(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4526876号
(P4526876)
(45)発行日　平成22年8月18日(2010.8.18)　(24)登録日　平成22年6月11日(2010.6.11)

(51)Int.Cl.　　F I
G06F 12/00 (2006.01)　　G06F 12/00 513D
G06F 17/30 (2006.01)　　G06F 17/30 180D

請求項の数 24　(全 27 頁)

(21)出願番号　特願2004-169084 (P2004-169084)
(22)出願日　平成16年6月7日 (2004.6.7)
(65)公開番号　特開2004-362595 (P2004-362595A)
(43)公開日　平成16年12月24日 (2004.12.24)
審査請求日　平成19年6月7日 (2007.6.7)
(31)優先権主張番号　10/456,139
(32)優先日　平成15年6月6日 (2003.6.6)
(33)優先権主張国　米国 (US)

(73)特許権者　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　ドミトリー　ソンキン
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　オールド　レッドモンド　ロード　７２５０　ユニット　ケー－１４０

最終頁に続く

(54)【発明の名称】データベースオブジェクトスクリプト生成方法およびシステム

(57)【特許請求の範囲】
【請求項１】
リレーショナルデータベースにおいて少なくとも１つのオブジェクト参照からＳＱＬスクリプトを生成する方法であって、
コンピュータが、少なくとも１つのオブジェクト参照を受け取ることと、
コンピュータが、前記少なくとも１つのオブジェクト参照の関係依存性を検出することと、
コンピュータが、前記少なくとも１つのオブジェクト参照を含む階層オブジェクトツリーを構築することと、
コンピュータが、前記検出された関係依存性に基づいて前記階層オブジェクトツリーから依存性リストを導出することであって、前記依存性リストは、ユーザプログラム経由で編集可能であり、オブジェクトが生成されるべき順序を表す線形リストを備えることと、
コンピュータが、前記依存性リストに対応するＳＱＬスクリプトを生成することと
を備えることを特徴とする方法。
【請求項２】
前記構築することは、重複オブジェクト参照を除去し、非重複オブジェクト参照および関連メタデータを階層オブジェクトツリーに入力することを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】
前記構築することは、進行を示すイベントをトリガし、オプションのオブジェクト操作

を提供することを備えることを特徴とする請求項１に記載の方法。
【請求項４】
前記オプションのオブジェクト操作は、前記少なくとも１つのオブジェクト参照のフィルタリングを備え、ユーザーおよびプログラムのうち１つまたは複数によって選択されたオブジェクト参照を削除することができることを特徴とする請求項３に記載の方法。
【請求項５】
前記オプションのオブジェクト操作は、前記選択されたオブジェクト参照を除去し、該選択されたオブジェクト参照に依存するすべての後続の参照を除去することを備えることを特徴とする請求項４に記載の方法。
【請求項６】
前記依存性リストは、依存性制約を満たすためのオブジェクト生成順序として表現された線形リストを備えることを特徴とする請求項１に記載の方法。
【請求項７】
前記導出することは、オブジェクト参照の下層の依存性を再帰的にステップスルーし、オプションのオブジェクト操作を提供し、前記下層の依存性を削除することができることを特徴とする請求項１に記載の方法。
【請求項８】
前記生成することは、スクリプティングオプションが前記ＳＱＬスクリプトを修正できるようにすることを備えることを特徴とする請求項１に記載の方法。
【請求項９】
前記リレーショナルデータベースは、ＳＱＬデータベースであることを特徴とする請求項１に記載の方法。
【請求項１０】
前記関係依存性は、親－子、子－孫、および孫－曾孫のうち１つまたは複数であることを特徴とする請求項１に記載の方法。
【請求項１１】
前記少なくとも１つのオブジェクト参照は、サーバー／データベース／テーブルを備えるフォーマットを用いてユニフォームリソース名を備えることを特徴とする請求項１に記載の方法。
【請求項１２】
進行監視およびオブジェクト操作のうち少なくとも１つを許可する１つまたは複数のイベントをトリガすること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１３】
オブジェクト操作は、オブジェクト参照の追加、修正および削除のうち少なくとも１つを備えることを特徴とする請求項１２に記載の方法。
【請求項１４】
リレーショナルデータベースにおいて１つまたは複数のオブジェクト参照からＳＱＬスクリプトを生成する方法を実行するための、コンピュータ実行可能命令を有するコンピュータ可読記憶媒体であって、前記方法は、
少なくとも１つのオブジェクト参照を受け取ることと、
前記少なくとも１つのオブジェクト参照の関係依存性を検出することと、
前記少なくとも１つのオブジェクト参照を含む階層オブジェクトツリーを構築することと、
前記検出された関係依存性に基づいて前記階層オブジェクトツリーから依存性リストを導出することであって、前記依存性リストは、ユーザプログラム経由で編集可能であり、オブジェクトが生成されるべき順序を表す線形リストを備えることと、
前記依存性リストに対応するＳＱＬスクリプトを生成することと
を備えたことを特徴とするコンピュータ可読記憶媒体。
【請求項１５】
前記構築することは、重複オブジェクト参照を除去し、非重複オブジェクト参照および

関連メタデータを階層オブジェクトツリーに入力することを備えたことを特徴とする請求項１４に記載のコンピュータ可読記憶媒体。
【請求項１６】
前記構築することは、進行を示すイベントをトリガし、オプションのオブジェクト操作を提供することを備えたことを特徴とする請求項１４に記載のコンピュータ可読記憶媒体。
【請求項１７】
前記オプションのオブジェクト操作は、前記少なくとも１つのオブジェクト参照のフィルタリングを備え、前記オブジェクト参照を削除することができることを特徴とする請求項１６に記載のコンピュータ可読記憶媒体。
【請求項１８】
前記オプションのオブジェクト操作は、オブジェクト参照を除去し、該除去されたオブジェクト参照に依存するすべての後続の参照の除去を備えたことを特徴とする請求項１７に記載のコンピュータ可読記憶媒体。
【請求項１９】
前記依存性リストは、前記少なくとも１つのオブジェクト参照が依存性制約を満たすように生成される順序で表された線形リストを備えたことを特徴とする請求項１４に記載のコンピュータ可読記憶媒体。
【請求項２０】
前記導出することは、オブジェクト参照の下層の依存性を再帰的にステップスルーし、オプションのオブジェクト操作を提供し、前記下層の依存性を削除することができることを特徴とする請求項１４に記載のコンピュータ可読記憶媒体。
【請求項２１】
前記生成することは、スクリプティングオプションが前記ＳＱＬスクリプトを修正することを可能にすることを備えたことを特徴とする請求項１４に記載のコンピュータ可読記憶媒体。
【請求項２２】
前記少なくとも１つのオブジェクト参照は、サーバー／データベース／テーブルを備えるフォーマットを用いてユニフォームリソース名を備えたことを特徴とする請求項１４に記載のコンピュータ可読記憶媒体。
【請求項２３】
リレーショナルデータベースにおけるオブジェクト参照からＳＱＬスクリプトを生成するためのコンピュータシステムであって、
少なくとも１つのオブジェクト参照を受け取るための入力デバイスと、
プロセッサであって、前記スクリプトを生成するためのコンピュータ命令が実行されて、
前記オブジェクト参照の関係依存性を検出する動作と、
前記オブジェクト参照を含む階層オブジェクトツリーを構築する動作と、
前記検出された関係依存性に基づいて前記階層オブジェクトツリーから依存性リストを導出する動作であって、前記依存性リストは、ユーザプログラム経由で編集可能であり、オブジェクトが生成されるべき順序を表す線形リストを備える動作と、
前記依存性リストに対応する前記ＳＱＬスクリプトを生成する動作と
を実行するプロセッサと、
前記スクリプトを、表示デバイス、後続のコンピュータプログラムおよびストレージデバイスのうち１つまたは複数に渡すための通信ポートと
を備えたことを特徴とするコンピュータシステム。
【請求項２４】
前記オブジェクト参照は、サーバー／データベース／テーブルを備えるフォーマットを有するユニフォームリソース名を備えたことを特徴とする請求項２３に記載のコンピュータシステム。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般に、コンピュータデータベースの分野に関し、より詳細には、リレーショナルデータベースにおける複雑なオブジェクトのための自動スクリプト生成の分野に関する。
【背景技術】
【０００２】
ＳＱＬ（ｓｔｒｕｃｔｕｒｅｄ　ｑｕｅｒｙ　ｌａｎｇｕａｇｅ）は、リレーショナルデータベースと通信するために使用される米国規格協会（ＡＮＳＩ）規格である。ＳＱＬは、リレーショナルデータベース管理システムのための標準言語である。ＳＱＬステートメントは、リレーショナルデータベースのデータ更新またはデータ検索などのタスクを実行するために使用される。多くのデータベースシステムがＳＱＬを使用しているが、これらシステムの多くがそれら独自の追加の拡張を有しており、これらは通常そのシステム上でのみ使用されている。しかし、標準ＳＱＬコマンドである「Ｓｅｌｅｃｔ」、「Ｉｎｓｅｒｔ」、「Ｕｐｄａｔｅ」、「Ｄｅｌｅｔｅ」、「Ｃｒｅａｔｅ」および「Ｄｒｏｐ」などを使用して、リレーショナルデータベースで行ないたいと望む多くのことを達成することができる。
【発明の開示】
【発明が解決しようとする課題】
【０００３】
リレーショナルデータベースは一般に、行および列を含むテーブルとして構造化される。任意の行および列の挿入のためのデータエントリ、またはセルのためのデータエントリは通常、そのセル内で許容されているデータ型を定義するために使用されている１組のデータ制約に適合している。このようなデータエントリでの１つの古くからの問題は、データベースに配置することが望まれる、大きく複雑なオブジェクト（ｃｏｍｐｌｅｘ　ｏｂｊｅｃｔ）についての定義の欠如である。利用されるＳＱＬデータ型に関する制約は一般に、ＳＱＬなどのリレーショナルデータベースに入力することができるデータの種類を制限している。大きいオブジェクトは、ＳＱＬデータベースに配置された場合、データベース処理の速度を低下させるか、１つまたは複数のかさばる複雑なオブジェクトに対処するために、メモリなどのシステムリソースや処理時間を使用することがある。
【０００４】
ＳＱＬなどのリレーショナルデータベースにおけるクエリのスクリプティングおよびオブジェクトの利用には、時間がかかり、専門知識集約的になることがある。このようなスクリプトの作成者は、他のオブジェクトに関連して新規の複雑なオブジェクトの依存性を理解しなければならず、これは、こうしたオブジェクトを、適切なメタデータと共にターゲットのリレーショナルデータベース上に正しくインスタンス化するためである。このスクリプティングの専門知識は、例えば、自身のリレーショナルデータベースを自身のビジネスに関連した複雑なオブジェクトの追跡および検索のために利用したいと望む平均的なユーザーの経験を超えている場合がある。あるいはまた、システムマネージャは、リレーショナルデータベースの保守を支援するためにスクリプティングを使用することがある。このスクリプティングアクティビティには、準備および実行のための時間と配慮を要する。複雑なオブジェクトに関してリレーショナルデータベースのためにスクリプトを自動生成することは、現在、容易に達成することはできない。
【０００５】
したがって、ＳＱＬデータベースにおいて参照され、処置されることが望まれるオブジェクトのための統一表現に対する必要性がある。加えて、アプリケーションおよびシステムの保守作業のために、スクリプトを生成して、オブジェクトをＳＱＬなどのリレーショナルデータベースに配備することを容易にするためのメカニズムに対する必要性がある。本発明は、複雑な依存性ツリーおよびリストを作成し、リレーショナルデータベースメタ

データ構造の複雑な知識を必要とすることなくスクリプトを生成するためにこれらツリーおよびリストを修正する様々なシステム、方法および技術により、前述の必要性に対処し、これらの問題を解決する。

【課題を解決するための手段】

【０００６】

本発明は、特にＳＱＬデータベース管理システムに適用することができるリレーショナルデータベースのためのスクリプタを含む。

【０００７】

複数の独立したソフトウェアモジュールまたは複数の機能の連結がオブジェクト参照を入力し、スクリプトを出力する。本発明の例示的なフェーズまたはモジュールでは、渡された１つまたは複数のオブジェクト参照から階層オブジェクトツリーを作成する。オブジェクト参照を使用する複雑なオブジェクト（ｃｏｍｐｌｅｘ　ｏｂｊｅｃｔ）は、ユニフォームリソース名により表現することができる。モジュールは重複したオブジェクト参照を除去し、依存性ツリーを生成する。モジュールはまた、ツリーが作成されているとき、ならびに完成後に、ツリーを編集するための機会も提供する。

【０００８】

別の例示的なモジュールまたはフェーズでは、ユーザーまたは以前のモジュールから階層依存性ツリーを入力し、依存性リストを生成する。依存性リストは作成の順序を表す線形リストであり、オブジェクトが依存性制約を満たすために使用することができる。このモジュールはまた、依存性リストが作成されているとき、ならびに完成後に、依存性リストを編集するための機会も提供する。

【０００９】

別の例示的なモジュールまたはフェーズでは、依存性リストからスクリプトを生成する。依存性リストは、ユーザーにより生成されるか、または以前のモジュールから入力されることがある。このモジュールは、オブジェクトを依存性リスト上でインスタンス化し、オブジェクトに対応するスクリプティングメソッドをコールする。このモジュールは、スクリプトが生成されているとき、ならびに完成後に、スクリプトを編集するための幅広い範囲の柔軟性をユーザーまたはコントロールプログラムに提供する。

【００１０】

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、添付の図面と共に読むとよりよく理解される。本発明を例示する目的で、図面にて本発明の例示的な構成を示すが、本発明は、開示した特定の方法および手段に限定されない。

【発明を実施するための最良の形態】

【００１１】

（概要）

本発明は、リレーショナルデータベースにおける様々な複雑さのオブジェクトの自動スクリプティングを提供にする。依存性リストがプロセスに入力され、依存性または階層ツリーが生成され、様々なオブジェクトの関係が反映される技術を提供する。次に、依存性または階層ツリーをプロセスの別の部分に入力することができ、階層ツリーが順序化された依存性リストに加工される。次に、依存性リストをスクリプトに加工することができ、このスクリプトを使用してリレーショナルデータベースオブジェクトを所与のターゲットデータベースに配備することができる。本発明の様々なフェーズまたはモジュールを別々に操作するか、またはタンデムに操作することができる。

【００１２】

（例示的なコンピューティングデバイス）

図１および以下の考察は、本発明を実施することができる適切なコンピューティング環境の簡単な全体的説明を提供するためのものである。しかし、ハンドヘルド、ポータブルおよび他のコンピューティングデバイス、ならびに、すべての種類のコンピューティングオブジェクトが、本発明に関連して使用するために企図されていることを理解されたい。したがって、汎用コンピュータが以下で説明されているが、これは一例であり、本発明は

、ネットワーク／バス相互運用性およびインタラクションを有するクライアントのような、他のコンピューティングデバイスにより実施することができる。このように、本発明は、ほんの少しまたは最小限のクライアントリソースが関係するネットワーク化されたホストサービスの環境において実施することができ、これは例えば、クライアントデバイスが単に、アプライアンス内に配置されたオブジェクト、または他のコンピューティングデバイスおよびオブジェクトなど、ネットワーク／バスへのインターフェースとしての機能を果たすネットワーク環境である。本質的には、データを格納することができるいかなる場所、または、そこからデータを取り出すことができるものは、本発明による操作のために望ましいか、あるいは適した環境である。

【００１３】

必須ではないが、本発明は、デバイスまたはオブジェクトについての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施することができ、そして／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内に含めることができる。ソフトウェアは一般に、クライアントワークステーション、サーバーまたは他のデバイスなど、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のようなコンピュータ実行可能命令の一般的なコンテキストにおいて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、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、これらは特定のタスクを実行するか、あるいは特定の抽象データ型を実装する。通常、プログラムモジュールの機能は、様々な実施形態において要望に応じて組み合わせたり、分散させたりすることができる。また、当業者は、本発明が他のコンピュータ構成により実施できることを理解するであろう。本発明と共に使用するのに適している可能性のある他の周知のコンピューティングシステム、環境および／または構成には、これらに限定されないが、パーソナルコンピュータ（ＰＣ）、現金自動預け払い機、サーバーコンピュータ、ハンドヘルドもしくはラップトップデバイス、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、プログラム可能な民生用電子機器、ネットワークＰＣ、アプライアンス、ライト、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。また、本発明は分散コンピューティング環境において実施することもでき、この環境ではタスクが通信ネットワーク／バスまたは他のデータ伝送媒体を通じてリンクされるリモート処理デバイスによって実行される。分散コンピューティング環境では、プログラムモジュールは、メモリストレージデバイスを含んでいるローカルおよびリモートのコンピュータストレージメディアに位置することができ、クライアントノードがサーバーノードとして振る舞うことがある。

【００１４】

このように、図１は、本発明を実施することができる適切なコンピューティングシステム環境１００の一例を例示しているが、上記で明らかにしたように、コンピューティングシステム環境１００は、適切なコンピューティング環境の一例でしかなく、本発明の使用または機能の範囲についていかなる限定も示唆するものではない。コンピューティング環境１００は、例示的オペレーティング環境１００において例示したコンポーネントのいずれか１つまたは組み合わせに関していかなる依存性または要件を有するものとしても解釈されるべきではない。

【００１５】

図１を参照すると、本発明を実施するための例示的システムは、コンピュータシステム１１０の形態の汎用コンピューティングデバイスを含んでいる。コンピュータシステム１１０のコンポーネントは、これらに限定されないが、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を含む。システムバス１２１はいくつかのタイプのバス構造のいずれとしてもよく、これには任意の様々なバスアーキテクチャを使用するメモリバスもしくはメモリコントローラ、周辺バスおよびローカルバスが含まれる。限定ではなく、例として、このようなアーキテクチャには、ＩＳＡ（Ｉｎｄｕｓｔｒｙ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＭＣＡ（Ｍｉｃｒｏ Ｃｈａｎｎｅｌ Ａｒｃｈ

ｉｔｅｃｔｕｒｅ）バス、拡張ＩＳＡ（Ｅｎｈａｎｃｅｄ　ＩＳＡ）バス、ＶＥＳＡ（Ｖｉｄｅｏ　Ｅｌｅｃｔｒｏｎｉｃｓ　Ｓｔａｎｄａｒｄｓ　Ａｓｓｏｃｉａｔｉｏｎ）ローカルバス、およびＰＣＩ（Ｐｅｒｉｐｈｅｒａｌ　Ｃｏｍｐｏｎｅｎｔ　Ｉｎｔｅｒｃｏｎｎｅｃｔ）バス（メザニンバスとしても知られる）が含まれる。

【００１６】

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によってアクセスすることができる任意の利用可能なメディアとすることができ、これには、揮発性および不揮発性のメディア、リムーバブルおよび非リムーバブルのメディアが含まれる。限定ではなく、例として、コンピュータ可読媒体は、コンピュータストレージメディアおよび通信メディアを含むことができる。コンピュータストレージメディアは、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他のデータなど、情報の格納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、リムーバブルおよび非リムーバブルのメディアを含む。コンピュータストレージメディアには、これらに限定されないが、ＲＡＭ（Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ）、ＲＯＭ（Ｒｅａｄ－Ｏｎｌｙ　Ｍｅｍｏｒｙ）、ＥＥＰＲＯＭ（Ｅｌｅｃｔｒｉｃａｌｌｙ　Ｅｒａｓａｂｌｅ　Ｐｒｏｇｒａｍｍａｂｌｅ　Ｒｅａｄ　Ｏｎｌｙ　Ｍｅｍｏｒｙ）、フラッシュメモリもしくは他のメモリ技術、ＣＤＲＯＭ（Ｃｏｍｐａｃｔ　Ｒｅａｄ　Ｏｎｌｙ　Ｍｅｍｏｒｙ）、再書込み可能なコンパクトディスク（ＣＤＲＷ）、デジタル多用途ディスク（ＤＶＤ）もしくは他の光ディスクストレージ、磁気カセット、磁気テープ、磁気ディスクストレージもしくは他の磁気ストレージデバイス、または、所望の情報を格納するために使用することができ、コンピュータシステム１１０によってアクセスすることができる他の任意の媒体が含まれる。通信メディアは通常、搬送波または他のトランスポートメカニズムなどの変調データ信号に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他のデータを具体化し、任意の情報配信メディアを含む。「変調データ信号」という用語は、信号に情報を符号化するような仕方でその特性の１つまたは複数が設定または変更されている信号を意味する。限定ではなく、例として、通信メディアには、有線ネットワークまたは直接配線接続などの有線メディア、ならびに、音響、ＲＦ、赤外線および他の無線メディアなどの無線メディアが含まれる。上記のいずれの組み合わせも、コンピュータ可読メディアの範囲内に含まれるべきである。

【００１７】

システムメモリ１３０は、読み取り専用メモリ（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のコンピュータストレージメディアを含む。基本入出力システム１３３（ＢＩＯＳ）は、起動中など、コンピュータシステム１１０内の要素間で情報を転送する助けとなる基本ルーチンを収容し、通常はＲＯＭ１３１に格納される。ＲＡＭ１３２は通常、処理装置１２０によって即時にアクセス可能および／または現在操作中であるデータおよび／またはプログラムモジュールを収容する。限定ではなく、例として、図１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、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を例示している。

【００１８】

コンピュータシステム１１０はまた、他のリムーバブル／非リムーバブル、揮発性／不揮発性のコンピュータストレージメディアを含むこともできる。単に例として、図１は、非リムーバブルの不揮発性磁気メディアに対する読み書きを行なうハードディスクドライブ１４１、リムーバブルの不揮発性磁気ディスク１５２に対する読み書きを行なう磁気ディスクドライブ１５１、および、ＣＤＲＯＭ、ＣＤＲＷ、ＤＶＤまたは他の光メディアなど、リムーバブルの不揮発性光ディスク１５６に対する読み書きを行なう光ディスクドライブ１５５を例示している。例示的オペレーティング環境で使用することができる他のリムーバブル／非リムーバブル、揮発性／不揮発性のコンピュータストレージメディアには、これらに限定されないが、磁気テープカセット、フラッシュメモリカード、デジタル多

用途ディスク、デジタルビデオテープ、半導体ＲＡＭ、半導体ＲＯＭなどが含まれる。ハードディスクドライブ１４１は通常、インターフェース１４０などの非リムーバブルメモリインターフェースを通じてシステムバス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、インターフェース１５０などのリムーバブルメモリインターフェースによってシステムバス１２１に接続される。

【００１９】

上述し、図１に例示したドライブおよびそれらの関連するコンピュータストレージメディアは、コンピュータシステム１１０のため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他のデータのストレージを提供する。図１では、例えば、ハードディスクドライブ１４１が、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、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を格納するものとして例示されている。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、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と同じものにすることも異なるものにすることもできることに留意されたい。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、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、最低でも、これらが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与えられている。ユーザーは、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などの入力デバイスを通じて、コマンドおよび情報をコンピュータシステム１１０へ入力することができる。他の入力デバイス（図示せず）には、マイクロフォン、ジョイスティック、ゲームパッド、衛星アンテナ、スキャナなどが含まれる可能性がある。これらおよび他の入力デバイスは、多くの場合、システムバス１２１に結合されるユーザー入力インターフェース１６０を通じて処理装置１２０へ接続されるが、パラレルポート、ゲームポートまたはユニバーサルシリアルバス（ＵＳＢ）など、他のインターフェースおよびバス構造によって接続することもできる。モニタ１９１または他のタイプの表示デバイスも、ビデオインターフェース１９０などのインターフェースを介してシステムバス１２１へ接続され、これはビデオメモリ（図示せず）と通信することもできる。モニタ１９１に加えて、コンピュータシステムはまた、出力周辺インターフェース１９５を通じて接続することができるスピーカ１９７およびプリンタ１９６などの他の周辺出力デバイスを含むこともできる。

【００２０】

コンピュータシステム１１０は、リモートコンピュータ１８０などの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または分散環境において動作することができる。リモートコンピュータ１８０は、パーソナルコンピュータ、サーバー、ルーター、ネットワークＰＣ、ピアデバイスまたは他の共通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に関連して上述した要素の多くまたはすべてを含むが、メモリストレージデバイス１８１のみを図１に例示した。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およびワイドエリアネットワーク（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。このようなネットワーキング環境は、家庭、オフィス、企業全体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において一般的である。

【００２１】

ＬＡＮネットワーキング環境において使用される場合、コンピュータシステム１１０は、ネットワークインターフェースまたはアダプタ１７０を通じてＬＡＮ１７１へ接続される。ＷＡＮネットワーキング環境において使用される場合、コンピュータシステム１１０は通常、インターネットなどのＷＡＮ１７３を介して通信を確立するためのモデム１７２または他の手段を含む。モデム１７２を内蔵または外付けとすることができ、ユーザー入力インターフェース１６０または他の適切なメカニズムを介してシステムバス１２１へ接続することができる。ネットワーク環境では、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を、リモートのメモリストレージデバイスに格

納することができる。限定ではなく、例として、図１はリモートアプリケーションプログラム１８５をメモリデバイス１８１上に存在するものとして例示している。図示したネットワーク接続は例示的であり、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できることが理解されよう。

【００２２】

様々な分散コンピューティングフレームワークが、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収束に照らして開発され、また開発されつつある。個人およびビジネスユーザーは同様に、アプリケーションおよびコンピューティングデバイスのためのシームレスに相互運用可能なウェブ対応のインターフェースを備え、コンピューティングアクティビティをますますウェブブラウザまたはネットワーク指向にしている。

【００２３】

例えば、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、サーバー、ウェブベースのデータストレージなどのビルディングブロックサービス、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。本明細書の例示的な実施形態は、コンピューティングデバイス上に存在するソフトウェアに関連して説明されるが、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ース（ＡＰＩ）、または、コプロセッサ、表示デバイスおよびリクエスト側オブジェクトのいずれかの間の「ミドルマン」オブジェクトを介して実施して、本発明による操作を、．ＮＥＴ（商標）の言語およびサービスのすべてによって実行したり、サポートしたり、あるいはアクセスしたりすることができ、これは他の分散コンピューティングフレームワークにおいても同様である。

【００２４】

（本発明の例示的な実施形態）

リレーショナルデータベース管理オブジェクトスクリプタは、オブジェクトのためのスクリプトを生成する。一実施形態では、オブジェクトはＳＱＬデータベースにおいて実施される。本発明は、その一般的な適用可能性に関しても、ＳＱＬ環境の実施形態に関しても説明することができる。そのようなスクリプトの一例は、Ｔｒａｎｓａｃｔ－ＳＱＬスクリプトである。スクリプティングにより、ＳＱＬデータベース上のオブジェクトに係わるシステム管理タスクの自動化が可能となる。スクリプタオブジェクトは、ＳＱＬ管理オブジェクト名前空間に存在するオブジェクトであるが、スクリプタの使用に依存するＳＱＬ管理オブジェクトはない。スクリプタは、各インスタンスクラスによって実施される２つの操作、ｃｒｅａｔｅおよびｄｒｏｐ操作を使用することができる。それゆえ、インスタンスオブジェクトは、ＳＱＬデータベース上にこうしたオブジェクトの作成または削除のためのスクリプトテキストを生成する目的でそれら自体のインスタンスのスクリプティングのみを担当する。結果として、スクリプタは、依存性の発見、スクリプトのメモリへの出力、ファイルまたは表示、インスタンスオブジェクトをスクリプトする目的でこれらのオブジェクトをコールすること、および、スクリプティング操作のコンテキストおよび進行をコントロールすることを含む残りの機能を担当する。

【００２５】

スクリプタオブジェクトモデルを部分的に使用して、本発明を実施することができる。スクリプタオブジェクトモデルは、スクリプティングのための１つのエントリポイントである。オブジェクトモデルは、スクリプティング操作のコンテキストを保持する。モデルの中のオブジェクトは、ユニフォームリソース名を使用することで一意に識別することができる。例えば、「ｄｂｏ」によって所有されたテーブル「ａｕｔｈｏｒｓ」を収容する「ｐｕｂｓ」というデータベース名を有するＳＱＬ管理オブジェクトは、以下のユニフォームリソース名によって一意に参照することができる。

```
Server/Database[@Name='pubs']/table[@Name='authors' and schema='dbo']
```

【００２６】

ユニフォームリソース名の使用により、オブジェクトを一意に識別する標準フォーマットが提供され、他の標準の使用と調和され、本発明の適用において柔軟性および将来の発展を可能にする。

【００２７】

スクリプタは、基礎となるデータ構造の中間操作を可能にするいくつかの異なるフェーズにおいて動作する。これらのフェーズは、全体として実行することができ、最小限の操作を可能にし、または独立して実行することができ、これにより基礎となるデータ構造を修正する最大限の自由が与えられる。スクリプタが実行する例示的なフェーズは、依存性を発見すること、依存性リストを生成すること、依存性リストからスクリプトを生成することである。

【００２８】

図２は、本発明の例示的なフェーズの間の関係、ならびに、フェーズおよびユーザーまたは他のコントロールプログラムの間のデータの流れを示すブロック図である。ユーザープログラム２６０は、例示的な異なるフェーズ、すなわち、発見、リスト生成およびスクリプト生成において本発明と相互に作用することができる。各フェーズの期間中に、ユーザープログラムは特定の操作に入力を提供する。各フェーズの期間中に結果が戻され、これを操作し、プロセスの次のフェーズに入力することができる。各フェーズの期間中に、イベントは、ユーザープロセスがこれらのイベントにサブスクライブしている場合、ユーザープロセスに送信され、これによりプロセスの進行についての情報が提供される。このイベントメカニズムにより、ユーザーは個々のオブジェクトについてのスクリプタの振る舞いを修正できるようになり、これは、ユーザーがこれらのイベントに関するプロセスに応答し、インタラクトすることができるようにすることによって行なわれる。スクリプト生成フェーズ中に渡されたスクリプティングオプションは、スクリプタ出力の全体的なコントロールを提供する。

【００２９】

各フェーズは望ましくは分離され、各フェーズに対する入力は必ずしもスクリプタによって生成される必要はない。ユーザープログラムは、それ自体の内部アルゴリズムに基づいてそれ自体のデータ構造を作成し、これらのデータ構造をスクリプタに入力として提供することができる。それゆえ、スクリプタフェーズを３つの個別のコンポーネントと見なすことができる。

【００３０】

図２に戻ると、発見フェーズは、ユーザープログラム２６０からオブジェクト参照２０２を受信して、依存性発見メカニズム２１０を開始する。これらのオブジェクトは、ユニフォームリソース名を使用して参照することができる。依存性データおよびインスタンスデータ（メタデータ）２０４は、発見プロセスを支援するためにシステムからコールされる。特に、依存性データは、サーバーまたはシステムカタログ２７０から検索可能な依存性を介して特定することができる。依存性の細分性は、依存性発見操作の細分性も決定することに留意されたい。２つ以上のオブジェクトの間の依存性が、依存性検索および発見または検出アルゴリズムを使用して発見されると、オブジェクト関係データ２０８が依存性ツリー２１２に入力される。このプロセスは発見メカニズム２１０に戻って２１４、依存性ツリーに対するエントリの作成を継続することができる。プロセスの最後に、依存性ツリー２１２が生成され、ユーザープログラム２６０に利用できるようにする２１６。

【００３１】

依存性検索および検出技術は、その操作の点から説明することができる。さらに、このアルゴリズムは、親オブジェクトのグラフ化（ｇｒａｐｈｉｎｇ）、元のオブジェクトのグラフ化、そして次に子オブジェクトのグラフ化として説明することができる。グラフ化は本質的には、依存性ツリーの構築である。最初に、依存性発見メカニズム２１０がオブジェクト参照を見て、このオブジェクトを列挙されたオブジェクトとしてコールする。オブジェクトの実際のインスタンス化は、この時点では必要なく、それゆえ、オブジェクトそれ自体はコールされず、オブジェクトの参照のみが呼び出される必要がある。このオブ

ジェクトは既知のオブジェクトリストに追加される。依存性発見メカニズムは、既知のオブジェクトリスト内にない追加オブジェクトの各親を見つける。依存性メカニズムは次に、親オブジェクトをコールすることができ、これを既知のオブジェクトリストに追加する。最高層の親が検出されたとき、この最高層の親が階層ツリーである依存性ツリーに追加される。次いで、依存性発見メカニズムは下方へ、既知のオブジェクトリスト上のオブジェクトのすべての子についてサーチする。これにより、元のオブジェクトが依存性ツリーに追加されるだけでなく、発見された親の子も、元のオブジェクトも、子および孫オブジェクトも同様である。この再帰的アルゴリズムは、正しい依存性順序を生成する。

【００３２】

図２のフェーズ２は、依存性リスト生成に関係する。依存性ツリーは、フェーズ１（２１６）から獲得することができ、あるいは、ユーザープログラム２６０へのインターフェース２１８を介して獲得することができる。いずれの場合も、依存性ツリー２１８が受け取られ、リスト生成メカニズム２２０がオブジェクトリスト２３０を生成するタスクに対してアクティブ化される。リスト生成メカニズム２２０は、依存性ツリー２１２およびインスタンスデータ（メタデータ）２２２とインターフェースして２２４、２２６、オブジェクトリスト２３０（依存性リストとも称する。）に対するエントリ２２８を生成する。オブジェクトリストエントリは、オブジェクトおよびその関係データが検出されると、生成することができる。プロセスが戻って依存性ツリーからの追加の項目を読み取り、依存性ツリー要素が果てるまで、オブジェクトリストを生成する。オブジェクトリストエントリが検出されるなどのイベントとして、イベント２２９が生成されて、ユーザープログラムに、オブジェクトリストを編集することによってプロセスを操作する機会が提供される。プロセスの最後で、オブジェクトまたは依存性リスト２３０が生成され、ユーザープログラム２６０に利用できるようにされ２３２、ユーザープログラムによって完全に編集可能となる。

【００３３】

図２のフェーズ３はオブジェクトリストおよびスクリプタオプション２３４を使用して、スクリプト生成メカニズム２４０を介して、スクリプト（例えば、Ｔｒａｎｓａｃｔ－ＳＱＬスクリプト）を生成する。オブジェクトリストは、以前のフェーズ２３２によって生成されたオブジェクトリストと同じものとすることができ、あるいは、ユーザープログラム２６０へのインターフェース２３４を介して受け取られたオブジェクトリストとすることができる。いずれの場合も、スクリプト生成メカニズム２４０は、ユーザープログラムによって選択またはデフォルトにされたスクリプティングオプション２３４を、インスタンスデータ（メタデータ）２３６と共に受け取る。スクリプトジェネレータメカニズム２４０は、オブジェクトリスト２３０が処理されるときに２３８、２４２、スクリプトエントリ２４４を生成することがある。スクリプト２５０が生成されているとき、この生成されたスクリプトを編集するための新しいエントリまたは機会のようなイベント２５２をユーザープログラム２６０に提示することができる。最終的には、オブジェクトリスト２３０が完全に処理され、完全なスクリプト２５０が生成される。次に、後続の処理、または、ユーザーもしくは同等のインターフェースへの配信のためにこのプロセスがスクリプト２５４をユーザープログラムに配信することができる。

【００３４】

図２に示すように、スクリプティングの全体的なプロセスを、オブジェクト参照２０２の受領から開始し、スクリプト２５４の生成で終了する１つのプロセスと見ることができる。あるいは、このプロセスを、連結することができる複数の独立したプロセスと見ることもできる。図３、４および５は、フェーズ１、２および３の個別プロセスをそれぞれ表している。

【００３５】

図３は、本発明の例示的な依存性発見フェーズ方法３００についての流れ図である。依存性発見フェーズは、プロセスに渡された単一または複数のオブジェクト参照から階層オブジェクトツリー（グラフ）を作成する。このプロセスは、ＳＱＬデータベースなどのリ

レーショナルデータベースでの使用が望まれるオブジェクトに関係する１つまたは複数のオブジェクト参照を受け取ることによって開始する３１０。次いで、このプロセスは、参照されたオブジェクトに関係する依存性を検出するためのサーチの実行に移る。依存性データは、オブジェクト間の関係に関する情報として定義される。例えば、ＳＱＬのビューは、その存在について、関係するテーブルに依存することがある。逆に、この関係するテーブルは、従属するＳＱＬビューを有することがある。テーブルインスタンスまたはメタデータなどのオブジェクトインスタンスを記述するインスタンスデータは、付属物として、カバーされていない関係依存性データの一部である可能性がある３２０。

【００３６】

依存性が検出されると、依存性データおよび関係するオブジェクトを表現する依存性または階層ツリーが生成される３３０。階層ツリーまたはグラフを作成するために使用されるアルゴリズムは、重複を除去する３４０。というのは、複雑な階層をスクリプトする間に、同じオブジェクトへの参照が発生する可能性があるからである。複数回参照されるこれらのオブジェクトは、スクリプトアウト（ｓｃｒｉｐｔ　ｏｕｔ）することができる。依存性ツリーまたは階層ツリーは例えば、親、第１の子、次の兄弟、孫、および曾孫タイプの依存性構造を含むことができる。依存性が検出されている間、このプロセスは、依存性ツリーが完成しているかどうかをテストする３５０。完成していない場合、このプロセスは、オブジェクトの編集を可能にすることができる３６０。編集がリクエストされない場合３６０、このプロセスは、次のオブジェクト参照に移動することによって継続する３９０。編集が望まれる場合３６０、ツリーを編集することができる３７０。例えば、削除されたオブジェクトは結果として、オブジェクトの削除だけでなく、関係ブランチにおけるすべての後続の子オブジェクトの削除となることもある。これによりユーザーはオブジェクトを、それらが最終グラフに追加される前に操作したり、フィルタしたりすることができる。編集が完了した後、このプロセスは次のオブジェクト参照に移動し３８０、依存性データのサーチを継続する３２０。

【００３７】

何のオブジェクト編集も、コントロールプロセスまたはユーザーによってリクエストされない場合、依存性検出３００は次のオブジェクト参照へ継続し、依存性データのサーチが再開する３２０。最終的に、依存性ツリーまたは階層ツリーが完成され３５０、依存性ツリーが後続のプロセスへの出力３９９として、あるいはユーザー出力として利用できるようにされる３９５。

【００３８】

図４は、本発明の例示的な依存性走査フェーズ方法４００についての流れ図である。依存性走査フェーズ（ｄｅｐｅｎｄｅｎｃｙ　ｗａｌｋ　ｐｈａｓｅ）は、プロセスに渡された依存性ツリーから依存性の線形リストを作成する。このプロセスは、依存性ツリーをプロセスへの入力として受け取ることによって開始する４１０。この入力は、図３などの出力３９９から導出することができ、あるいはユーザーまたは進行するプロセスによって別個に入力することができる。

【００３９】

図４に戻ると、依存性走査方法４００は、依存性ツリーから導出された関係依存性の検出４２０に進む。依存性が検出されると、このプロセスは、依存性リストエントリを生成する４３０。依存性リストは線形リストであることが好ましく、これは、オブジェクトが依存性制約を満たすために作成されるべき順序をリストする。例えば、ユーザー定義のデータ型はオブジェクトの一部として、場合によってはメタデータとして、そのオブジェクトまたはユーザーデータ型に依拠するテーブルが作成されるのに先立って、存在しなければならないことがある。

【００４０】

このプロセスが最初のパススルーで完了していないと仮定すると４４０、プロセス４００は、作成されたリストに対する編集を可能にする。リストが編集されることになる場合４５０、この編集を可能にすることができ４６０、依存性リストの参照されたオブジェク

トの除去または修正を行なうことができる。編集が完了した後、このプロセスは、依存性ツリーの次のオブジェクトへ継続し４７０、関係依存性についての検出を継続する４２０。編集が望まれなかった場合４５０、このプロセスの進行が通知され４８０、プログラムが次のツリーオブジェクト、および関係依存性の継続検出へインデックスする４２０。

【００４１】

最終的に、このプロセスは、ツリーが完全にチェックされ、依存性リストエントリ作成のプロセスが完了されるまで４４０、繰り返される。次いで、依存性リストは、後続のプロセス、またはユーザーによる使用のための出力４９９に対して利用できるようにすることができる。

【００４２】

図５は、本発明の例示的なスクリプティングフェーズ方法５００についての流れ図である。スクリプティングフェーズは、プロセスに渡された依存性リストからＴｒａｎｓａｃｔ－ＳＱＬスクリプトなどのスクリプトを生成する。このプロセスは、依存性リストをプロセスへの入力として受け取ることによって開始する５１０。この入力は、図４などの出力４９９から導出することができ、あるいはユーザーまたは進行するプロセスによって別個に入力することができる。図５に戻ると、スクリプティングフェーズ５００は、依存性リストオブジェクト参照のインスタンス化５２０に進む。次に、参照されたオブジェクトに対応するスクリプトへのオブジェクト参照に関するコールが行なわれる５３０。このコールの結果生じるスクリプト要素が戻されると、プロセスが繰り返される際にスクリプトが累積される５４０。

【００４３】

このプロセスが完了していないと仮定すると５５０、プロセス５００は、スクリプトの編集を可能にする５６０。編集が望まれる場合、ユーザーまたは他のコントロールプロセスがスクリプトを編集することができ５７０、次いで、依存性リストの次のオブジェクトへ継続し５８０、次のオブジェクト参照の次のインスタンス化を可能にする５２０。編集が望まれない場合５６０、進行通知がコントロールプロセスまたはユーザーに示され５９０、プロセス５００はインスタンス化のために次のオブジェクト参照へ継続する。

【００４４】

最終的に、このプロセスは、依存性リスト上のすべてのオブジェクトをステップスルーし、このプロセスを完了する５５０。その後、累積されたスクリプトは、後続のプロセス、またはユーザーによる使用のための出力５９９として利用することができるようにされる５９５。

【００４５】

図３、４および５におけるプロセスは、本発明の意図から逸脱することなく、プロセスにおいていずれかの時点での進行通知、あるいはいずれかの時点での編集を含めるか、または排除するように変更することができる。例えば、図５において、スクリプト進行モニタポイントを、本発明から異なることなく、スクリプトアキュムレータが書き込んだ後５４０、または完了チェックの後５５０、または編集の後５７０、またはスクリプトが使用可能にされた後５９５に配置することができる。流れ図に対するこのオプションとしてのポイント監視の変更は、本発明の趣旨を変えることなく図３、４および５に等しく同様に適用される。

【００４６】

本発明の一実施形態を、図６の統一モデリング言語（ＵＭＬ）ダイアグラムによって表したアーキテクチャで提示する。ＵＭＬダイアグラム６００は、スクリプタオブジェクトモデル内の各クラスを図で記述している。この実施形態は、ユーティリティがＳＱＬデータベース内にあると仮定している。依存性ウォーカー（ＤｅｐｅｎｄｅｎｃｙＷａｌｋｅｒ）６１０は、ＳＱＬサーバーデータベースに収容されているクラス間の依存性または関係を発見するための機能を提供する。これは、スクリプタクラスのためのベースクラスである。依存性ウォーカー６１０はフィルタ委任（ＦｉｌｔｅｒＤｅｌｅｇａｔｅ）６１２および進行レポート委任（ＰｒｏｇｒｅｓｓＲｅｐｏｒｔＤｅｌｅｇａｔｅ）６１４を使

用する。フィルタ委任６１２は、発見フェーズ期間中に見つけたクラスの修正および／または除去を可能にするイベントである。進行レポート委任６１４は、依存性および／またはスクリプティングフェーズの進行についての進行情報を提供するイベントである。

【００４７】

スクリプタ（Ｓｃｒｉｐｔｅｒ）６２０はメインスクリプティングクラスであり、スクリプティング機能をエンドユーザーに公開する。スクリプタ６２０は進行レポート委任６１４およびエラーイベント委任（ＥｒｒｏｒＥｖｅｎｔＤｅｌｅｇａｔｅ）６２４を使用する。スクリプティングオプション（ＳｃｒｉｐｔｉｎｇＯｐｔｉｏｎｓ）６２２は、スクリプタ６２０の振る舞いの変更を可能にするクラスである。スクリプティングオプション６２２は、スクリプタ６２０クラスのプロパティとして公開される。しかし、スクリプティングオプションクラス６２２を別々にインスタンス化することができ、これによりこのクラスを、個別のクラスに関するスクリプトメソッドの引数として渡すことが可能になる。

【００４８】

依存性ノードクラス（ＤｅｐｅｎｄｅｎｃｙＮｏｄｅ）６３０は、オブジェクト参照としてＵＲＮを収容する。依存性ノード６３０クラスは、すべての依存性ツリーまたはリストクラスについてのベースクラスである。依存性ツリーノードクラス（ＤｅｐｅｎｄｅｎｃｙＴｒｅｅＮｏｄｅ）６３２は、親および子関係についての情報を保持するクラスである。これは依存性ノードクラス６３０を拡張する。依存性ツリークラス（ＤｅｐｅｎｄｅｎｃｙＴｒｅｅ）６３４はスクリプタ関係を収容する。このクラスは依存性ツリーノードクラス６３２を拡張することができる。依存性ツリークラス６３４は、スクリプタクラス６２０のプロパティとして公開される。

【００４９】

アレイリスト（ＡｒｒａｙＬｉｓｔ）６４０は、．ＮＥＴフレームワークからの標準クラスであり、０からｎ個のクラスの線形リストにて参照を保持することができる。アレイリストクラスは、依存性リストクラス（ＤｅｐｅｎｄｅｎｃｙＬｉｓｔ）６４２に対するベースリストとして機能する。依存性リストクラス６４２は、依存性リストノード（ＤｅｐｅｎｄｅｎｃｙＬｉｓｔＮｏｄｅ）６４４クラスの線形リストを保持することができる。これは依存性リスト６４２クラスによって使用され、スクリプタクラス６２０のプロパティとして公開される。依存性リストノード６４４は、オブジェクトが元のリストの一部（すなわち、ルートオブジェクト）であったかどうかについての情報を収容する。これは依存性ノードクラス６３０を拡張する。

【００５０】

本発明の一実施形態によれば、スクリプタは、操作および有用性における柔軟性を可能にするオプションを有することができる。図２のプロセスはまた、フィルタを適用することによって操作することもできる。このフィルタをフェーズ２および３の期間にコールして、オブジェクトをこれらのフェーズ期間中に除去したり、変更したりできるようにすることができる。スクリプタジェネレータは、（子またはより下層のオブジェクトを含む）オブジェクトを排除するために使用することができるプロパティを実装することができる。さらに、スクリプティングフェーズの１つの期間に変更されたオブジェクトもまた、変更を含め、スクリプトアウトすることができる。

【００５１】

オブジェクト名は、名前付けプロパティがそのオブジェクトに関して利用可能であれば、これを修正することによって変更することができる。オブジェクトは、その目的ために永続化させる必要はない。必要ならば、スキーマプロパティがそのオブジェクトに関して利用可能であれば、これを設定することによってスキーマを変更することができる。スクリプティングエラーイベントは、スクリプト作成フェーズ期間に発生するエラーを受け取るように設定することができる。エラーが発生したとき（すなわち、オブジェクトをサーバー上で見つけることができないとき）、エラー発生時に継続するためのオプションを可能にすることによって、継続することが企図されている。

【００５２】
システムオブジェクトをスクリプトして、ユーザーが新しい（非システム）オブジェクトを、システムオブジェクトの構造に基づいて作成できるようにすることができる。以下のスクリプティングルールを、システムオブジェクトに適用することができる。
【００５３】
（１）ユーザーは、システムオブジェクトをスクリプタに渡し、スクリプトを生成することが許可されるべきである。
【００５４】
（２）このスクリプトは、サーバーがシステムオブジェクトを作成することを許可しない可能性があるので、事前の修正なくサーバー上で実行されるべきではない。
【００５５】
（３）ユーザーが複数のオブジェクトを渡し、オブジェクトの１つがシステムオブジェクトである場合、このシステムオブジェクトが許可される可能性がある。発見中にシステムオブジェクトが検出される場合にスクリプティングを停止するためのオプションを選択することができる。これは、全データベース、または依存性を含む多くのオブジェクトをスクリプトアウトするとき、重要となることがある。
【００５６】
異なるスクリプト操作を使用して、１つのパスにおいて１つまたは複数のオブジェクトがスクリプトされる。オプションとして、フィルタ機能を通すことができ、これを使用して、一意にオブジェクトを識別するために使用されるユニフォームリソース名をフィルタして除くことができる。このフィルタは、発見フェーズ（例えば、図２のフェーズ１）期間中にオブジェクトが依存性ツリーに追加されるとき、コールされる。これは、カスタマイゼーションの目的のために有用である場合がある。フィルタされたオブジェクトおよびそのすべての依存性はスクリプトされない。
【００５７】
別の実施形態では、スクリプティングオプションを本発明の一部として提供することができる。これらのオプションにより、ユーザー、または、エグゼクティブもしくはアプリケーションプログラムなどのコントロールプログラムが、スクリプティングプロセスをコントロールすることができる。オプションのいくつかを以下に説明する。
【００５８】
【表１－１】

スクリプト出力フォーマットオプション

スクリプタオプション	説　明
ファイルに付加：	指示された出力ファイルに付加する。デフォルトで、スクリプトメソッドは既存のファイルを上書きする。
ANSIファイル：	生成されたスクリプトファイルは、マルチバイト文字を使用する。

【００５９】

【表１－２】

ドロップ；	Transact-SQLを生成して、参照されたコンポーネントを除去する。スクリプトは、コンポーネントを除去しようと試みる前に、存在についてテストする。
PWD暗号化；	パスワードをスクリプトにより暗号化する。
ヘッダを含める；	生成されたスクリプトを、生成の日時および他の記述情報が入っているヘッダでプレフィックスする。
存在しない場合を含める；	コンポーネントを作成するTransact-SQLを、存在についてのチェックでプレフィックスする。スクリプトが実行されるとき、コンポーネントは、指名されたコンポーネントのコピーが存在しないときにのみ作成される。
コマンドタームなし；	スクリプト内の個々のTransact-SQLステートメントを、接続特有のコマンドターミネータを使用して区切らない。デフォルトでは、個々のTransact-SQLステートメントを区切る。
スキーマ修飾；	オブジェクトを除去するために生成されたTransact-SQL内のオブジェクト名は、参照されたオブジェクトの所有者によって修飾される。参照されたオブジェクトを作成するために生成されたTransact-SQLは、現在のオブジェクト所有者を使用してオブジェクト名を修飾する。
スキーマ修飾外部キー；	スキーマ修飾テーブルは、外部キー制約について参照する。
タイムスタンプをバイナリにする；	テーブルまたはユーザー定義のデータ型のためのオブジェクト作成をスクリプトするとき、タイムスタンプデータ型の仕様をバイナリに変換する。

【００６０】

【表１－３】

ファイル出力のみ；	大部分のSQLオブジェクトスクリプティングメソッドは、戻り値およびオプションの出力ファイルを指定する。使用され、出力ファイルが指定されるとき、このメソッドはスクリプトをコール側に戻さず、スクリプトを出力ファイルに書き込むのみである。依存性のスクリプトアウトにより潜在的に非常に大きい文字列が生じる可能性があるので、文字列出力が望まれないときは常に、このオプションを指定する。
ユニコードファイル；	デフォルトでユニコード出力が生成される。
ログインSID；	スクリプトされたログインのためのセキュリティ識別子を含める。
DDLヘッダのみ；	ストアドプロシージャなど、本文テキストを有するオブジェクトのためのDDLヘッダのみをスクリプトする。デフォルトは、完全なDLLをスクリプトアウトすることである。
DDL本体のみ；	ストアドプロシージャなど、本文テキストを有するオブジェクトのためのDDL本体のみをスクリプトする。デフォルトは、完全なDLLをスクリプトアウトすることである。

【００６１】

【表１】

スクリプト依存性オプション

スクリプタオプション	説　明
依存性あり：	出力スクリプトリストを、すべての依存オブジェクトを含むように拡張する。
データベースパーミッション：	生成されたTransact-SQLデータベースの特権を定義するスクリプト。データベースパーミッションは、ステートメントを実行する権利を付与または拒否する。
スクリプトインデックス：	OR論理演算子を使用して結合されたクラスタ化インデックス、非クラスタ化インデックスおよびDRIインデックス。テーブルおよびビューオブジェクトに適用される。
パーミッション：	OR論理演算子を使用して結合されたSQL SMOスクリプトオブジェクトパーミッションおよびSQL SMOスクリプトデータベースパーミッション。
プライマリオブジェクト：	参照されたコンポーネントを作成するTransact-SQLを生成する。
拡張プロパティ：	オブジェクトスクリプティングの一部として拡張プロパティスクリプティングを含める。
XML名前空間：	XML名前空間をオブジェクトスクリプティングの一部として含める。
フルテキストカタログ：	コマンドバッチは、サーチフルテキストカタログを作成するTransact-SQLステートメントを含む。

【００６２】

【表３－１】

サーバー間レベルのスクリプティング

スクリプタオプション	説　明
照合なし；	デフォルトは、照合を生成することである。テーブルオブジェクトに特有。
フルテキストインデックス；	コマンドバッチは、サーチフルテキストインデクシングを定義するステートメントを含む。
バインディング；	sp-bindefaultおよびsp-bindruleステートメントを生成する。スクリプティングがＳＱＬサーバーテーブルを参照するときにのみ、適用される。
クラスタ化インデックス;	クラスタ化インデックスを定義するTransact-SQLを生成する。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。
DRI-全部；	OR論理演算子を使用して結合されたDRIとして定義されたすべての値。
DRI-全制約；	OR論理演算子を使用して結合されたDRI-チェック、DRIデフォルト、DRI外部キー、DRIプライマリキー、DRIユニークキー、DRI XMLキー。
DRI-全キー;	OR論理演算子を使用して結合されたDRI外部キー、DRIプライマリキー、DRIユニークキー。
XMLインデックス；	生成されたスクリプトは、XMLインデックスを作成する。
DRI-チェック；	生成されたスクリプトは、列特有のCHECK制約を作成する。宣言型参照保全性が依存関係を確立するとき、スクリプティングを指示する。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。

【００６３】

【表３－２】

DRI-クラスタ化；	生成されたスクリプトは、クラスタ化インデックスを作成する。宣言型参照保全性が依存関係を確立するとき、スクリプティングを指示する。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。
DRI-デフォルト；	生成されたスクリプトは、列特有のデフォルトを含む。宣言型参照保全性が依存関係を確立するとき、スクリプティングを指示する。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。
DRI-外部キー；	生成されたスクリプトは、外部キー制約を作成する。宣言型参照保全性が依存関係を確立するとき、スクリプティングを指示する。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。
DRI-非クラスタ化；	生成されたスクリプトは、非クラスタ化インデックスを作成する。宣言型参照保全性が依存関係を確立するとき、スクリプティングを指示する。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。
DRI-プライマリキー；	生成されたスクリプトは、PRIMARY KEY制約を作成する。宣言型参照保全性が依存関係を確立するとき、スクリプティングを指示する。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。
DRI-ユニークキー；	生成されたスクリプトは、ユニークインデックスを使用して定義された候補キーを作成する。宣言型参照保全性が依存関係を確立するとき、スクリプティングを指示する。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。
DRI-インデックス；	ユニークインデックスを使用して宣言型参照保全性を実施するPRIMARY KEY制約をスクリプトする。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。

【００６４】

【表３－３】

DRIチェックなし：	DRIチェックまたはDRI外部キーを使用しているとき、生成されたスクリプトは、制約作成を最適化するWITH NO CHECK文節を含む。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。
アイデンティティなし：	生成されたTransact-SQLステートメントは、アイデンティティプロパティの定義、シードおよび増分を含まない。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。
非クラスタ化インデックス：	非クラスタ化インデックスを定義するTransact-SQLを生成する。スクリプティングがＳＱＬサーバーテーブルを参照するときにのみ、適用される。
オブジェクトパーミッション：	データベースオブジェクトをスクリプトしているとき、Transact-SQL特権を定義するステートメントを含める。
トリガ：	Transact-SQLを定義するトリガを生成する。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。
ユーザー型を基本型にする：	ユーザー定義データ型の仕様を適切なSQLサーバー基本データ型に変換する。スクリプティングがSQLサーバーテーブルを参照するときにのみ、適用される。
ファイルグループなし：	コマンドバッチは、ｆｉｌｅｇｒｏｕｐ使用を指示する「ON<filegroup>」文節を含まない。

【００６５】

【表４】

その他

スクリプタオプション	説　明
システムオブジェクトを可能にする（ブール）；	システムオブジェクトのスクリプティングを可能にする。指定されない場合、システムオブジェクトがフィルタリングして除かれる。
エージェントアラートジョブ；	SQLサーバーエージェントサービスジョブおよびアラートを作成するTransact-SQLスクリプトを生成する。
エージェント通知；	アラートをスクリプト中であるとき、アラートについての通知を作成するスクリプトを生成する。
ANSI埋め込み；	コマンドバッチは、SET ANSI PADDING ONおよびSET ANSI PADDING OFFステートメントを、生成されたスクリプト内でCREATE TABLEステートメントの前および後に含める。
What Ifインデックスなし；	コマンドバッチは、CREATE STATISTICSステートメントを含まない。
テーブルパーティショニングスキームなし；	コマンドバッチは、テーブルオブジェクトのためのパーティションスキームを含まない。
インデックスパーティショニングスキームなし；	コマンドバッチは、インデックスオブジェクトのためのパーティションスキームを含まない。
アセンブリなし；	コマンドバッチは、アセンブリを含まない。
ビュー列なし；	ビューオブジェクトのための指定された列をスクリプトしない。ビュー列はビュー作成の時点で、これらを明確に指定することによって、あるいは選択されたステートメントによって定義されたように記録される。
データベースコンテキストを含める；	USE［データベース］ステートメントをスクリプトのヘッダに追加する。［データベース］は、スクリプトされるオブジェクトの収容データベース名である。

【００６６】

上述のように、本発明の例示的な実施形態を、様々なコンピューティングデバイスおよびネットワークアーキテクチャに関連して説明したが、基礎となる概念を、自動スクリプタを実施することが望まれる任意のコンピューティングデバイスまたはシステムに適用す

ることができる。このように、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デバイスに適用することができる。例示的なプログラミング言語、名前および例は本明細書で、様々な選択を表すものとして選ばれているが、これらの言語、名前および例は、限定となるように意図されていない。本発明によって達成される同等、類似、あるいは等価のシステムおよび方法を達成するオブジェクトコードを提供する多くの方法があることは、当業者には理解されるであろう。

【００６７】

本明細書で説明した様々な技術は、ハードウェアもしくはソフトウェア、または、適切な場合には両方の組み合わせに関連して実施することができる。このように、本発明の方法および装置、またはそのある態様もしくは一部は、フロッピー（登録商標）ディスケット、ＣＤ－ＲＯＭ、ハードドライブなどの有形のメディア、または他のいずれかの機械可読ストレージメディアにおいて実施されたプログラムコード（すなわち、命令）の形態を取ることができ、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、実行されるとき、このマシンは本発明を実施するための装置となる。プログラム可能なコンピュータ上のプログラムコード実行の場合、コンピューティングデバイスは一般に、プロセッサ、プロセッサによって読み取ることのできるストレージ媒体（揮発性および不揮発性のメモリおよび／またはストレージ要素を含む）、少なくとも１つの入力デバイス、および少なくとも１つの出力デバイスを含む。例えば、データ処理ＡＰＩなどの使用を通じて、本発明の信号処理サービスを利用することができる１つまたは複数のプログラムは、コンピュータと通信するために、高水準の手続き型またはオブジェクト指向型プログラミング言語において実施されることが好ましい。しかし、必要に応じて、このプログラムは、アセンブリまたはマシン言語において実施することができる。いずれの場合にも、これらの言語は、コンパイルされるか、またはインタープリートされた言語、およびハードウェア実施と組み合わせることができる。

【００６８】

また、本発明の方法および装置は、電気配線またはケーブル接続を介して、光ファイバーを通じて、あるいは任意の他の伝送の形態を介してなど、いくつ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具体化された通信を介して実施することもでき、プログラムコードがマシンに受信され、ロードされ、実行されるとき、ＥＰＲＯＭ、ゲートアレイ、プログラム可能なロジックデバイス（ＰＬＤ）、クライアントコンピュータ、ビデオレコーダもしくは同種のもの、または上記の例示的な実施形態において説明したような信号処理機能を有する受信マシンは、本発明を実施するための装置となる。汎用プロセッサ上で実施されるとき、プログラムコードはプロセッサと結合して固有の装置を提供し、この装置が、検討した本発明の機能を呼び出すように動作する。さらに、本発明に関連して使用されたいかなるストレージ技術も常にハードウェアおよびソフトウェアの組み合わせとすることができる。

【００６９】

本発明を、様々な図の好ましい実施形態に関連して説明したが、他の類似の実施形態を使用することができ、また本発明の同じ機能を実行するために、本発明から逸脱することなく、説明した実施形態に修正および追加を行なうことができることを理解されたい。さらに、特に、無線ネットワーク化されたデバイスの数が増加し続けているので、ハンドヘルドデバイスオペレーティングシステムおよび他のアプリケーション特有のオペレーティングシステムを含む、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が強調されるべきである。それゆえ、本発明はいかなる単一の実施形態にも限定されるべきではなく、むしろ、付属の特許請求の範囲による広がりおよび範囲において解釈されるべきである。

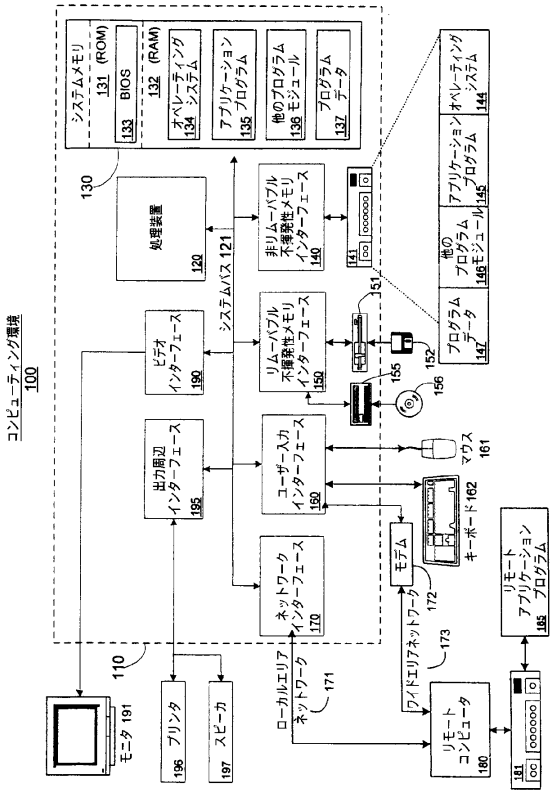
【図面の簡単な説明】

【００７０】

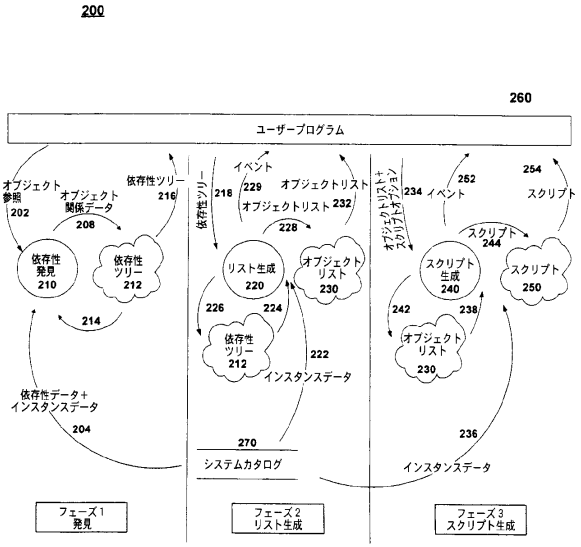
【図１】本発明の態様を実施することができる例示的なコンピューティング環境を示すブロック図である。

【図２】本発明の態様を実施することができる例示的な全体の流れ図である。
【図３】本発明の第１のモジュールまたはフェーズに適用される例示的な流れ図である。
【図４】本発明の第２のモジュールまたはフェーズに適用される例示的な流れ図である。
【図５】本発明の第３のモジュールまたはフェーズに適用される例示的な流れ図である。
【図６】本発明の一実施態様の例示的な静的アーキテクチャ図である。
【符号の説明】
【００７１】
２０２　オブジェクト参照
２０４　依存性データおよびインスタンスデータ（メタデータ）
２０８　オブジェクト関係データ
２１０　依存性発見メカニズム
２１２　依存性ツリー
２１８　インターフェース
２２０　リスト生成メカニズム
２２２　インスタンスデータ
２２８　エントリ
２２９、２５２　イベント
２３０　オブジェクトリスト
２３４　オブジェクトリストおよびスクリプタオプション
２３６　インスタンスデータ（メタデータ）
２４０　スクリプト生成メカニズム
２４４　スクリプトエントリ
２５０、２５４　スクリプト
２６０　ユーザープログラム
２７０　システムカタログ
６００　ＵＭＬダイアグラム

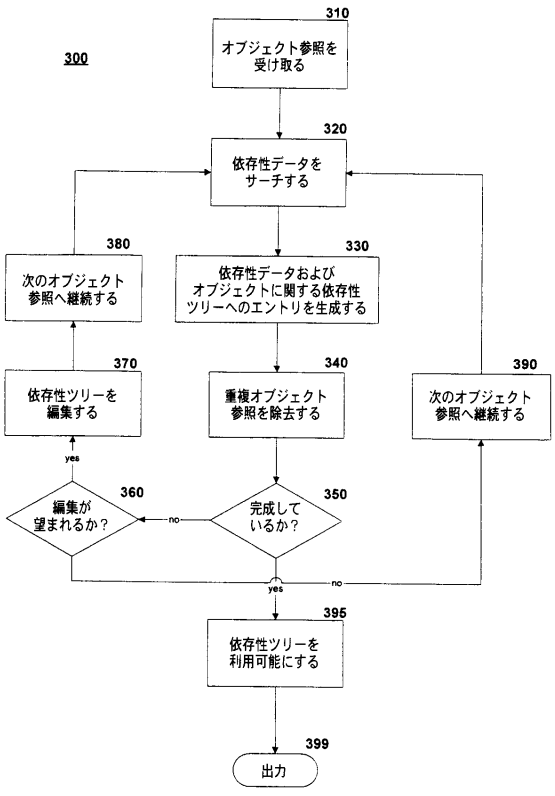
【図１】


コンピューティング環境
100
システムメモリ
131
ROM
BIOS
133
RAM
132
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
他のプログラムモジュール
136
プログラムデータ
137
処理装置
120
システムバス121
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
リムーバブル不揮発性メモリインターフェース
150
非リムーバブル不揮発性メモリインターフェース
140
ユーザー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
モニタ191
プリンタ
196
スピーカ
197
マウス
161
キーボード162
モデム
172
ローカルエリアネットワーク
171
ワイドエリアネットワーク
173
リモートコンピュータ
180
リモートアプリケーションプログラム
185
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
他のプログラムモジュール
146
プログラムデータ
147

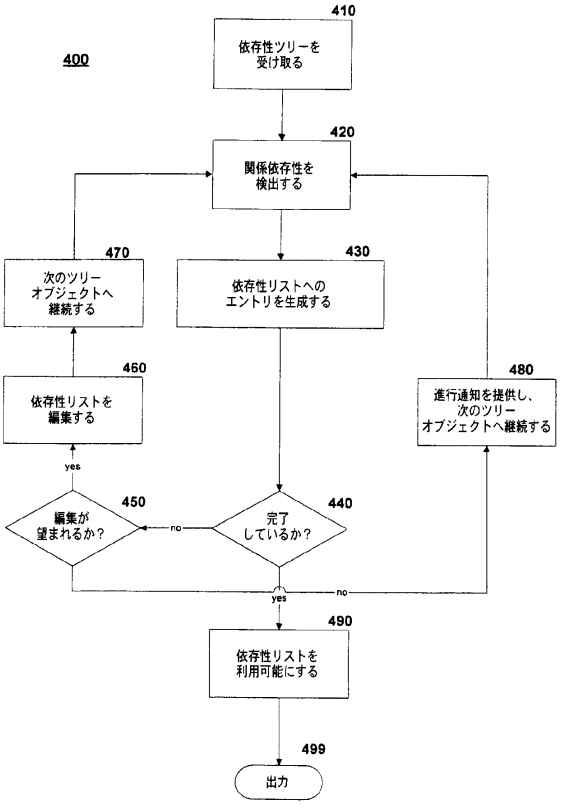
【図２】


200
260
ユーザープログラム
オブジェクト参照
202
オブジェクト関係データ
208
依存性ツリー
216
依存性発見
210
依存性ツリー
212
214
依存性データ＋インスタンスデータ
204
イベント
218
229
オブジェクトリスト
232
228
依存性ツリー
リスト生成
220
オブジェクトリスト
230
226
224
依存性ツリー
212
222
インスタンスデータ
270
システムカタログ
オブジェクトリスト＋スクリプトオプション
234
イベント
252
254
スクリプト
スクリプト
244
スクリプト生成
240
スクリプト
250
242
238
オブジェクトリスト
230
236
インスタンスデータ
フェーズ1
発見
フェーズ2
リスト生成
フェーズ3
スクリプト生成

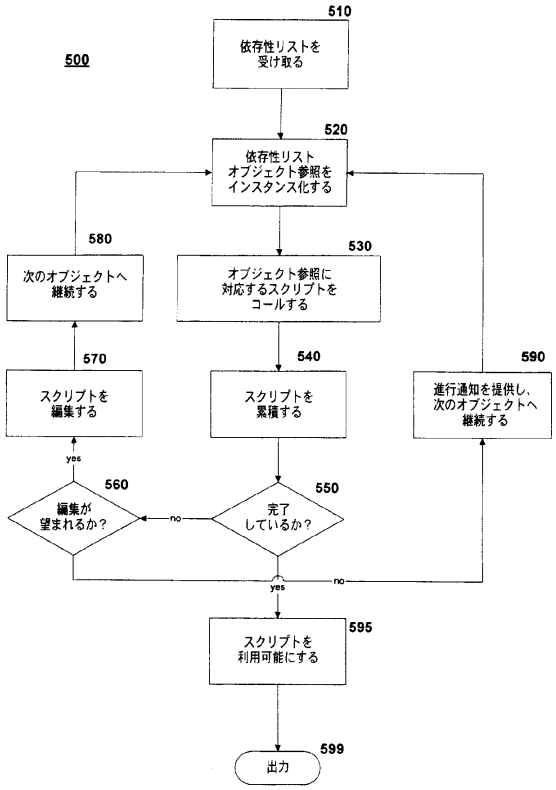
【図３】


300
310
オブジェクト参照を受け取る
320
依存性データをサーチする
330
依存性データおよびオブジェクトに関する依存性ツリーへのエントリを生成する
340
重複オブジェクト参照を除去する
350
完成しているか？
no
yes
360
編集が望まれるか？
no
yes
370
依存性ツリーを編集する
380
次のオブジェクト参照へ継続する
390
次のオブジェクト参照へ継続する
395
依存性ツリーを利用可能にする
399
出力

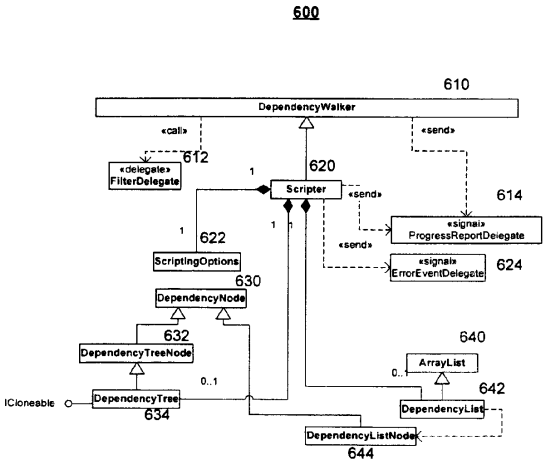
【図４】


400
410
依存性ツリーを受け取る
420
関係依存性を検出する
430
依存性リストへのエントリを生成する
440
完了しているか？
no
yes
450
編集が望まれるか？
no
yes
460
依存性リストを編集する
470
次のツリーオブジェクトへ継続する
480
進行通知を提供し、次のツリーオブジェクトへ継続する
490
依存性リストを利用可能にする
499
出力

【図５】


500
510
依存性リストを
受け取る
520
依存性リスト
オブジェクト参照を
インスタンス化する
530
オブジェクト参照に
対応するスクリプトを
コールする
540
スクリプトを
累積する
550
完了
しているか？
no
yes
560
編集が
望まれるか？
yes
570
スクリプトを
編集する
580
次のオブジェクトへ
継続する
590
進行通知を提供し、
次のオブジェクトへ
継続する
no
595
スクリプトを
利用可能にする
599
出力

【図６】


600
610
DependencyWalker
«call»
«send»
612
«delegate»
FilterDelegate
620
Scripter
«send»
614
«signal»
ProgressReportDelegate
«send»
622
ScriptingOptions
«signal»
ErrorEventDelegate
624
630
DependencyNode
632
DependencyTreeNode
640
ArrayList
0..1
642
DependencyList
ICloneable
DependencyTree
634
DependencyListNode
644

フロントページの続き

(72)発明者　マイケル　ウォーリス
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　２２　コート　１７５１１

審査官　桜井　茂行

(56)参考文献　特開２０００－０６７０７５（ＪＰ，Ａ）
米国特許第６６２９０９１（ＵＳ，Ｂ１）
特開平０６－１７５９０６（ＪＰ，Ａ）
米国特許第５９７８８１１（ＵＳ，Ａ）
米国特許第５５０４８８５（ＵＳ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１２／００
Ｇ０６Ｆ　１７／３０