。しかしながら、もし１００人が同一のレポートを毎秒要求したとすると、基本となるデータのキャッシュは多大なパフォーマンスの向上をもたらすことになる。

【００４５】

これらの特徴に基づいて、さまざまな規則が開発される。データに基づいたパフォーマンスは、データベース間での変換あるいは同一のデータベース内におけるデータ管理によって向上することが可能である。たとえば、もしデータの書き込み（更新）が頻繁に行われれば、いわゆるビッグデータワイド・カラムデータベースを使用するのが好ましい。この場合、カラムベースのデータに対するクエリは監視される。もし、非インデックスカラムに繰り返しクエリが実行されると、第２のインデックスが作成される必要が生じる。あるいは、もし、一定の時間が経過後、クエリが特定のインデックスを使用しないのであれば、そのインデックスは削除されうる。

【００４６】

別の例においては、基礎的なデータモデルがキーバリューペアのセットに基づいている場合には、ドキュメントストレージエンジンが使用されるべきである。たとえば、アレイの中のアレイに見えるデータ構造を検索するために規則が作成される。関連して、写真や動画など特定のバイナリデータは、ファイルベースのデータシステムに保存されることが最適である。キーバリューストアを使用したシナリオと同様に、コントローラ３０２は、別のインターフェースに保存された関係データにもリンクされたネイティブバイナリーデータインターフェースを見えるようにすることもできる。たとえば、動画についてオブジェクトタイプがある。上述したオブジェクト例にあるように、そのような動画のそれぞれは、ファイルベースのデータベースに保存されたバイナリーオブジェクトファイルにリンクした独自のキー識別子を有するが、他のメタデータは関係データベースに保存される。

【００４７】

データがいわゆるＡＣＩＤ（不可分性、一貫性、独立性、永続性）プロパティを厳格に要求する場合には、制約付き関係データベースが最も適する。しかし、このシナリオの場

合においても、ベストフィットを決定するために特定のトレードオフを分析するべきである。たとえば、高い同時性および膨大な量のトランザクションにより、銀行のＡＴＭからのデータは、ＡＣＩＤの代わりに、ワイドカラムデータベースを用いてより良好に実行されるＢＡＳＥ（基本的な可用性、ソフトステート、結果整合性）モデルに基づいている。

【００４８】

基礎となるデータモデルが、ネットワーク、グラフ、オブジェクト間の接続などを記述するデータについては、グラフデータベースへの保存が最適である。この場合、非常に時間のかかる関係データベースにおける多元接続操作を含む、外部キー関係などの多くの関係を意味するクエリパターンを検索するために規則が確立される。

【００４９】

さらに別の例においては、たとえば、得られたレポートクエリに高い反復性がある場合には、基礎となるデータベースストレージフォーマットに関わらず、キャッシュを使用することが有益である。公知であるが、キャッシュの規則は、キャッシュにおけるデータがどのくらいの頻度で変更されているかを判断し、キャッシュ無効は時間ベースで行われるか、および／または、ソースデータに変化が生じた際に無効とする能力を有する。この場合、キャッシュされたデータは、それ自身の別のオブジェクトとして保存される。たとえば、キャッシュオブジェクトのソースデータは、ワイドカラムデータベースストレージフォーマットに存在するが、実際にキャッシュされたデータは、変換後、キャッシュメモリ内のキーバリューストレージフォーマットに保存される。

【００５０】

上述したように、すべてのデータは、トリプルストアデータベース３０８に最初に保存され、コントローラ３０２は、トリプルストアフォーマットから別のフォーマットへの変換がいつ必要になるか、あるいはその逆も変換がいつ必要になるかを判断する。一実施態様においては、実際に第１データベースストレージフォーマットから第２データベースストレージフォーマットへデータを変換するプロセスは、少なくとも最初はトリプルストアデータフォーマットに、すべてのデータが保存されるためのトリプルストアデータベース３０８の機能性に含まれる。したがって、別のデータベースストレージフォーマットからトリプルストアデータベースストレージフォーマットへの変換だけでなく、トリプルストアデータベースストレージフォーマットから別のデータベースストレージフォーマットへの変換も必要とされる。必然的に、与えられた変換に利用される特定の技術は、第１（ソース）データベースストレージフォーマットおよび第２（ターゲット）データベースストレージフォーマットの性質に依存する。

【００５１】

一般的には、トリプルストアデータベースストレージフォーマットへの変換は、ソースデータベースストレージフォーマットにおける最も基本的かつ根本的なデータ構造を識別すること、およびこれらのデータ構造をトリプルにマッピングすることに基づいて行われる。たとえば、キーバリューストレージフォーマットをトリプルストアストレージフォーマットに変換する場合は、ＲＤＦなどに基づいた変換プロセスは、それぞれのキーバリューを通じて繰り返し行われ、対応するトリプルを作成する。ワイドカラムストレージフォーマットからトリプルストアストレージフォーマットへ変換する場合は、変換プロセスはそれぞれのキースペース、カラムファミリー、カラムおよびロー形成トリプルを通じて繰り返し行われる。ドキュメントストレージフォーマットからトリプルストアストレージフォーマットへ変換する場合は、変換プロセスはそれぞれのコレクション、ドキュメントおよびキーバリュー形成トリプルと通じて繰り返し行われる。グラフデータベースストレージフォーマットから変換する場合は、変換プロセスは、ノード間における次の接続と形成トリプルによって、データ内のすべてのノードを通じて繰り返し行われる。関係データベースストレージフォーマットから変換する場合は、変換プロセスは、初めはそれぞれのテーブルを通じて繰り返し行われ、たとえば、述部が「is a table of」に固定されたトリプルを確立する。また、外部のキー関係その他のインデックスあるいはプロパティは、いずれもそれぞれのテーブルにおいて識別され、トリプルなどのフォームに含めまれる。（

たとえば、”x:table1.column1 y:is_foreign_key_toz:table2.column2”）。それぞれのテーブル内においては、変換プロセスはそれぞれのカラムを通じても繰り返し行われる。それぞれのカラムは、最初に、カラム名であり、かつ、与えられたセル内に含まれた実際のデータバリューである、トリプルサブジェクトを伴った「is a column of」の固定されたトリプルの述部に基づいて、トリプルフォーマットにおいて定義される。同様にして、変換プロセスは、それぞれセルを伴い、それぞれトリプルになる、それぞれのローを通じて繰り返し行われる。

【００５２】

同様に、トリプルストアデータベースストレージフォーマットから別のデータベースストレージフォーマットへの変換は、基本的にトリプルに基づく。ここで、上述したように、トリプルストアデータベースストレージフォーマットは、ｎｑｕａｄフォームであるため、オブジェクト識別を備える第４の要素を含み、このオブジェクト識別は、トリプルデータのコンテクストの変換を確立するために利用される。したがって、トリプルストアストレージフォーマットからキーバリューストレージフォーマットへ変換する場合は、それぞれのトリプルはキーバリューに変換される。トリプルストアストレージフォーマットからワイドカラムストレージフォーマットへ変換する場合には、変換プロセスは、まず、トリプルデータにおけるすべての明確な述部を識別し、それぞれについてカラムファミリーを作成する。その後、変換プロセスは、それぞれのトリプルを通じて繰り返し行われ、それぞれについてローを形成する。コントローラ３０２におけるクエリパーサなどによりもたらされる従前のクエリ情報に基づいて、変換されているデータへのインデックススキームが、データ従前の使用に基づいて得られる。そのようなインデックススキームを得る技術は、たとえば「Oracle Database Performance Tuning Guide (11g Release 1(11.1):Automatic SQL Tuning（http://docs.oracle.com/cd/B28359_01/server.111/b28274/sql_tune.htm#PFGRF028にて入手可能）」で教示されたように、公知である。その後、第２のインデックスは、必要に応じて、得られたインデックススキームに基づいて作成される。トリプルストアストレージフォーマットからドキュメントストレージフォーマットへ変換する場合は、トリプルデータにおいて変換されているすべてのトリプルは、まず、ドキュメントに対応する述部（たとえば、「is_contained_in」）を特定するために分析される。その後、変換プロセスはそれぞれのトリプルを通じて繰り返し行われ、それぞれのトリプルに基づいてキーバリューエントリーを作成する。キーバリューエントリーはその後、対応するドキュメントにリンクされる。トリプルストアストレージフォーマットをグラフストレージフォーマットへ変換する場合は、変換プロセスはトリプルを通じて繰り返し行うことができ、頂点と辺を構築する。

【００５３】

上述したコントローラにより開始される変換のほかにも、相当量のデータが、既存のＲＤＦデータベースに保存されている。これら既存のデータベースを利用するためには、そのような既存のＲＤＦデータを関係データに変換する能力が、トリプルストアデータベース３０８に与えられる。説明のために、トリプルデータはＲＤＦフォーマットに基づくことを前提とするが、他のトリプルフォーマットも使用することは可能である。特に、外部ＲＤＦデータの変換は、２つのデフォルトカラム（１から始まる連続整数を備える、テーブルの主キーとして機能する、識別カラムと、一般的にはＲＤＦ用語のよるリソース名を示すストリングを含むリソース名カラム）を有するテーブルの作成から開始される。この基本的なテーブルから、トリプルデータ内のほぼすべてのプロパティ（述部）が特定され、テーブル内においてカラムに変換される。いくつかのプロパティ（ここではメタプロパティと称す）は、意味データ自体ではなくデータの基本的なオントロジー構造に関する情報を提供するため、すべてのＲＤＦプロパティがこのように利用されるわけではない。オントロジー情報は、変換されるトリプルデータの関係データベース表現をさらに発展させるために利用される。テーブルを拡大するためにＲＤＦプロパティを利用することについては、簡単な例によってさらに説明する。

【００５４】

表８は、複数のＲＤＦステートメントを示す。

【００５５】

【表８】

<lord of the rings> <subject> <middle earth story>.
<lord of the rings> <author> <J. R. R. Tolkien>.
<lord of the rings> <pages> <4709>.
<a song of ice and fire> <subject><seven kingdoms>.
<a song of ice and fire> <author><George R.R. Martin>.
<a song of ice and fire> <pages><4674>.

Table 8.

【００５６】

追加的なテーブルカラムを特定するためのプロパティの使用に関して上述した変換原理に従って、表８におけるＲＤＦステートメントは、表９に示す関係表示に変換することができる。

【００５７】

【表９】

id	resourceName	subject	author	pages
1	lord of the rings	middle earth story	J.R.R.Tolkien	4709
2	a song of ice and fire	seven kingdoms	George R.R. Martin	4674

Table 9.

【００５８】

本例が示すように、ＲＤＦの関係データへの変換は、データ構造あるいはメタデータの変換であり、データ自体の変換ではない。変換プロセスをさらに発展させるために、ＲＤＦメタプロパティにおいて発見されたメタプロパティを利用することは有益である。

【００５９】

ＲＤＦと関係ストレージフォーマットとは、どちらもクラス（型）およびインスタンス（実体）の観点に依存するという点で、データと類似した観点を有している。一方では、ＲＤＦにおいては、クラスおよびインスタンスは明確に定義され、rdf:class、rdf:type、rdfs:domain、rdfs:rangeなど、指定のメタプロパティにサポートされている。その一方で、関係フォーマットにおいては、クラス／インスタンスの観点は、明確に定義されていないものの、「テーブルおよびタプル」と呼ばれる他のフォームにおいて効果的に実行される。テーブルは、クラスとみなされる一方、カラムは、クラスプロパティおよびタプル（ロー／レコード）としてインスタンスとしてみなされる。したがって、一実施態様においては、ＲＤＦ形式のデータを関係形式のデータに変換するためのアプローチは、ＲＤＦクラスを関係テーブルに、ＲＤＦインスタンスを関係タプルに変換することにより行われる。このため、ＲＤＦにおけるそれぞれのリソースのクラスを決定することは必須となる。このタスクは、ＲＤＦにおける利用可能なメタプロパティの利用により容易化される。

【００６０】

したがって、外部ＲＤＦデータが提示されると、変換プロセス（変換プロセスの例は、図４に関連して下記に詳述する）は、分類を示すメタプロパティの発生を特定するために、まずはリソースをスキャンすることにより、その中のリソースの分類を試みる。これら

公知のメタプロパティについては、下記に個別に記載する。

【００６１】

第１のＲＤＦメタプロパティは、rdf:typeである。正式には以下のように定義される。

「rdf:typeは、リソースはクラスのインスタンスであると述べるために使用される
rdf:Propertyのインスタンスである。
フォームはトリプルである：
R rdf:type Cは、ＣがRdfs:Classのインスタンスであり、ＲはＣのインスタンスであることを述べている。」

【００６２】

したがって、変換プロセスが与えられたリソースについてこのメタプロパティを発見すると、そのリソースのクラスを明確に把握する。

【００６３】

第２のＲＤＦメタプロパティは、rdfs:domainである。正式には以下のように定義される。

「rdfs:domainは、与えられたプロパティを有するすべてのリソースは、１つ以上のクラスのインスタンスであることを述べるために使用されるrdf:Propertyのインスタンスである。
フォームはトリプルである：
P rdfs:domain Cは、ＰはクラスRdf:Propertyのインスタンスであり、ＣはクラスRdfs:Classのインスタンスであり、述部がＰであるトリプルのサブジェクトによって表示されたリソースはクラスＣのインスタンスであることを述べている。
プロパティＰが１つ以上のrdfs:domainプロパティを有する場合には、述部がＯであるトリプルのサブジェクトによって表示されたリソースは、rdfs:domainプロパティによって述べられたすべてのクラスのインスタンスである。」

【００６４】

換言すれば、このメタデータは、rdfs:domainトリプルのサブジェクトが、オブジェクトのプロパティであることを示すとともに、その述部としてプロパティを有するその他のトリプルのサブジェクトは必然的にそのクラスに属するということを示す。表１０に示すＲＤＦステートメントを検討する。

【００６５】

【表１０】

<author> <rdfs:domain> <book>.
<lord of the rings> <author> <J.R.R.Tolkien>.

Table 10.

【００６６】

これらのステートメントより、「著者（author）」は、クラス「本（book）」のプロパティであることがわかる。「author」プロパティが「ロードオブザリング（lord of the rings）」のサブジェクトの述部として使用されている場合は、「lord of the rings」が「book」のクラスに属することが推断できる。公知であるが、このような推断は、ＲＤＦＳ（ＲＤＦスキーマ）推測エンジンを用いて特定することができる。

【００６７】

第３のＲＤＦメタプロパティは、rdfs:rangeである。rdfs:rangeは、結果として生じる推断がトリプルステートメントにおいて、サブジェクトではなく、オブジェクトに適用される点を除き、実質的にはrdfs:domainに類似する。表１１に示したＲＤＦステートメントを検討する。

【００６８】

【表１１】

<eat> <rdfs:range> <food>.
<human> <eat> <vegetables>.

Table 11.

【００６９】
これらのステートメントより、「食べる（eat）」はクラス「食べ物（food）」のプロパティであることがわかる。プロパティ「eat」がオブジェクト「野菜（vegetables）」の述部として使用されている場合は、「vegetables」がクラス「food」に属することが推断される。公知であるが、このような推断は、ＲＤＦＳ推測エンジンを用いて特定することができる。
【００７０】
第４のＲＤＦメタプロパティは、rdfs:subClassOfである。<A> <rdfs:subClassOf><B>というフォームのステートメントがあるとすると、「Ａ」はクラスであり、「Ａ」はクラス「Ｂ」のすべてのプロパティを共有するということがわかる。
【００７１】
さらに、クラスのプロパティに関する既存の知識も利用される。すなわち、頻繁に生じうることであるが、与えられたリソースがそのクラスを示すオントロジー情報を一切有していない場合には、変換プロセスは、利用可能なプロパティを特定し、既存のクラス／テーブルとこれらのプロパティを比較し、可能であればこれらを一致させるよう試みる。
【００７２】
上述したメタプロパティに依存した変換プロセスを、図４にさらに示す。特に、図４は、トリプルストアデータベース３０８および関係データベース３０４のコンポーネントを示し、特にデータ変換に関わるコンポーネントを詳細に示す。図示のように、ＲＤＦデータは、ＲＤＦ　ＤＢＭＳ４０２によって維持され、同様に、関係データは、ＤＢＭＳ４０４によって維持される。一実施態様においては、外部ＲＤＦデータストア４０６からのＲＤＦデータは、公知のＲＤＦローダー４０８を介して、ＲＤＦ　ＤＢＭＳ４０４にインポートされる。外部ＲＤＦデータから関係データへの変換を完遂するために、トリプルストアデータベース３０８は、変換ブリッジ４１２および推測エンジン４１４を含む。集合的には、変換ブリッジ４１２および推測エンジン４１４は、ＲＤＦデータ４１０を関係データ４１６に実際に変換するＲＤＦＳコンバータを構成する。すなわち、下記に詳述するが、変換ブリッジ４１２は、メタプロパティを特定するためにＲＤＦデータ４１０を点検し、必要に応じて推測エンジン４１４の補助を受けながら、リレーショナル（関係）データベースストレージフォーマットに基づいて構成された関係データ４１６を拡大するために使用されるプロパティを決定する。
【００７３】
特に、変換ブリッジ４１２は、ＲＤＦデータ４０内のトリプルを処理して、それぞれのトリプルのサブジェクトおよびオブジェクトの両方に関するメタプロパティを検索する。したがって、メタプロパティ「rdf:type」が発見されたそれぞれのステートメントについては、変換ブリッジ４１２は、まずオブジェクトを抽出し、リソースのクラスを特定する。その後、変換ブリッジ４１２は、抽出されたクラス名と同一のテーブル名を有するテーブルを特定するため、すべてのテーブルを検索する。そのようなテーブルが発見されると、変換ブリッジ４１２は、既存のテーブルのプロパティ（カラムの定義など）と新規リソースのプロパティとを比較する。これらが一致しなければ、変換ブリッジ４１２は、テーブルカラムの定義に新規リソースのプロパティを追加する。つまり、新規リソースのプロパティを含むためにテーブルカラムの定義を拡大する。そのようなテーブルが発見されなかった場合には、変換ブリッジ４１２は、クラスの属性を判定するために、ＲＤＦデータにおけるリソースのクラスに関する「rdfs:domain」および「rdfs:range」メタプロパティを検索する。さらに、変換ブリッジ４１２は、クラスのオブジェクトのプロパティを検

索する。これらの追加的な作業を行ったにもかかわらず、そのようなプロパティあるいは属性が発見されない場合には、新規リソース名からテーブル名を採用し、「_UNKNOWN_CLASS」のストリングをテーブル名に続けた新規テーブルを作成する。

【００７４】

メタプロパティ「rdfs:subClassOf」が発見された場合には、変換ブリッジ４１２は、このリソースがクラスであり、このリソースはテーブルとして表されるべきと認識する。この現在のクラスとその親クラスの両方について、変換ブリッジ４１２は、いずれかのクラスがそれに関するプロパティをまだ有するかを判断するために検索する。「rdf:type」を伴うリソースおよびオブジェクトとしてのいずれかのクラスが発見されると、リソースに関連するすべてのプロパティが他のクラスのプロパティとして抽出される。メタプロパティ「rdfs:domain」あるいはプロパティとしての「rdfs:range」、およびオブジェクトとしていずれかの１クラスとともにプロパティが発見されると、そのプロパティは、推測エンジン４１４を用いて、対応するクラスのプロパティとして抽出される。「rdfs:subClassOf」プロパティとともに現在のクラスあるいは親クラスのいずれか１つが発見された場合には、これらサブ／親クラスに基づいてこれらのステップが反復される。さらに、現在のクラスについては、変換ブリッジ４１２は、現在のクラスの名前として同一のテーブル名を有するテーブルを特定するためにすべてのテーブルを検索する。そのようなテーブルが発見されると、変換ブリッジ４１２は、カラム定義のような既存のテーブルのプロパティと新規リソースのプロパティとを比較する。これらが一致しない場合には、変換ブリッジ４１２は、テーブルのカラム提起に新規リソースのプロパティを追加するが、そのようなテーブルが発見されない場合には、現在のクラス名に基づいて新規テーブルが作成され、現在のクラスのために以前集められたプロパティはカラム定義として利用される。より多くの「rdfs:subClassOf」ステートメントが発見された場合には、新規の現在のクラスおよび親クラスに基づいて、以前のステップが反復される。

【００７５】

ＲＤＦデータ４１０を処理するに際して、変換ブリッジ４１２は、与えられたリソースがオントロジー情報（上述したメタプロパティにより提供されたもの）を有していないことを判定する。この場合、変換ブリッジ４１２は、リソースの既知のプロパティの比較に基づいてリソースの分類を試みようとする。特に、変換ブリッジ４１２には、信頼レベルｃ（０≦ｃ≦１）が提供される。たとえば、信頼レベルは、ワークステーション２０６のユーザや、アドミニストレータなどにより提供される。信頼レベルのソースにかかわらず、変換ブリッジ４１２は、現在のユーザがすでにアクセスした、すべての利用可能なテーブルを検索し、それぞれのテーブルについて、カラムの数をカウントし、そのカラム計数値とプロパティの未分類リソースの数であるプロパティ計数値とを比較する。カラム計数値およびプロパティ計数値の大きい方をｎ、小さい方をｍとすると、これら２つの共通プロパティの数ｐがカウントされる。ｐ≧ｍ＊ｃである場合、テーブルのカラムおよびリソースのプロパティの類似性が十分に高いことを示すので、変換ブリッジ４１２は、そのテーブル名をリストに一時的に記録する。このようにしてすべてのテーブルが処理された後、前記リストが検索され、このリストが空の場合、すなわち、十分に類似するテーブルが特定されなかったことを示した場合には、未分類リソースを、既知の情報によって分類することは不可能である。この場合、変換ブリッジ４１２は、この未分類リソースを新規クラスとして扱い、この未分類リソース名のあとに「_UNKNOWN_CLASS」のストリングを付した新規テーブルを作成し、このリソースを新規テーブルに挿入する。一方で、リストが空でなかった場合には、ｐの最大値を有するテーブルが特定される。すると、変換ブリッジ４１２は、特定されたテーブルはリソースのクラスであると推測して、上述したようにプロパティを比較し、必要に応じてテーブルカラム定義を拡大する。その後、前記リソースはテーブルに挿入される。この方法では、ＲＤＦデータ４１０がオントロジー情報（メタプロパティ）を有さず、すべてのリソースが完全に異なるプロパティを共有している場合に、最悪のケースのシナリオが生じる。最悪のケースのシナリオにおいては、変換ブリッジ４１２は、それぞれのテーブルに１つのレコードが付された潜在的に多数のテーブルを

生成することになる。このような問題を回避するため、信頼レベルを０に設定することで、すべての未分類リソースが同一のクラスとして扱われ、同一のテーブルに挿入されるが、これも望ましい結果ではない。したがって、信頼レベルは、作成されたテーブル数と分類の正確さとのバランスを図る。

【００７６】

ＲＤＦデータ４１０から関係データ４１６への変換が完了すると、ＲＤＦデータ４１６は、関係ＤＢＭＳ４０４に追加される。ＲＤＦローダー４０８と同様に、関係ＤＢＭＳ４０４は、公知であるが、関係データをＲＤＦデータ４２０に直接エクスポートすることができるＲＤＦエクスポーター４１８と通信する。

【００７７】

図４は、ＲＤＦ　ＤＢＭＳ４０２および関係ＤＢＭＳ４０４とともに使用される追加的なコンポーネントを示す。たとえば、公知であるが、それぞれのユーザが有する特定の権利（ユーザの権限）、有効なユーザ（User）の識別、および特定のユーザロール（Roles）の識別を図示したように管理するために、アドミニストレーションコンポーネント４２２が設けられる。さらに図示したように、ユーザがＲＤＦと関係データにアクセスするためのさまざまな方法を提供するために、多数のクエリインターフェースが設けられる。たとえば、公知のＳＰＡＲＱＬエンドポイント４２４は、いわゆるＳＰＡＲＱＬ　ＲＤＦクエリプロトコル４２６をサポートする。このようにして、ユーザは、ＳＰＡＲＱＬクエリ４２８を使用して、ＲＤＦ　ＤＢＭＳ４０４に直接アクセスする。代替的に、上述の統一ＡＰＩ４３０は、ＲＤＦ　ＤＢＭＳ４０２にアクセスするためのＳＰＡＲＱＬクエリ４２８およびＳＱＬのようなクエリ４３２をサポートするだけでなく、関係ＤＢＭＳ４０２にアクセスするためのＳＱＬクエリ４３３の使用もサポートする。

【００７８】

図３について再度言及すると、上述したタイプのオブジェクト３１０は、コントローラ３０２によって採用されるオブジェクト中心のアプローチを強調するために、コントローラ３０２内の中央に示されている。さらに、コントローラ３０２は、オブジェクトの使用に起因する多数の機能を提供する。表１に示したように、オブジェクトは、オブジェクトの性質によって、異なる状態を反映するために複数の値を採りうる１つ以上の状態インジケータを含む。状態管理コンポーネント３１２は、システム３００内におけるすべてのオブジェクトに関するそのような状態情報を追跡する。たとえば、下記に詳述するが、個別のオブジェクトは、互いに幅広い関係を有することが可能であり、そのような関係は状態インジケータに反映される。たとえば、特定のデータを代表するオブジェクトは、そのオブジェクトが他のデータオブジェクトを駆動するか（たとえば、「単位価格」のデータオブジェクトが「購入累計金額」のデータオブジェクトを駆動するか）、あるいは他のデータオブジェクトによって駆動されるか（「購入累計金額」のデータオブジェクトは「単位価格」のデータオブジェクトによって駆動されるか）というインジケータを含む。あるいは、ウィジェットは、本明細書中で使用されているように、互いにさまざまな関係を有する他のオブジェクト（あるいはウィジェット）の集合体となり得るオブジェクトを参照する。これらの構成オブジェクト（および／または他のウィジェット）間の関係は、「contains（有する）」、「has child（子供がある）」、「has parent（親がいる）」などの複数の状態値に反映される。さらに、状態データは、オブジェクトの一時的な使用状態、たとえば「can be used（使用可能）」、「is used（使用中）」、あるいは「has been used（使用済み）」などの状態値に反映される。さらに、状態インジケータは、バイナリーの性質、すなわち、「hidden（見えない）」に対する「visible（見える）」の状態値、あるいは「enabled（できる）」に対する「disabled（できない）」の状態値の場合のような性質を有する。上述した例に示されるとおり、無数の使用状態インジケータおよび値が採用され得る。

【００７９】

履歴管理コンポーネント３１４は、それぞれのオブジェクトへの修正に関する情報を維持し、どの修正が最新かをトラッキングする。上述した状態インジケータと同様に、修正

状態は、「current（現在の）」、「archived（記録された）」、「deleted（削除された）」、あるいは「historic（過去の）」を含み、これらのすべては、データベース３０４～３０８内におけるそれぞれのオブジェクトのために履歴管理コンポーネント３１４により追跡される。

【００８０】

切断コンポーネント３１６は、所定のオブジェクトがコントローラ３０２との接続が失われた場合に生じうるコンフリクト状態を管理するために設けられる。下記に詳述するように、コントローラ３０２に追跡されたオブジェクト、特に、ソフトウェアウィジェットあるいは他の離散機能コンポーネントに関するオブジェクトは、エンドユーザアプリケーションを構成するために使用される。このため、アプリケーションが特定のオブジェクトに基づいて構築されることから、アプリケーションの作者は、切断時においても利用可能な一定のオブジェクトを示すよう問い合わせを受けるが、この情報は切断コンポーネント３１６により追跡される。その後、アプリケーションは、エンドユーザアプリケーションサーバを介して、エンドユーザに利用可能となる。エンドユーザが、このエンドユーザアプリケーションサーバ上のアプリケーションにアクセスすると、このエンドユーザアプリケーションサーバは、切断時の機能のためにどれくらいのローカルストレージを割り当てることが可能か判断するために、クライアント処理装置、たとえば、デスクトップコンピュータ、ラップトップコンピュータ、モバイルワイヤレスデバイスなどとかけあう。ローカルストレージの所望量は、切断時にも利用可能なように必要とされる特定のオブジェクトにある程度依存する。クライアント処理装置とのかけあいプロセスは、同一のアプリケーションを使用する複数の他のエンドユーザ処理装置のために繰り返されるため、それぞれのクライアント処理装置は、指定されたオブジェクトのための同一のローカルストレージを含む。

【００８１】

エンドユーザのクライアント処理装置との間で切断が生じると、コントローラ３０２は、公知の技術を用いてこの状態を検出し、他のエンドユーザクライアント装置には、切断コンポーネント３１６によってこの事実が通知される。追加的に、切断されたエンドユーザクライアント装置は、指定されたオブジェクトの操作を維持するために、ローカルストレージを使用するモードに切り替わる。たとえば、指定されたオブジェクトが購入注文を追跡するウィジェットである場合、たとえば「会社Ａから会社Ｂに、１０００部分の発注書（Ｐ．Ｏ．）を送る」というウィジェットの使用が、データの作成、読み取り、更新、削除が引き続き可能とするという意味でローカルストレージのみにおいて維持される。一方で、他のエンドユーザクライアント装置は、同一の指定されたオブジェクトを切断されたクライアント装置とコンフリクトしかねない方法で使用すること（たとえば、会社Ａから会社Ｂに、２０００部分の発注書を送る）を含め、通常通り作動し続けることができる。このような場合には、切断コンポーネント３１６は、他のエンドユーザクライアントによる指定されたコンポーネントの使用を追跡する。切断されたクライアント装置がコントローラ３０２との接続を再開すると、ローカルストレージに保存されたデータは、コントローラ３０２にアップロードされ、切断コンポーネント３１６はコンフリクトの発生を検出する。要するに、切断コンポーネント３１６は、切断中に切断されたエンドユーザクライアントによって使用された、すべての指定されたコンポーネントに関するすべてのデータを「隔離」する。

【００８２】

コンフリクトを検出すると、切断コンポーネント３１６は、さまざまな方法でコンフリクトを解消することが可能である。一実施態様においては、切断コンポーネント３１６はさまざまなエンドユーザクライアント装置の階層に関する規則を有する。たとえば、どのエンドユーザクライアントが他のエンドユーザクライアントに優先すべきかを判断するために、企業内あるいは同様に階層的に組織された組織内において、特定のエンドユーザクライアントは、肩書き、地位、その他の優先性を示すインジケータに関連づけられることにより、より高い優先度を有するエンドユーザクライアントによって提供されるデータに

基づいて、コンフリクトが自動的に解消される。そのような自動的な解決が不可能な場合には、コントローラ３０２は、コンフリクトを起こしているデータを、コンフリクトを解消するための要求とともに、コンフリクトを起こしているエンドユーザクライアント装置に送ることが可能である。その後、コンフリクトを起こしているクライアントがコンフリクトを解消できると仮定して、どのようにコンフリクトを解消することが可能か（つまり、どのデータを保存するか）を示しながら、データは切断コンポーネント３１６に戻される。

【００８３】

状態管理コンポーネント３１２により維持された状態情報に基づいて、それぞれのオブジェクトについてソーシャルネットワークが構築される。すなわち、それぞれのオブジェクトについて維持された関係情報を用いると、オブジェクトおよびオブジェクト同士の関係のネットワーク表示を作成することが可能となる。たとえば、「従業員のファーストネーム」オブジェクトと「従業員のラストネーム」オブジェクトは、それぞれ、「従業員の名前」に関連する「belongs to（所属する）」状態を反映し、他のオブジェクトなどへ独自の接続を有する。そのようなネットワークは、公知のネットワーク発見技術を用いたネットワークコンポーネント３１８により得られる。たとえば、下記の自動データマイニングコンポーネント３３２を実行するために使用される、データマイニングサーバ２２２によって、公知のデータマイニング技術、たとえば、根本原因解析、分類、クラスタリング、結合規則発見、および／または回帰分析を利用することができる。

【００８４】

さらに、図示のように、根本原因解析コンポーネント３２０（オブジェクトソーシャルネットワークを生産するためのネットワークコンポーネント３１８により用いられる根本原因解析とは異なるもの）が設けられる。ニューラルネットワーク解析や回帰分析などの公知の技術を用いて、オブジェクトソーシャルネットワーク内において、ネットワークコンポーネント３１８によって提供された根本原因は、特定のオブジェクトに関連して特定される。より正確には、ソーシャルネットワークは、根本原因を常に直接特定することは不可能であるが、時には潜在的な因果関係である相関関係が特定される。つまり、比較的簡素で明瞭なソーシャルネットワークについては、根本原因は確実性を伴って特定される。しかしながら、複雑かつ／または不明瞭なソーシャルネットワークについては、人間による追加的な解析を条件に特定することが可能である。たとえば、「従業員の能率」オブジェクトに関する複数のオブジェクトには、「従業員の年齢」、「従業員のスキルレベル」、「曜日」、「工場の気温」などが含まれる。ニューラルネットワーク解析の場合には、これらのオブジェクトの基礎となるデータは、「従業員の能率」オブジェクトの値を予測するにあたり最も重要となる要素を効果的に明らかにするネットワーク機能を明らかにするために、公知の技術を用いて解析される。そのような根本原因の識別は、これまでに存在しなかったオブジェクト間の関係を作るために利用されるか、あるいは従前に定義された関係を更新あるいは削除するために利用される。

【００８５】

上述したように、システム２００および３００に保存されたデータとともに使用されるアプリケーションは、複数の階層型ユーザインターフェースを使用して開発される。図示の例においては、階層型ユーザインターフェースは、第１メジャーデベロッパインターフェース３２２、第２メジャーデベロッパインターフェース３２４、およびマイナーデベロッパインターフェース３２６を備える。すべてのデベロッパインターフェース３２２～３２６はオプションであり、これらのいかなる組み合わせも利用可能である。一般的には、それぞれのデベロッパインターフェース３２２～３２６は、２つの使用パターンあるいはロールを有する。具体的には、異なるユーザにより異なる目的のために個別に用いられる独立プラットフォームと、一元的なシステムとして機能するために他のプラットフォームと協働するプラットフォームである。一実施態様においては、第１メジャーデベロッパインターフェース３２２、第２メジャーデベロッパインターフェース３２４、およびマイナーデベロッパインターフェース３２６は、ソフトウェア開発において、連続したより高次

の抽象化レイヤーとして機能する。抽象化のレベルが高次元になるほど、多くのプログラミングの詳細が漸次的に隠されていくことになるため、アプリケーションの開発用としては使用しやすくなる。

【００８６】

したがって、一実施態様においては、第１メジャーデベロッパインターフェース３２２は、Ａｐａｃｈｅ　Ｅｃｌｉｐｓｅのような公知の統合開発環境（ＩＤＥ）である。第１メジャーデベロッパインターフェース３２２を用いると、比較的熟練したプログラマーであれば、いかなるタイプのソフトウェアでも開発することができる。第２メジャーデベロッパインターフェース３２４は、ＧＵＩアプリケーションビルダー（完全に機能的なＧＵＩアプリケーションを中程度の抽象化レベルで構築するために用いることができ、第１メジャーデベロッパインターフェース３２２を実行するために用いられる同一のアプリケーションを含む）を用いて実行することができる。マイナーデベロッパインターフェース３２６は、公知であるが、ソフトウェア開発スキルをほとんど有しない個人に、高次のファンクショナルビルディングブロックに基づいてアプリケーションソンを構築することを可能にする、ゾーホークリエータ（Zoho Creator）などの複数のグラフィカルウェブアプリケーションビルダーを備える。したがって、第１メジャーデベロッパインターフェース３２２による低次の抽象化は、特定のプログラミング言語特徴を扱うユーザにのみ利用可能であるが、第２メジャーデベロッパインターフェース３２４で用いられる機能は、プログラミング言語に依存しておらず、さらに、マイナーデベロッパインターフェース３２６においては、プログラミングに特有な用語あるいは特徴はまったく必要とされない。

【００８７】

操作中において、公知であるが、第１メジャーデベロッパインターフェース３２２は、第１メジャーデベロッパインターフェース３２２のユーザにソフトウェアコードの生成および修正を許可する複数のモードを提供する。たとえば、いくつかのＩＤＥは、定義され、選択可能なタスクを備える。与えられたタスクを選択するにあたり、コードテンプレートも選択されるため、ＩＤＥは、選択されたテンプレートに基づいて自動的にコードを生成する。あるいは、ユーザは、一連のドロップダウンメニューから操作を定義することができる。このドロップダウンメニューは、利用可能な操作を示すために常に更新され続ける。ユーザがさまざまな操作を選択すると、コードが自動的に生成される。さらに別の実施形態においては、中間コマンドを提供するためにユーザにより提供された自然言語テキストを解析するため、自然言語処理エンジンが使用される。この中間コマンドは、その後、自動的に生成されるコードを提供するために解析される。いずれの実施形態においても、自動的に生成されたコードは、最終的な所望のコードを提供するために、ユーザによって希望通りに修正される。

【００８８】

第２メジャーデベロッパインターフェース３２４は、さまざまなユーザインターフェースコントロールがツールボックス内に備えられている、公知の「ドラッグアンドドロップ」式のグラフィカルユーザインターフェースを提供する。利用可能なさまざまなユーザインターフェースコントロールは、選択されたコントロールのインスタンスを作成するために、デザインエリアにドラッグされる。このインスタンスは、その後選択され、一定のビヘイビアを示すよう構成される。同様にして、所望のイベント定義、フローコントロールあるいはアクションはいずれも、選択されたコントロールインスタンスに追加される。このようなコントロールを結合させることにより、ウィジェットあるいはより完全なアプリケーションが作成されて、ユーザインターフェースの所望の機能性が実装される。これが完全に構成されると、結果として得られたウィジェットあるいはアプリケーションが発行される。

【００８９】

なお、第１および第２メジャーデベロッパインターフェース３２２、３２４により作成されたコードおよび／あるいはウィジェットは、すべてコントローラ３０２によりオブジェクトとして保存される。

【００９０】
第２メジャーデベロッパインターフェース３２４と同様に、マイナーデベロッパインターフェース３２６も「ドラッグアンドドロップ」式のＧＵＩに基づいている。しかしながら、マイナーデベロッパインターフェース３２６用に設けられたツールボックスは、デザインエリアにおいて選択され結合される、発行済みのウィジェットあるいはアプリケーションを含む。完全なアプリケーションが定義されると、公知の技術を利用した第２メジャーデベロッパインターフェース３２４は、たとえばユーザインターフェースマークアップ言語（Ｑｔメタ言語（ＱＭＬ）など）および／またはファンクショナルマークアップ言語（ビヘイビアマークアップ言語（ＢＭＬ）など）を用いて、個別のウィジェットの操作およびこれらそれぞれの互いの関係を示すアプリケーションメタファイルを生成する。結果として得られたアプリケーションメタファイルは、その後、ソースおよび実行可能なコードを生成するコードジェネレータ３２８に送られる。このようなコードジェネレータの例としては、エクリプスファウンデーション（Eclipse Foundation）から発売されているＡｃｃｅｌｅｏオープンソースコードジェネレータがある。結果として得られたソースコードおよび実行可能なコードな、コントローラ３０２によりオブジェクトとして保存され、実行可能なコード３３０は、適切なアプリケーションサーバなどを介して、エンドユーザに利用可能となる。

【００９１】
上述したように、インターフェース３２２～３２６は、協働的に用いることができる。たとえば、第１メジャーデベロッパインターフェース３２２は、互換性のある特定のプログラミング言語を用いて、開発中のコンストラクトに集中するために用いることも可能である。つまり、第２メジャーデベロッパツール３２４による使用のためのプログラミング言語エンティティおよびロジックラッパを構築することが可能である。たとえば、第１メジャーデベロッパインターフェース３２２を用いると、開発者は、Ｊａｖａ　ＧＵＩコンポーネント（たとえば、テキスト入力ボックス）を特定のオブジェクトにラップすることができ、コントローラ３０２を通して、そのオブジェクトを第２メジャーデベロッパインターフェース３２４において利用可能とすることによって、第２メジャーデベロッパインターフェース３２４は、このオブジェクトを今後の使用のためにツールボックスに追加することが可能となる。このようにして、第１メジャーデベロッパインターフェース３２２は、第２メジャーデベロッパインターフェース３２４にとっては「プラグイン」であるとみなすことができ、このような機能によって第２メジャーデベロッパインターフェースの機能が拡張される。

【００９２】
一方、第２メジャーデベロッパインターフェース３２４は、第２メジャーデベロッパインターフェース３２４が開発することができるアプリケーションのタイプに集中すること、すなわち、マイナーデベロッパインターフェース３２６が使用するためのＧＵＩコンポーネントおよびロジックラッパを構築することに関して、協働的に使用することが可能である。たとえば、第２メジャーデベロッパインターフェース３２４を用いて、開発者は、「サブミット（Submit）」ボタンをラップして、このボタンへのシングルクリックにより、すべてのデータを現在のスクリーン上に集めたり、データベース３０４～３０６に送ったり、このオブジェクトをマイナーデベロッパインターフェース３２６に付与したり、マイナーデベロッパインターフェース３２６がこのオブジェクトを後に使用できるようにするため、マイナーデベロッパインターフェース３２６にそのツールボックスへのオブジェクトの追加を許可したりすることを可能とするロジックを持たせることができる。この場合、第２メジャーデベロッパインターフェース３２４は、マイナーデベロッパインターフェース３２６にとっては「プラグイン」であるとみなすことができ、このような機能によってマイナーデベロッパインターフェースの機能が拡張される。

【００９３】
図３に示すように、システム３００は、保存されたデータをユーザが扱う能力を高めるさまざまな機能を備える。一実施形態では、自動データマイニングコンポーネント３３２

は、データベース３０４～３０６に保存されたデータについて適用される、公知のさまざまなデータマイニングアルゴリズムを、コントローラ３０２を介して、実行する。具体的には、自動データマイニングコンポーネント３３２は、与えられたデータマイニングタスクのための処理前データを最良にし、かつ、このデータマイニングタスクのために最良のデータマイニングアルゴリズムを選択する。

【００９４】

公知ではあるが、データマイニングでは、解析されるデータに前処理が行われた場合に、最良の結果がもたらされる。しかしながら、そのような前処理は、解析されるデータの性質に強く依存する。自動データマイニングコンポーネント３３２は、最良のデータ前処理を自動的に選択するためにトレーニングをすることができる。このため、サンプルのデータセットがまず集められ、その統計上の特徴が抽出される。そのような統計上の特徴は、たとえば、平均値、中央値、値域、および標準偏差などの数学的特徴を含む。さらに、属性数やそれぞれの属性のタイプ（たとえば、名目上あるいは数値上）、データセットのサイズなどの単なる事実も含む。このようにしてデータセットが特徴づけられると、Ｎ個の公知のデータ処理前アルゴリズムが、データセットに対して実行され、それぞれの処理前アルゴリズムについて結果として生じた処理前データを個別に保存する。その後、Ｍ個の公知のデータマイニングアルゴリズムは、それぞれの処理前データセットにおいて実行されることにより、Ｎ×Ｍ個のデータマイニング結果セットを生じる。それぞれのデータマイニング結果セットは、その後、関係する処理前およびデータマイニングアルゴリズムの組み合わせの精度および正確性を評価するため、公知の技術を用いて評価される。可能な場合は、それぞれのデータ処理前アルゴリズムのパラメータは、処理前アルゴリズムおよびパラメータの最良の組み合わせ、およびデータマイニングアルゴリズムを特定するために変動する。特定されると、処理前アルゴリズム／パラメータ／データマイニングアルゴリズムの最良の組み合わせがクラス属性として指定され、前記データセットの統計学的特徴が入力属性として指定される。これらのクラス／入力属性は、その後、処理前選択学習モデルを増加させるために用いられ、実質的に合致する統計学的特徴を有するその後のデータセットも同様の方法で前処理される。

【００９５】

さらに、与えられたデータマイニングタスクに対しては、特定のデータマイニングアルゴリズムが他のデータマイニングアルゴリズムより優れている場合がある。最良の処理前アルゴリズムを選択するためのトレーニングについて上述したのと同様に、自動データマイニングコンポーネント３３２も、実行されるべき特定のデータマイニングタスクに基づいて自動的に最良のデータマイニング技術を選択するために、トレーニングをすることができる。このため、サンプルのデータセットが再度集められ、その統計学的特徴が抽出される。このように特徴づけられると、Ｎ個の公知のデータ処理前アルゴリズムが、前記データセットに対して実行されるため、それぞれのデータマイニングアルゴリズムについて、結果として得られたデータセットが個別に保存される。データマイニング結果セットのそれぞれは、その後、それぞれのデータマイニングアルゴリズムの精度および正確性を評価するため、公知の技術を用いて評価される。可能な場合は、それぞれのデータマイニングアルゴリズムのパラメータも、データマイニングアルゴリズムおよびパラメータの最良の組み合わせを特定するために変動させる。特定されると、データマイニングアルゴリズムおよびパラメータの最良の組み合わせがクラス属性として指定され、データセットの統計学的特徴が入力属性として指定される。これらのクラス／入力属性は、その後、データマイニング選択学習モデルの増加に用いられ、与えられたデータマイニングタスクを受けるために使用され、かつ、実質的に合致する統計学的特徴を有するその後のデータセットも同様の方法で前処理される。

【００９６】

一実施態様においては、最良の処理前および／またはデータマイニングアルゴリズムの利点は、さらなる処理によって得られる。この処理においては、前処理されるデータセットあるいは与えられたデータマイニングタスクを受けるデータセットは、上述したように

、再度統計学的に特徴づけられる。結果として生じる統計学的特徴に基づき、最良の処理前あるいはデータマイニングアルゴリズムｋは、上述したように、前記データセットおよび入力属性の統計学的特徴間の類似性に基づいて選択される。同時に、前記入力データベースは、公知であるが、データの削減を受けて、すべての利用可能な処理前あるいはデータマイニングアルゴリズムが、削減された入力データセットに対して適用され、最良の処理前あるいはデータマイニングアルゴリズムｎが選択される。さらに同時平行して、最良の処理前あるいはデータマイニングアルゴリズムｍを特定するために機械学習モデルが使用される。その後、最良の処理前あるいはデータマイニングアルゴリズムｈを選択するために、異なる処理前あるいはデータマイニングアルゴリズムｋ、ｍおよびｎの結果が比較される。これら処理前あるいはデータマイニングアルゴリズムｈは、その後、前記入力データセットに対して実行され、結果が平均化される。結果として生じた平均出力は、その後、可能性のある最良の処理前あるいはデータマイニングアルゴリズムの組み合わせを示す。

【００９７】

別のデータマイニングの実施形態においては、自然言語処理エンジン３３６とともに任意に作動する質問駆動型データマイニングコンポーネント３３４が設けられている。質問駆動型データマイニングコンポーネント３３４は、データマイニングの経験がほとんどないかまったくないユーザに、データマイニングタスクを行えるメカニズムを提供する。エンドユーザが、コントローラ３０２にデータマイニング要求を与えると、コントローラ３０２は、その要求を質問駆動型データマイニングコンポーネント３３４に直接送るか、要求が自然言語テキストで表されている場合には、必要なデータセットの解析のために質問駆動型データマイニングコンポーネント３３４が使用可能な命令に変換するためのＮＬＰエンジン３３６を通じて、質問駆動型データマイニングコンポーネント３３４に送られる。

【００９８】

より具体的には、質問駆動型データマイニングコンポーネント３３４は、たとえば、その特定の目的のためのユーザインターフェースを介して、自然言語で表されたユーザの質問を受領する。これらの複雑な質問、たとえば、「なぜ」や「どのように」といった形式で表現された質問が受領されると、質問駆動型データマイニングコンポーネント３３４は、後述するように、ＮＬＰエンジンコンポーネント３３６による処理を行う。ＮＬＰエンジンコンポーネント３３６が質問の複雑な形式を扱えない場合には、ＮＬＰエンジンコンポーネント３３６は、その質問を解析し、自動データマイニングコンポーネント３３２によって実行されるデータマイニングタスクの形に解析する。ＮＬＰエンジンコンポーネント３３６は、質問駆動型データマイニングコンポーネント３３４に、データマイニング操作の必要性を伝え、これにより、質問駆動型データマイニングコンポーネント３３４は、前記データマイニングタスクを開始するために必要なパラメータ（たとえば、本明細書の付属資料に基づいて表現された要求の形式）を生成する。これらのパラメータは、自動データマイニングコンポーネント３３２におけるとの同様に、データマイニングタスクを開始するために、質問駆動型データマイニングコンポーネント３３４に戻された結果とともに使用される。ユーザに結果を提供するために、質問駆動型データマイニングコンポーネント３３４は、その後、前記結果を、ＮＬＰエンジンコンポーネント３３６に送る。

【００９９】

一実施態様においては、上述したように、利用可能なデータマイニング操作を提供するために、自動データマイニングコンポーネント３３２は、特定のタイプのデータマイニング操作の実行を要求するＨＴＴＰ（ハイパーテキストトランスファープロトコル）ＰＯＳＴ要求のフォーマットにて外部要求を受領するよう、ＡＰＩメソッドを提示する。要求に対して、ＡＰＩは、要求された操作を他のＨＴＴＰ　ＰＯＳＴのフォーマットで完了するための時間の見積を回答することができる。公知であり、また、上述したように、根本原因解析、分類、クラスタリング、結合規則、発見、回帰分析などのさまざまな異なるタイプのデータマイニングタスクが提供される。

【０１００】
要約すると、ＡＰＩによる処理は、以下のように表される。
１．ＡＰＩメソッドは、ＨＴＴＰ　ＰＯＳＴ要求を受領する。
２．ＡＰＩメソッドは、要求データを抽出し、データを分析する。
３．ＡＰＩメソッドは、要求データを確認する。要求が受け入れ可能である場合には、ステップ５へと処理を進める。
４．要求が受け入れ不能である場合には、ＡＰＩメソッドは、エラー情報を含むＨＴＴＰ　ＰＯＳＴ応答を返し、処理は終了する。
５．要求が受け入れ可能である場合には、ＡＰＩメソッドは、必要となる時間の見積を計算する時間概算モジュールに対し、選択されたデータに基づいて要求を実行するよう命令する。
６．ＡＰＩメソッドは、時間の見積を含んだＨＴＴＰ　ＰＯＳＴ応答を返す。
７．要求における情報に基づき、ＡＰＩメソッドは、関係するオブジェクトをコントローラを介して特定し、それによって要求されたデータを特定し、自動的にデータ処理ステップを適用し、上述したように最良のアルゴリズムを選択し、データマイニング処理を実行する。
８．処理が完了すると、ＡＰＩメソッドは、要求者に対して結果を返す。
【０１０１】
本発明の好ましい実施形態においては、ＨＴＴＰに基づいたインターフェースを介して送られたメッセージは、ＪＳＯＮスタンダードフォーマットを使用する。ＡＰＩメソッドに関するさらなる詳細は、本明細書の付属資料に記載されている。
【０１０２】
このように、自動データマイニングコンポーネント３３２によって提示されたＡＰＩは、ＰＯＳＴ応答で返されたエラーメッセージとともに、要求されたヘッダおよび本明細書の付属資料に記載されたＪＳＯＮスキーマに対するそれぞれのＰＯＳＴ要求を確認する。一実施態様においては、自動データマイニングコンポーネント３３２は、ｃｓｖやａｒｆファイルフォーマットなどの予め定義されたフォーマットにて、データセットアップロードを受け入れ、アップロードされたそれぞれのデータセットに独自の識別が付されるべきである。さらに、自動データマイニングコンポーネント３３２は、公知であるが、エンドユーザが既存のデータベースから受け入れ可能なフォーマットにデータをエクスポートすることを補助するために、１つ以上のデータセットエクスポーターヘルパーツールを提供することができる。
【０１０３】
自動データマイニングコンポーネント３３２に関して上述したように、質問駆動型データマイニングコンポーネント３３４は、自動的に最良の処理前および／またはデータマイニングアルゴリズムを選択することも可能である。要求されたデータマイニングタスクについて時間の見積を提供するために、質問駆動型データマイニングコンポーネント３３４は、自動データマイニングコンポーネント３３２により提示されたＡＰＩを介して時間の見積を入手することができる。そのような見積は、サイズ、選択されたデータ準備方法、選択されたトレーニングスキームなどを含む前記入力データの特徴に基づいて計算され、コンピューティングリソースは、このタスクに割り当てられる。これは、最初に、十分なサイズを有し、一般的なデータマイニングタスク特徴を最も良く反映するデータ特徴において十分に多様である、学習ベースデータセットに対して、機械学習アルゴリズムを用いて行われる。このデータセットについて学習を終えたあと、質問駆動型データマイニングコンポーネント３３４は、時間の見積に使用することができるモデルを開発する。この時間見積学習モデルへの改良は、初期トレーニングおよび開発に続き、すべてのユーザ入力の収集を通じてもたらされ、定期的に前記機械学習アルゴリズムを再度実行することによって、この時間見積学習モデルの精度は絶えず向上する。
【０１０４】
上述したように、質問駆動型データマイニングコンポーネント３３４は、自然言語で表

された要求を受け入れ、この要求はＮＬＰエンジン３３６によってさらに処理される。下記に詳述するように、ＮＬＰエンジン３３６は、自然言語インターフェーストゥーデータベース（ＮＬＩＤＢ）および自然言語アプリケーションジェネレーション（ＮＬＡＧ）の２つの主要な機能を提供する。

【０１０５】

ＮＬＩＤＢ機能は、人間に理解可能な自然言語でエンドユーザがクエリを依頼することを可能にする。たとえば、そのようなクエリは、「何（what）」、「誰（who）」、「どのように（how）」などの表現を含むことが多い。たとえば、「われわれの商品が最も売れている州はどこか？」や、「昨年１万ドル以上を稼いだ人は誰か？」などである。ＮＬＰエンジン３３６におけるＮＬＩＤＢモジュールは、前記自然言語による質問を分析し、ＳＱＬなどのより技術的なクエリ言語に翻訳するか、好ましくは、上述したようにＳＱＬのような統一されたＡＰＩに翻訳し、次に、基本的なデータエンジンのネイティブクエリＡＰＩに翻訳されることが好ましい。

【０１０６】

ＮＬＩＤＢモジュールは、自然言語の質問を分析するために、逆のアプローチをとる。つまり、統計学的パーサを用いたユーザの質問の分析は行われない。このような技術は、しばしば不正確な結果を導くからである。むしろ、ＮＬＩＤＢモジュールシステムは、いくつかの処理後、すべてのサポートされた質問および対応する回答を含む、あらかじめ定義された質問／回答テーブル（Ｏ＆Ａテーブル）において利用可能な質問に対して、ユーザの入力を簡単にマッピングする。実行される際には、このＱ＆Ａテーブルに含まれる回答は、関連する質問に回答するデータを入手するために使用されるクエリである。前記Ｑ＆Ａテーブルは、利用可能なスキーマおよびデータベースに保存されたデータに基づいて生成される。自然言語の文法規則を適用することによって、ＮＬＩＤＢモジュールは、同一の質問の異なるフォームを含む明確な回答を有する、すべてのあり得る質問を生成する。このストラテジは、分析の精度およびリアルタイムのパフォーマンスを得るために、この巨大なリストを保存するために必要とされる、比較的安価なストレージキャパシティを犠牲にする。この分析はストリングマッチングと同程度に単純であるため、そのパフォーマンスは非常に速く、かつリアルタイムの反応を達成する。

【０１０７】

サポートされた質問のいずれにもユーザの入力が合致しなかった場合には、回答の探索に最善を尽くすために、公知の統計学的分析（ＳＰ）プロセスが用いられる。このＳＰプロセスは、まず質問からストップワードを削除し、キーワードのみを残す。ＳＰプロセスは、その後、前記データベースにおいてテキストサーチを行い、関係するデータオブジェクトを戻すが、このプロセスでは正しい回答が発見されることは保証されない。質問に対する正しいあるいは関係する回答を得るように最善を尽くすが、まったく関係ない回答や、フォーマットされておらず理解が困難なデータを戻す場合がある。ＳＰプロセスが行われた後、ユーザには、潜在的な回答のリストと、アクティブ学習のために、回答にグレードをつける要求が与えられる。返された結果の中にユーザが正しい回答を発見した場合には、その回答に対しては、「いいね（サムアップ）」のような単純なスタイルで高いグレードを付与することが、ユーザに要求される。ユーザがどの回答にも満足しない場合には、「悪いね（サムダウン）」のような単純なスタイルで低いグレードを付与することが、ユーザに要求される。ユーザが回答にグレードをつけなかった場合には、そのグレードはニュートラルであるとみなされる。すべてのユーザ入力はそれぞれ記録される。サポートされた質問に合致せずにＳＰによって処理された質問はついては、対応する記録を保存するよう設計されたレポジトリがある。エキスパートチームはこれらの記録を分析することができ、最も一般的に誤って処理された入力については、サポートされた質問にその質問を追加し、前記Ｑ＆Ａテーブルを更新する。

【０１０８】

上述したように、１つ以上のデータベースに保存されたデータに基づいたすべての回答可能な質問は、基礎となるデータベーススキーマのナリッジ（知識）を通じて特定される

。次に、スキーマのフィールドがエンドユーザにより定義される。スキーマフィールドは、通常は意味のある単語で定義されるが、数字、コードあるいは意味を有しない文字などの非自然言語の単語やシンボルが使用されないという保証はない。非自然言語のシンボルが用いられたスキーマフィールドについては、まず、ＮＬＩＤＢモジュールが、データタイプからこのスキーマフィールドの意味論的意味を定義するよう試みる。データタイプが利用可能でなかったり、必要性を満たさなかったりした場合には、ＮＬＩＤＢモジュールは、前記スキーマフィールドの意味論的意味を定義するようユーザに要求する。このことは、上述したように、たとえば、マイナーデベロッパインターフェース３２６を介して行われる。

【０１０９】

解釈可能なスキーマフィールド名については、ＮＬＩＤＢモジュールは、オントロジー上の定義、すなわち、基礎となるオントロジーにおいて用いられる構造の記述において、その単語を調べる。意味が特定されると、ＮＬＩＤＢモジュールは、ユーザのクエリにおける単語に代替するものとして、別名のリストの拡張を開始する。この拡張は、複数の異なる方法で行うことが可能である。１つの方法によれば、高レベルのオントロジー上の定義が別名として用いられる。たとえば、「従業員」は「人」と同じ意味である。他の方法によれば、公知の類義語を特定するためにシソーラスが用いられる。逆に、他の方法によれば、反意語の辞書を用いることにより、与えられた単語について反意語を特定することが可能である。動詞は、欠如概念およびその反意語の組み合わせとして使用することができる（たとえば、「故障した」と「機能していない」は同一の意味をなす）ため、この方法は特に動詞に対して有用である。これらの方法を組み合わせて使用することにより、ＮＬＩＤＢモジュールは、前記スキーマにおける特定の単語について別名のリストを作成することが可能となる。さらに、上述した技術を用いて、別名を特定することに利用することができる単語の数を拡大するために、略語の意味を解明することが望ましい。たとえば、「Ｐ．Ｏ．」はさまざまな意味を有するが、購買課のアプリケーションにおいては、「発注書」を意味するから、このコンテクストを含む略語の定義のリストにおいて、そのように定義される。このようなコンテクストが十分でない場合には、利用可能なオプションのリストをユーザに提示することにより、意味を明確化させることも可能である。

【０１１０】

スキーマの単語およびこれら単語の別名が発見された後、ＮＬＩＤＢモジュールは、スキーマの単語およびこれら単語の関係に基づいて、可能性のある質問を組み立てる。このため、ＮＬＩＤＢモジュールは、オントロジー情報および自然言語シンタックスの両方を使用する。ある単語のオントロジー情報は、質問語に直接マッピングされる。たとえば、「ＤＯＢ」など、あるスキーマの単語が「時間」のタイプである場合は、「いつ・・・？」という質問が生成される。スキーマフィールド同士の関係は、質問を生成する上でもう一つの重要な基礎となる。たとえば、従業員のデータオブジェクトが「名前」フィールドと「ＤＯＢ」フィールドを含む場合には、「ジョン・ドウの生年月日は？」あるいは「ジョン・ドウはいつ生まれたか？」という質問が生成される。さらに、フィールド名を質問の単語に置き換えることに加え、ＮＬＩＤＢモジュールは、フィールド名を「提示せよ」や「～を知る必要がある」、「与えよ」などの命令の単語への置き換えも行う。このマッピングは、質問の単語から始まらない質問を生成する。

【０１１１】

質問が生成されると、それに伴って質問に対応するクエリが生成される。たとえば、「ジョン・ドウの生年月日は？」の質問は、対応するＳＱＬクエリである「SELECT DOB FROM Employee WHERE Name = ‘John Doe’」を有する。このクエリは質問として機能し、自然言語の質問とともに前記Ｑ＆Ａテーブルに保存される。

【０１１２】

使用に際しては、ＮＬＰエンジン３３６は、たとえば、テキストボックスを通じてエンドユーザが質問を入力することができるようにする。どの質問が利用可能であるかを提示するために、オートコンプリーションが用いられる。以前入力された、前記Ｑ＆Ａテーブ

ルにおいて利用可能な質問のいずれにも合致しない単語とともにユーザが単語を入力した場合、このオートコンプリーションは、サポートされていない可能性のある質問が入力されたことをユーザに警告するため、空のリストを提示する。ユーザは、スペルチェックサービスを用いた単語により、単語を入力する。誤記を含む単語が特定された場合には、たとえば、色を付けるなどして強調される。ユーザは、提示された単語の１つを用いてこれを訂正することも、そのまま放置しておくことも可能である。たとえば、英文法などの正式な自然言語構文法に従わない質問をユーザが入力した場合には、ユーザが入力を止めると、ユーザには、入力された質問に類似するが、構文的には正しい質問の提案リストが提示される。

【０１１３】

ユーザの入力した質問が利用可能な質問に合致しない場合には、ＮＬＩＤＢモジュールは、その質問を前記Ｑ＆Ａテーブルにおいて検索し、データベースクエリのフォームで保存された回答を発見し、データベースに対してクエリを実行し、エンドユーザに結果を返す。ユーザの入力した質問が利用可能な質問に合致しない場合には、上述したように、統計学的処理が行われる。

【０１１４】

ＮＬＡＧファンクションについては、スキーマフィールドはアプリケーションモジュールキーワードに置き換えられ、質問はファンクション説明ステートメントに置き換えられる点を除いて、ＮＬＩＤＢファンクションに関して上述した方法論と同じ方法論が用いられる。つまり、ＮＬＡＧファンクションは、たとえばマイナーデベロッパインターフェースのユーザなどのユーザが自然言語の説明に基づいてアプリケーションを生成することを手助けする。アプリケーションは、それぞれのモジュールがサブ機能を果たす機能モジュールあるいはコンポーネントにより組み立てられる。このアプリケーションの記述は、このアプリケーションに期待される機能あるいはアプリケーションが何を達成するかについて説明すべきである。例としては、「従業員を管理するプログラムが必要」や、より具体的なものとしては、「従業員の情報を追加、編集、更新、削除することができ、発注書を受け入れ、組み立てラインの状態を見ることができるアプリケーションが欲しい」などを含む。これらの記述は、高いレベルあるいは階層型の機能要求のいずれかを現す。

【０１１５】

オントロジーの辞書を使用することにより、ＮＬＰエンジン３３６内のＮＬＡＧモジュールは、異なるレベルの要求を認識する。この機能をサポートするために、たとえば、上述したように、ウィジェットなどのアプリケーションモジュールの作者は、動詞－名詞パターンのフォーマットにてモジュールの機能の説明を提供しなければならない。たとえば、従業員管理モジュールは、「従業員管理」の説明を有することができ、組み立てラインダッシュボードモジュールは、「組み立てラインの状態を提示せよ」との記述を含むことができる。これらの動詞－名詞ペアは、その後、オントロジーの辞書にて検索され、別名の拡張、質問（この場合には、ステートメントである）の生成、およびクエリ（この場合には、モジュールアセンブリである）の生成を含め、ＮＬＩＤＢファンクションについて上述した処理と同一の処理がなされる。前記質問（ステートメント）構文解析の段階は、オートコンプリーションによりユーザ入力を制限することと、合致しなかった入力の統計学的処理に関しても同様である。ユーザ入力の解析が無事に行われ、モジュールのリストが返されると、たとえば、マイナーデベロッパインターフェース３２６などのアプリケーション開発ツールは、上述したように、ユーザがモジュールを統一されたアプリケーションに組み立てることを可能にする。

【０１１６】

最後に、レポートエンジンコンポーネント３４０が設けられる。一実施態様においては、レポートエンジンコンポーネント３４０は、マイナーデベロッパインターフェース３２６のサブコンポーネントである。特に、レポートエンジンコンポーネント３４０は、システム内におけるすべての（選択された）データを含むグランドテーブルをまず生成することにより、ユーザがレポートを構築することを可能にするＧＵＩレポートビルダーである

。ユーザは、このグランドテーブルからカラムを削除し、合計（sum）や平均（average）などの集計関数をカラムに追加し、あるいは新規テーブルに結果として生じる既存のカラムについての計算に基づいた新規カラムを追加することができる。この処理は、最終的な所望のテーブルが得られるまで繰り返される。このテーブルがセットアップされると、ユーザは、すべてのテーブルを１つのスクリーンで見ることができ、レポートエンジンコンポーネント３４０は、テーブルカラム間の関係をビジュアル化する。さらに、ユーザは、レポート更新頻度を設定することができるため、レポートエンジンコンポーネント３４０は、構成データ要素が更新される度に更新を行う必要はない。

【０１１７】

好ましい実施形態を図示して説明したが、当業者であれば、本発明の内容からから離れることなく、さまざまな変更を施すことが可能である。したがって、上述した本発明の改良および変更、さらいはその均等物は、本発明の技術的範囲に含まれ、添付の請求の範囲に含まれると理解される。

【０１１８】

［付属資料］

１．データアップロード用ＡＰＩ

URL

https://www.beulahworks.com/dm/v1/data_upload

POST Request Required Fields

```
POST /dm/v1/data_uploadHTTP/1.1

Content-Type: text/csv
Content-Length: 3876502
Charset: utf-8
Accept-Charset: utf-8
Host: www.beulahworks.com:1234 (configurable)
Filename: “abc.arff”
{Data File}
```

コンテンツタイプのフィールドは、ＣＳＶおよびＡＲＦＦを含む、ＭＩＭＥタイプのすべてのサポートされたデータファイルフォーマットを有するべきである。

```
CSV: text/csv
ARFF: application/vnd.arff (Custom MIMEtype; can be set in web server)
```

データのサイズは無制限であり、ウェブサーバの構成ファイルにおいて設定することが可能である。

POST Response Required Fields

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8

{Response JSON}

Response JSON Schema
{
   "type": "object",
   "$schema": "http://json-schema.org/draft-03/schema",
   "required": true,
   "properties": {
```

```
        "statusCode": {
            "type": "string",
            "required": true
         },
        "statusDescription": {
            "type": "string",
            "required": true
         },
        "status": {
            "type": "string",
            "required": true,
            "enum": [
                "success",
                "failure"
            ]
         },
        "transactionId": {
            "type": "string",
            "required": true
         },
        "transactionTime": {
            "type": "string",
            "required": true
         },
        "datasetId": {
            "type": "string",
            "required": true
         }
    }
}
```

Response JSON Example

```
{
   "status": "success",
    "statusCode": "0",
    "statusDescription": "Success",
    "transactionTime": "2013-12-10T03:08:23:63Z",
    "transactionId":"241b9632-ebfb-4be2-9d6d-64910f995182",
    "datasetId":"FBADDC8E-4007-4901-9CBF-328318E83DC5",
}
```

「datasetId 」は、以下のＡＰＩメソッドにおいて使用される。

【０１１９】

２．トレーニングＡＰＩ

Ａ．分類トレーニング

URL

https://www.beulahworks.com/dm/v1/classification_train

POST Request Required Fields

Same as https://www.beulahworks.com/dm/v1/classification_train.

Request JSON Schema

```
{
	"type":"object",
	"$schema":"http://json-schema.org/draft-03/schema",
	"id":"http://jsonschema.net",
	"required":false,
	"properties":{
		"algorithm":{
			"type":"array",
			"id":"http://jsonschema.net/algorithm",
			"required":false,
			"items":
				{
					"type":"object",
					"id":"http://jsonschema.net/algorithm/0",
					"required":false,
					"properties":{
						"name":{
							"type":"string",
							"id":"http://jsonschema.net/algorithm/
0/name",
							"required":false
						},
						"options":{
							"type":"object",
							"id":"http://jsonschema.net/algorithm/
0/options",
							"required":false,
							"properties":{
								"prune":{
									"type":"boolean",
									"id":"http://jsonschema.ne
t/algorithm/0/options/prune",
									"required":false
								}
							}
						}
					}
				}

		},
		"classAttributeName": {
			"type":"string",
			"id":"http://jsonschema.net/classAttributeName",
			"required":false
		},
		"datasetId": {
			"type":"string",
			"id":"http://jsonschema.net/datasetId",
```

```
                "required":true
        },
        "modelName": {
                "type":"string",
                "id":"http://jsonschema.net/modelName",
                "required":true
        },
        "preprocessor":{
                "type":"array",
                "id":"http://jsonschema.net/preprocessor",
                "required":false,
                "items":
                        {
                                "type":"object",
                                "id":"http://jsonschema.net/preprocessor/0",
                                "required":false,
                                "properties":{
                                        "name":{
                                                "type":"string",
                                                "id":"http://jsonschema.net/preprocess
or/0/name",
                                                "required":false
                                        },
                                        "options":{
                                                "type":"object",
                                                "id":"http://jsonschema.net/preprocess
or/0/options",
                                                "required":false,
                                                "properties":{
                                                        "removeAttriIndex": {
                                                                "type":"number",
                                                                "id":"http://jsonschema.ne
t/preprocessor/0/options/removeAttriIndex",
                                                                "required":false
                                                        }
                                                }
                                        },
                                        "type":{
                                                "type":"string",
                                                "id":"http://jsonschema.net/preprocess
or/0/type",
                                                "required":false
                                        }
                                }
                        }


        }
    }
}
```

Request JSON Example

```
{
    "datasetId":"FBADDC8E-4007-4901-9CBF-328318E83DC5",
   "preprocessor": [
        {
           "name": "Remove",
           "type": "filter",
           "options": {
               "removeAttriIndex": 2
           }
        }
    ],
   "algorithm": [
        {
           "name": "J48",
           "options": {
               "prune": false
           }
        }
    ],
    "classAttributeName": "Gender",
    "modelName": "GenderPredictor"
}
```

「classAttributeName」は、データセットがＡＲＦＦファイルとしてアップロードされた場合には必要とされず、「algorithm」と「preprocessor」は、自動データマイニングには必要とされず、「algorithm」と「preprocessor」はアレイのタイプである（つまり、ＡＰＩは、マルチプリプロセッサおよびアルゴリズムをサポートする）。マルチプリプロセッサが特定された場合には、これらのすべてがデータセットに適用される。マルチアルゴリズムが特定された場合には、そのアルゴリズムは別々にデータセットに適用され、平均化された結果が報告される。

POST Response Required Fields

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8

{Response JSON}
```

Response JSON Schema

```
{
   "type": "object",
   "$schema": "http://json-schema.org/draft-03/schema",
   "required": true,
   "properties": {
       "statusCode": {
           "type": "string",
           "required": true
        },
       "statusDescription": {
           "type": "string",
```

```
            "required": true
        },
        "status": {
            "type": "string",
            "required": true,
            "enum": [
                "success",
                "failure"
            ]
        },
        "transactionId": {
            "type": "string",
            "required": true
        },
        "transactionTime": {
            "type": "string",
            "required": true
        },
        "jobId": {
            "type": "string",
             "required":true
        }
    }
}


Response JSON Example
{
   "status": "success",
    "statusCode": "0",
    "statusDescription": "Success",
    "transactionTime": "2013-12-10T03:08:23:63Z",
    "transactionId": "241b9632-ebfb-4be2-9d6d-64910f995182",
    "jobId":"FBADDC8E-4007-4901-9CBF-328318E83DC5"
}
```

「statusCode 」と「statusDescription 」が予め定義された標準サクセス／エラーメッセージのセットである場合、「transaction Time」は、ＡＰＩメソッドが応答を発行する時間であるＵＴＣ時間であり、「transactionID」は、ロギングおよびパーティション分割の目的に使用されるＵＵＩＤであり、「jobId」は、特定のジョブの時間見積を確認するために他のＡＰＩメソッドによって使用される。

【０１２０】

Ｂ．クラスタリングトレーニング

URL

https://www.beulahworks.com/dm/v1/ clustering_train

POST Request Required Fields

Same as https://www.beulahworks.com/dm/v1/classification_train.

Request JSON Schema

```
{
```

```
"type":"object",
"$schema":"http://json-schema.org/draft-03/schema",
"id":"http://jsonschema.net",
"required":false,
"properties":{
      "algorithm":{
            "type":"array",
            "id":"http://jsonschema.net/algorithm",
            "required":false,
            "items":
                  {
                        "type":"object",
                        "id":"http://jsonschema.net/algorithm/0",
                        "required":false,
                        "properties":{
                              "name":{
                                    "type":"string",
                                    "id":"http://jsonschema.net/algorithm/
0/name",
                                    "required":false
                              },
                              "options":{
                                    "type":"object",
                                    "id":"http://jsonschema.net/algorithm/
0/options",
                                    "required":false,
                                    "properties":{
                                          "numClusters": {
                                                "type":"number",
                                                "id":"http://jsonschema.ne
t/algorithm/0/options/numClusters",
                                                "required":false
                                          }
                                    }
                              }
                        }
                  }

      },
      "datasetId": {
            "type":"string",
            "id":"http://jsonschema.net/datasetId",
            "required":true
      },
      "preprocessor":{
            "type":"array",
            "id":"http://jsonschema.net/preprocessor",
            "required":false,
            "items":
```

```
                    {
                        "type":"object",
                        "id":"http://jsonschema.net/preprocessor/0",
                        "required":false,
                        "properties":{
                            "name":{
                                "type":"string",
                                "id":"http://jsonschema.net/preprocess
or/0/name",
                                "required":false
                            },
                            "options":{
                                "type":"object",
                                "id":"http://jsonschema.net/preprocess
or/0/options",
                                "required":false,
                                "properties":{
                                    "removeAttriIndex": {
                                        "type":"number",
                                        "id":"http://jsonschema.ne
t/preprocessor/0/options/removeAttriIndex",
                                        "required":false
                                    }
                                }
                            },
                            "type":{
                                "type":"string",
                                "id":"http://jsonschema.net/preprocess
or/0/type",
                                "required":false
                            }
                        }
                    }


            }
        }
}

Request JSON Example
{
    "datasetId":"FBADDC8E-4007-4901-9CBF-328318E83DC5",
   "preprocessor": [
        {
           "name": "Remove",
           "type": "filter",
           "options": {
               "removeAttriIndex": 2
           }
        }
```

```
    ],
    "algorithm":[
        {
           "name": "K-Means",
           "options": {
               "numClusters": 5
           }
        }
    ]
}
```

Response POST Required Fields
Same as https://www.beulahworks.com/dm/v1/classification_train.

Response JSON Schema
Same as https://www.beulahworks.com/dm/v1/classification_train.

Response JSON Example
Same as https://www.beulahworks.com/dm/v1/classification_train.

【０１２１】

Ｃ．結合規則発見トレーニング

URL
https://www.beulahworks.com/dm/v1/association_rule_train

POST Request Required Fields
Same as https://www.beulahworks.com/dm/v1/classification_train.

Request JSON Schema

```
{
      "type":"object",
      "$schema":"http://json-schema.org/draft-03/schema",
      "id":"http://jsonschema.net",
      "required":false,
      "properties":{
            "algorithm":{
                  "type":"array",
                  "id":"http://jsonschema.net/algorithm",
                  "required":false,
                  "items":
                        {
                              "type":"object",
                              "id":"http://jsonschema.net/algorithm/0",
                              "required":false,
                              "properties":{
                                    "name":{
                                          "type":"string",
                                          "id":"http://jsonschema.net/algorithm/
0/name",
                                          "required":false
                                    },
```

```
                        "options":{
                              "type":"object",
                              "id":"http://jsonschema.net/algorithm/
0/options",
                              "required":false,
                              "properties":{
                                    "numRules": {
                                          "type":"number",
                                          "id":"http://jsonschema.ne
t/algorithm/0/options/numRules",
                                          "required":false
                                    }
                              }
                        }
                  }


      },
      "datasetId": {
            "type":"string",
            "id":"http://jsonschema.net/datasetId",
            "required":true
      },
      "preprocessor":{
            "type":"array",
            "id":"http://jsonschema.net/preprocessor",
            "required":false,
            "items":
                  {
                        "type":"object",
                        "id":"http://jsonschema.net/preprocessor/0",
                        "required":false,
                        "properties":{
                              "name":{
                                    "type":"string",
                                    "id":"http://jsonschema.net/preprocess
or/0/name",
                                    "required":false
                              },
                              "options":{
                                    "type":"object",
                                    "id":"http://jsonschema.net/preprocess
or/0/options",
                                    "required":false,
                                    "properties":{
                                          "removeAttriIndex": {
                                                "type":"number",
                                                "id":"http://jsonschema.ne
t/preprocessor/0/options/removeAttriIndex",
```

```
                                                "required":false
                                            }
                                        }
                                    },
                                    "type":{
                                        "type":"string",
                                        "id":"http://jsonschema.net/preprocess
or/0/type",
                                        "required":false
                                    }
                                }
                            }


                }
            }
}

Request JSON Example
{
    "datasetId":"FBADDC8E-4007-4901-9CBF-328318E83DC5",
   "preprocessor": [
        {
           "name": "Remove",
           "type": "filter",
           "options": {
               "removeAttriIndex": 2
           }
        }
    ],
   "algorithm": [
        {
           "name": "Apriori",
           "options": {
               "numRules": 10
           }
        }
    ]
}
```


Response POST Required Fields
Same as https://www.beulahworks.com/dm/v1/classification_train.

Response JSON Schema
Same as https://www.beulahworks.com/dm/v1/classification_train.

Response JSON Example
Same as https://www.beulahworks.com/dm/v1/classification_train.

【０１２２】

Ｅ．回帰（予測）トレーニング

URL
https://www.beulahworks.com/dm/v1/regression_train

POST Request Required Fields
Same as https://www.beulahworks.com/dm/v1/classification_train..

Request JSON Schema
Same as https://www.beulahworks.com/dm/v1/classification_train.

Request JSON Example
Same as https://www.beulahworks.com/dm/v1/classification_train.

Response POST Required Fields
Same as https://www.beulahworks.com/dm/v1/classification_train.

Response JSON Schema
Same as https://www.beulahworks.com/dm/v1/classification_train.

Response JSON Example
Same as https://www.beulahworks.com/dm/v1/classification_train.

【０１２３】
Ｆ．予測時間
URL
https://www.beulahworks.com/dm/v1/estimate_time

POST Request Required Fields
Same as https://www.beulahworks.com/dm/v1/classification_train.

Request JSON Schema

```
{
      "type":"object",
      "$schema":"http://json-schema.org/draft-03/schema",
      "required":true,
      "properties":{
            "jobId": {
                  "type":"string",
                  "required":true
            }
      }
}
```

Request JSON Example

```
{
    "jobId": "FBADDC8E-4007-4901-9CBF-328318E83DC5"
}
```

Response POST Required Fields
Same as https://www.beulahworks.com/dm/v1/classification_train.

Response JSON Schema

```
{
   "type": "object",
   "$schema": "http://json-schema.org/draft-03/schema",
   "id": "http://jsonschema.net",
   "required": true,
   "properties": {
       "estimatedFinishDate": {
           "type": "string",
           "id": "http://jsonschema.net/estimatedFinishDate",
           "required": true
        },
       "estimatedTime": {
           "type":"string",
           "id": "http://jsonschema.net/estimatedTime",
           "required": true
        },
       "jobId": {
           "type": "string",
           "id": "http://jsonschema.net/jobId",
           "required": true
        },
       "statusCode": {
           "type": "string",
           "id": "http://jsonschema.net/statusCode",
           "required": true
        },
       "statusDescription": {
           "type": "string",
           "id": "http://jsonschema.net/statusDescription",
           "required": true
        },
       "status": {
           "type": "string",
           "id": "http://jsonschema.net/status",
           "required": true,
           "enum": [
               "success",
               "failure"
           ]
        },
       "transactionID": {
           "type": "string",
           "id": "http://jsonschema.net/transactionID",
           "required": true
        },
       "transactionTime": {
           "type": "string",
           "id": "http://jsonschema.net/transactionTime",
           "required": true
        }
```

```
    }
}

Response JSON Example
{
   "status": "success",
    "statusCode": "0",
    "statusDescription": "Success",
    "jobId":"FBADDC8E-4007-4901-9CBF-328318E83DC5",
    "estimatedTime": "1 hour 30 minutes",
    "estimatedFinishDate":"2013-12-10T04:38:23:63Z",
    "transactionTime": "2013-12-10T03: 08: 23:63Z",
    "transactionID":"241b9632-ebfb-4be2-9d6d-64910f995182"
}
```

https://www.beulahworks.com/dm/v1/classification_train.sにおける同一のフィールドの他にも、エラーが発生しなかった場合には、「jobId」は、見積もられたジョブのコンファメーションであり、「estimatedTime」は、選択されたジョブにかかる時間の見積を示し、「estimatedFinishDate」は、選択されたジョブが完了するデータおよび時間の見積を示す。

【０１２４】

Ｇ．コールバックＰＯＳＴ

URL

https://www.beulahworks.com/dm/v1/callback

```
POST Request Required Fields
POST callback_url(configurable) HTTP/1.1

Content-Type: application/json
Charset: utf-8
Accept-Charset: utf-8
Host: callback_host (configurable)
{Request JSON}

Request JSON Schema
{
   "type": "object",
   "$schema": "http://json-schema.org/draft-03/schema",
   "required": true,
   "properties": {
       "dataPreparationInfo": {
           "type": "object",
           "required": false,
           "properties": {
               "mode": {
                   "type": "string",
                   "required": true
               },
               "schemes": {
                   "type": "object",
```

```
            "required": true,
            "properties": {
                "nullDataHandling": {
                    "type": "string",
                    "required": true
                },
                "outlierRemoval": {
                    "type": "string",
                    "required": true
                }
            }
        }
    }
},
"jobId": {
    "type": "string",
    "required": true
},
"modelName": {
    "type": "string",
    "required": true
},
"statusCode": {
    "type": "string",
    "required": true
},
"statusDescription": {
    "type": "string",
    "required": true
},
"status": {
    "type": "string",
    "required": true,
    "enum": [
        "success",
        "failure"
    ]
},
"trainingInfo": {
    "type": "object",
    "required": true,
    "properties": {
        "attributeNum": {
            "type":"string",
            "required": true
        },
        "attributes": {
            "type": "array",
            "required": true,
            "items": {
```

```
                    "type": "string",
                    "required": false
                }
            },
            "correctlyClassifiedInstancePercentage":{
                "type": "string",
                "required": true
            },
            "correctlyClassifiedInstancesNum": {
                "type": "number",
                "required": true
            },
            "folds": {
                "type": "number",
                "required": false
            },
            "incorrectlyClassifiedInstanceNum":{
                "type": "number",
                "required": true
            },
            "incorrectlyClassifiedInstancePercentage":{
                "type": "string",
                 "required":true
            },
            "instanceNum": {
                "type": "string",
                "required": true
            },
            "scheme": {
                "type": "string",
                "required":true
            },
            "testMode": {
                "type": "string",
                "required": true
            }
        }
    }
}
```


Request JSON Example


```
{
    "jobId": "FBADDC8E-4007-4901-9CBF-328318E83DC5",
    "status": "success",
    "statusCode": "0",
    "statusDescription": "Success.",
    "modelName": "activeCustomer_classification",
    "dataPreparationInfo": {
       "mode": "automatic",
```

```
        "schemes": {
            "outlierRemoval": "Gaussiandistribution",
            "nullDataHandling": "Arithmatic mean"
         }
    },
    "trainingInfo": {
        "scheme": "weka.classifiers.rules.ZeroR",
        "instanceNum": "300",
        "attributeNum": "3",
        "attributes": [
            "764e2634",
            "852d7435",
            "279h0236"
         ],
        "testMode": "crossvalidation",
        "folds": 10,
        "correctlyClassifiedInstancesNum":250,
        "correctlyClassifiedInstancePercentage":"83.3333%",
        "incorrectlyClassifiedInstanceNum":50,
        "incorrectlyClassifiedInstancePercentage":"16.6667%"
    }
}
```


Response POST Required Fields
Same as https://www.beulahworks.com/dm/v1/classification_train.

Response JSON Schema


```
{
   "type": "object",
   "$schema":"http://json-schema.org/draft-03/schema",
   "required": true,
   "properties": {
        "statusCode": {
            "type": "string",
            "required": true
         },
        "statusDescription": {
            "type": "string",
             "required":true
         },
        "status": {
            "type": "string",
            "required": true,
            "enum": [
                "success",
                "failure"
            ]
         },
        "transactionID": {
            "type": "string",
```

```
            "required": true
        },
        "transactionTime": {
            "type": "string",
            "required": true
        },
        "jobId": {
            "type": "string",
            "required": false
        }
    }
}

Response JSON Example
{
   "status": "success",
    "statusCode": "0",
    "statusDescription": "Success",
    "transactionTime": "2013-12-10T03:08:23:63Z",
    "transactionID":"241b9632-ebfb-4be2-9d6d-64910f995182",
    "jobId": "FBADDC8E-4007-4901-9CBF-328318E83DC5"
}
```

【０１２５】

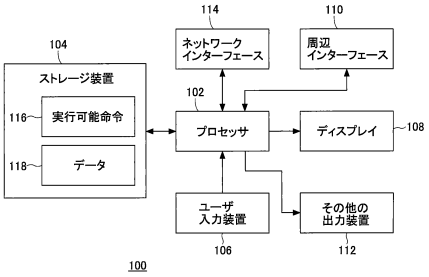
３．使用ＡＰＩ

使用ＡＰＩは、以下の点を除き、トレーニングＡＰＩと同一である。

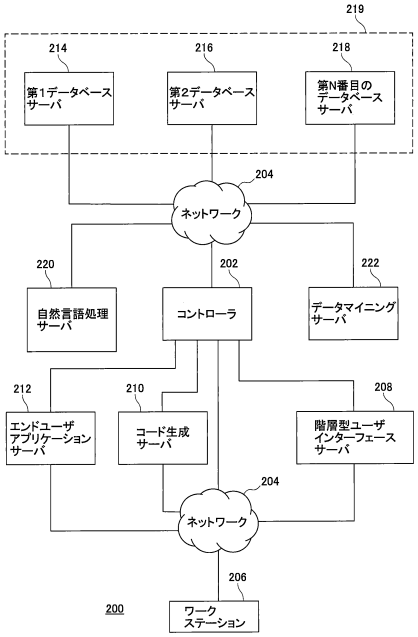
１．ＵＲＬが異なる。「train」は「use」に置き換えられ、たとえば、「https://www.beulahworks.com/dm/v1/classification_train」は、「https://www.beulahworks.com/dm/v1/classification_use」となる。他の使用ＡＰＩについても同様である。

２．「モデル」フィールドは任意である。「モデル」が提供されない場合には、システムは、タスクを実行するためにグランド機械学習モデルを使用する。

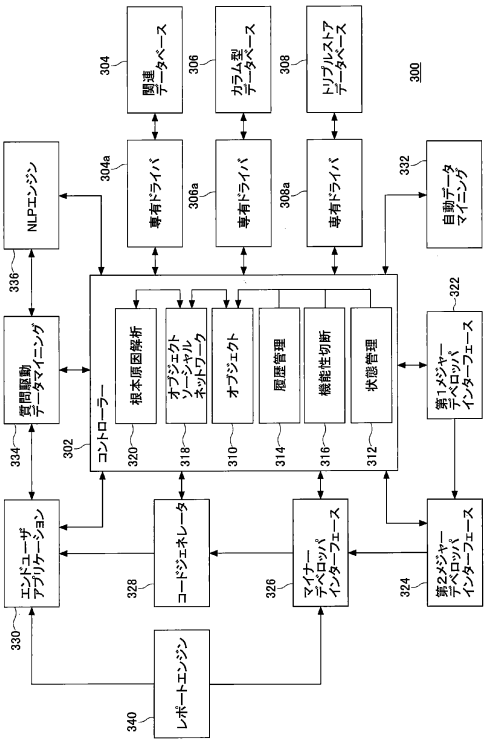
【図１】


ストレージ装置
実行可能命令
データ
プロセッサ
ディスプレイ
ネットワーク
インターフェース
周辺
インターフェース
ユーザ
入力装置
その他の
出力装置
100
102
104
106
108
110
112
114
116
118

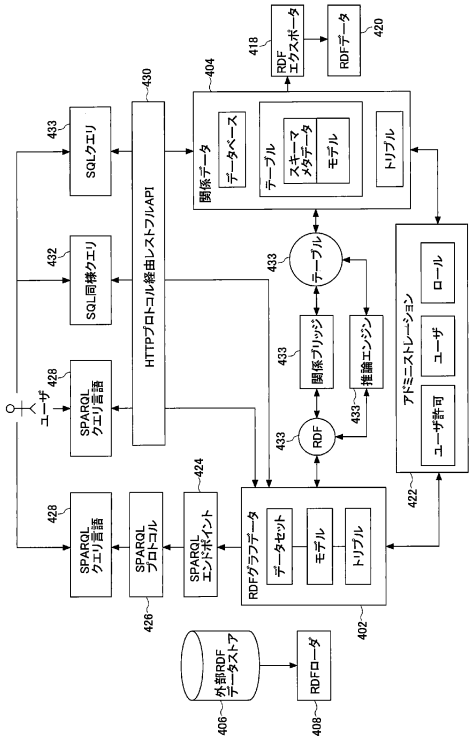
【図２】


第1データベース
サーバ
第2データベース
サーバ
第N番目の
データベース
サーバ
ネットワーク
自然言語処理
サーバ
コントローラ
データマイニング
サーバ
エンドユーザ
アプリケーション
サーバ
コード生成
サーバ
階層型ユーザ
インターフェース
サーバ
ネットワーク
ワーク
ステーション
200
202
204
206
208
210
212
214
216
218
219
220
222

【図３】


関連
データベース
カラム型
データベース
トリプルストア
データベース
NLPエンジン
専有ドライバ
専有ドライバ
専有ドライバ
自動データ
マイニング
質問駆動
データマイニング
コントローラー
根本原因解析
オブジェクト
ソーシャル
ネットワーク
オブジェクト
履歴管理
機能性切断
状態管理
第1メジャー
デベロッパ
インターフェース
エンドユーザ
アプリケーション
コードジェネレータ
マイナー
デベロッパ
インターフェース
第2メジャー
デベロッパ
インターフェース
レポートエンジン
300
302
304
304a
306
306a
308
308a
310
312
314
316
318
320
322
324
326
328
330
332
334
336
340

【図４】


RDF
エクスポータ
RDFデータ
関係データ
データベース
テーブル
スキーマ
メタデータ
モデル
トリプル
SQLクエリ
SQL同様クエリ
SPARQL
クエリ言語
HTTPプロトコル経由レストフルAPI
ユーザ
テーブル
関係プリッジ
推論エンジン
RDF
アドミニストレーション
ロール
ユーザ
ユーザ許可
SPARQL
クエリ言語
SPARQL
プロトコル
SPARQL
エンドポイント
RDFグラフデータ
データセット
モデル
トリプル
外部RDF
データストア
RDFローダ
402
404
406
408
418
420
422
424
426
428
430
432
433

フロントページの続き

(72)発明者　ジョンソン，ロベルト　イー．，ザ・サード
アメリカ合衆国，４６３０７，インディアナ州，クラウンポイント，ペインティッド　リーフ　コート，２７２２番地
(72)発明者　ヒルトン，ブレイン，ロバート
アメリカ合衆国，９７４０５，オレゴン州，ユージーン，ウエスト　トゥエンティーナインス　アベニュー，２０３５番地
(72)発明者　ネヴィル，デーヴィッド，アラン
アメリカ合衆国，４６３８５，インディアナ州，ヴァルパレーゾ，ジェイド　ブールバード，１５５３番地

審査官　田中　幸雄

(56)参考文献　特開２００６－２３６２９９（ＪＰ，Ａ）
特表２００６－５０３３５１（ＪＰ，Ａ）
特開２００８－１５２７３９（ＪＰ，Ａ）
特表２００９－５３１７９１（ＪＰ，Ａ）
中村勝一ほか，サービスプラットホームとしてのネットワーク計測の試み，電子情報通信学会技術研究報告，日本，社団法人電子情報通信学会，２００９年　9月１８日，Ｖｏｌ．１０９　Ｎｏ．２０８，４９－５４ページ

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１２／００
Ｇ０６Ｎ　５／０４