(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6434960号
(P6434960)
(45)発行日　平成30年12月5日(2018.12.5)　(24)登録日　平成30年11月16日(2018.11.16)

(51)Int.Cl.　　F I
G06Q 50/10 (2012.01)　　G06Q 50/10
G06F 12/00 (2006.01)　　G06F 12/00 ５１３Ａ

請求項の数 16　(全 48 頁)

(21)出願番号　特願2016-513952(P2016-513952)
(86)(22)出願日　平成26年3月26日(2014.3.26)
(65)公表番号　特表2016-529574(P2016-529574A)
(43)公表日　平成28年9月23日(2016.9.23)
(86)国際出願番号　PCT/US2014/031919
(87)国際公開番号　WO2014/186057
(87)国際公開日　平成26年11月20日(2014.11.20)
審査請求日　平成29年3月22日(2017.3.22)
(31)優先権主張番号　61/824,544
(32)優先日　平成25年5月17日(2013.5.17)
(33)優先権主張国　米国(US)
(31)優先権主張番号　61/843,203
(32)優先日　平成25年7月5日(2013.7.5)
(33)優先権主張国　米国(US)

(73)特許権者　502303739
オラクル・インターナショナル・コーポレイション
アメリカ合衆国カリフォルニア州９４０６５レッドウッド・シティー，オラクル・パークウェイ５００
(74)代理人　110001195
特許業務法人深見特許事務所
(72)発明者　アラン，デイビッド
アメリカ合衆国、９４０４０　カリフォルニア州、マウンテン・ビュー、ブラックフィールド・ウェイ、１１１２

最終頁に続く

(54)【発明の名称】フローベースのＥＴＬおよびエンティティリレーションシップベースのＥＴＬの組合せのサポート

(57)【特許請求の範囲】
【請求項１】
データマッピングの生成を促進する方法であって、
１つ以上のコンピュータシステムにて、論理設計のコンポーネントとしてエンティティリレーションシップのセットを特定する情報を受け取ることを含み、
前記エンティティリレーションシップのセットは、データセットにおける第１のエンティティの第１の属性と、前記データセットにおける第２のエンティティの第２の属性との間で定義されるリレーションシップを有し、
前記方法は、さらに、
前記１つ以上のコンピュータシステムに関連付けられる１つ以上のプロセッサにより、前記エンティティリレーションシップのセットに基づいて、データフローモデルを判定することと、
前記１つ以上のコンピュータシステムに関連付けられる前記１つ以上のプロセッサにより、前記論理設計における前記データフローモデルを示す情報を生成することと、
前記第１のエンティティの１以上の属性および前記第２のエンティティの１以上の属性を含む前記論理設計の下流のコンポーネントに属性のセットをエクスポートすることとを含む、方法。
【請求項２】
データソースの属性同士の間のリレーションシップを宣言する情報に基づいて、前記データセットにおける各エンティティの１つ以上の属性を導き出すことをさらに含む、請求

項１に記載の方法。
【請求項３】
前記論理設計を通って流れる情報の形を変更するオペレーションを示す情報を含む前記論理設計の１つ以上のコンポーネントを特定する情報を受け取ることをさらに含む、請求項１または２に記載の方法。
【請求項４】
前記論理設計を通って流れる情報のフローを制御するが前記論理設計を通って流れる情報の形を変更しないオペレーションを示す情報を含む、前記論理設計の１つ以上のコンポーネントを特定する情報を受け取ることをさらに含む、請求項１～３のいずれか１項に記載の方法。
【請求項５】
ターゲットデータストアに格納されるデータの１つ以上の属性を有するターゲットコンポーネントを示す情報を含む、前記論理設計の１つ以上のコンポーネントを特定する情報を受け取ることをさらに含む、請求項１～４のいずれか１項に記載の方法。
【請求項６】
前記データフローモデルは、エンティティが同時に２つのバイナリリレーションシップに参加するように、関連エンティティを吸収する３ウェイリレーションシップを含む、請求項１～５のいずれか１項に記載の方法。
【請求項７】
１つ以上のリレーションシップの導入による前記論理設計における変更を前記１つ以上のコンピュータシステムにて受け取ることと、
前記１つ以上のコンピュータシステムに関連付けられる前記１つ以上のプロセッサにより、更新されたデータフローモデルを判定することとをさらに含む、請求項１～６のいずれか１項に記載の方法。
【請求項８】
データマッピングの生成を促進するためのコンピュータ実行可能コードを備えるコンピュータ読取可能プログラムであって、
論理設計のコンポーネントとしてエンティティリレーションシップのセットを特定する情報を受け取るためのコードを含み、
前記エンティティリレーションシップのセットは、データセットにおける第１のエンティティの第１の属性と、前記データセットにおける第２のエンティティの第２の属性との間で定義されるリレーションシップを有し、
前記コンピュータ読取可能プログラムは、さらに、
前記エンティティリレーションシップのセットに基づいて、データフローモデルを判定するためのコードと、
前記論理設計における前記データフローモデルを示す情報を生成するためのコードと、
前記第１のエンティティの１以上の属性および前記第２のエンティティの１以上の属性を含む前記論理設計の下流のコンポーネントに属性のセットをエクスポートするコードとを含む、コンピュータ読取可能プログラム。
【請求項９】
請求項１～７のいずれか１項に記載の方法をコンピュータに実行させるためのコンピュータ読取可能プログラム。
【請求項１０】
データマッピングの生成を促進するシステムであって、
プロセッサと、
命令を格納するメモリとを含み、前記命令は、前記プロセッサによって実行されると、
論理設計のコンポーネントとしてエンティティリレーションシップのセットを特定する情報を受け取るように前記プロセッサを構成し、
前記エンティティリレーションシップのセットは、データセットにおける第１のエンティティの第１の属性と、前記データセットにおける第２のエンティティの第２の属性との

間で定義されるリレーションシップを有し、
前記プロセッサは、さらに、
前記エンティティリレーションシップのセットに基づいてデータフローモデルを判定し、
前記論理設計における前記データフローモデルを示す情報を生成し、
前記第１のエンティティの１以上の属性および前記第２のエンティティの１以上の属性を含む前記論理設計の下流のコンポーネントに属性のセットをエクスポートするように構成される、システム。
【請求項１１】
前記プロセッサはさらに、データソースの属性同士の間のリレーションシップを宣言する情報に基づいて、前記データセットにおける各エンティティの１つ以上の属性を導き出すように構成される、請求項１０に記載のシステム。
【請求項１２】
前記プロセッサはさらに、前記論理設計を通って流れる情報の形を変更するオペレーションを示す情報を含む前記論理設計の１つ以上のコンポーネントを特定する情報を受け取るように構成される、請求項１０または１１に記載のシステム。
【請求項１３】
前記プロセッサはさらに、前記論理設計を通って流れる情報のフローを制御するが前記論理設計を通って流れる情報の形を変更しないオペレーションを示す情報を含む、前記論理設計の１つ以上のコンポーネントを特定する情報を受け取るように構成される、請求項１０～１２のいずれか１項に記載のシステム。
【請求項１４】
前記プロセッサはさらに、ターゲットデータストアに格納されるデータの１つ以上の属性を有するターゲットコンポーネントを示す情報を含む、前記論理設計の１つ以上のコンポーネントを特定する情報を受け取るように構成される、請求項１０～１３のいずれか１項に記載のシステム。
【請求項１５】
前記データフローモデルは、エンティティが同時に２つのバイナリリレーションシップに参加するように、関連エンティティを吸収する３ウェイリレーションシップを含む、請求項１０～１４のいずれか１項に記載のシステム。
【請求項１６】
前記プロセッサはさらに、
１つ以上のリレーションシップの導入による前記論理設計における変更を受け取り、
更新されたデータフローモデルを判定するように構成される、請求項１０～１５のいずれか１項に記載のシステム。
【発明の詳細な説明】
【背景技術】
【０００１】
発明の背景
ペースが一層速くなっている今日のビジネス環境において、組織はより特殊化されたソフトウェアアプリケーションを使用する必要がある。さらに、組織は、異種混合のハードウェアプラットフォームおよびシステム上でこれらのアプリケーションの共存を保証する必要があるとともに、アプリケーションとシステムとの間でデータを共有する能力を保証する必要がある。
【０００２】
したがって、データ統合シナリオの開発に関係する問題を解決することが望まれており、そのいくつかが本願明細書において論じられ得る。さらに、データ統合シナリオの開発に関係する障害を低減することが望まれており、そのいくつかが本願明細書において論じられ得る。
【発明の概要】

【課題を解決するための手段】
【０００３】
発明の概要
この開示の以下の部分は、少なくとも主題の基本的な理解を提供する目的のために、この開示において発見される１つ以上のイノベーション、実施形態および／または例の簡素化された概要を提供する。この概要は、如何なる特定の実施形態または例の広範囲な概要も提供することを試みてはいない。さらに、この概要は、実施形態または例の主な／決定的な要素を識別するようには意図されておらず、または、この開示の主題の範囲を定めるようには意図されていない。したがって、この概要の１つの目的は、この開示において発見されるいくつかのイノベーション、実施形態および／または例を簡素化された形で、後述されるさらなる詳細な説明の前置きとして提供することであり得る。
【０００４】
さまざまな実施形態において、ユーザはデータ統合システムによって、プラットホームおよび技術に依存しない論理設計を作成し得る。ユーザは、どのようにデータがソースとターゲットとの間に流れることをユーザが望むかをハイレベルで規定する論理設計を作成し得る。ユーザのインフラストラクチャを考慮して、ツールが論理設計を分析し、物理的設計を作成し得る。論理設計は、設計における各ソースおよびターゲットに対応する複数のコンポーネントと、ジョインまたはフィルタのようなオペレーションと、アクセスポイントとを含み得る。物理的設計に転送された際の各コンポーネントは、データに対してオペレーションを行なうようコードを生成する。存在する技術（たとえばＳＱＬサーバ、オラクル、Ｈａｄｏｏｐなど）と使用される言語（ＳＱＬ、ｐｉｇなど）とに依存して、各コンポーネントによって生成されるコードは異なり得る。
【０００５】
１つの局面において、データ統合システムのユーザは、始めから終わりまで、論理設計における各コンポーネントにて、すべてのデータ属性を特定する必要はない。データ統合システムは、論理設計を通って流れる情報を完全に宣言する必要性を回避する、プロジェクタおよびセレクタタイプのような複数のコンポーネントタイプを提供する。データ統合システムは、所定のコンポーネントタイプによって表わされるオペレーションにて何の属性が必要とされるか決定し得る。これは設計およびメンテナンスの両方を簡素化する。さまざまな局面において、既存のＲＤＢＭＳリソースと、パフォーマンスの向上を達成するために別個のプロプラエタリＥＴＬサーバの必要性を回避する性能とをレバレッジするデータ変換および移行が提供される。
【０００６】
一実施形態において、データマッピングの生成を促進する方法は、論理設計のコンポーネントとしてエンティティリレーションシップのセットを特定する情報を受け取ることを含む。エンティティリレーションシップのセットに基づいて、等価なデータフローモデルが判定される。論理フロー設計において等価なデータフローモデルを示す情報が生成される。データソースの属性同士の間のリレーションシップを宣言する情報に基づいて、エンティティリレーションシップのセットを表わすデータセットの１つ以上の属性が導き出され得る。
【０００７】
さらに別の実施形態において、論理設計を通って流れる情報の形を変更するオペレーションを示す情報を含む論理設計の１つ以上のコンポーネントを特定する情報が受け取られ得る。論理設計を通って流れる情報のフローを制御するが論理設計を通って流れる情報の形を変更しないオペレーションを示す情報を含む、論理設計の１つ以上のコンポーネントを特定する情報が受け取られ得る。ターゲットデータストアに格納されるデータの１つ以上の属性を有するターゲットコンポーネントを示す情報を含む、論理設計の１つ以上のコンポーネントを特定する情報が受け取られ得る。
【０００８】
１つの局面において、論理フロー設計において等価なデータフローモデルを示す情報を

生成することは、下流のコンポーネントに属性のリストをエクスポートすることを含み得る。別の局面では、１つ以上のリレーションシップの導入による論理設計における変更が受け取られ得る。その後、更新された等価なデータフローモデルが判定され得る。

【０００９】

一実施形態において、データマッピングの生成を促進するためのコンピュータ実行可能コードを格納する一時的でないコンピュータ読取可能媒体は、論理設計のコンポーネントとしてエンティティリレーションシップのセットを特定する情報を受け取るためのコードと、エンティティリレーションシップのセットに基づいて等価なデータフローモデルを判定するためのコードと、論理フロー設計において等価なデータフローモデルを示す情報を生成するためのコードとを含む。

【００１０】

さらに別の実施形態において、データマッピングの生成を促進するシステムは、プロセッサと、命令を格納するメモリとを含み、命令は、プロセッサによって実行されると、論理設計のコンポーネントとしてエンティティリレーションシップのセットを特定する情報を受け取り、エンティティリレーションシップのセットに基づいて等価なデータフローモデルを判定し、論理フロー設計において等価なデータフローモデルを示す情報を生成するように、プロセッサを構成する。

【００１１】

一実施形態において、データマッピングの生成を促進するシステムは、論理設計のコンポーネントとしてエンティティリレーションシップのセットを特定する情報を受け取るように構成される受取部と、エンティティリレーションシップのセットに基づいて等価なデータフローモデルを判定するように構成される判定部と、論理フロー設計において等価なデータフローモデルを示す情報を生成するように構成される生成部とを含む。

【００１２】

１つの局面において、システムはさらに、データソースの属性同士の間のリレーションシップを宣言する情報に基づいて、エンティティリレーションシップのセットを表わすデータセットの１つ以上の属性を導き出すように構成される導出部を含み得る。１つの局面において、受取部はさらに、論理設計を通って流れる情報の形を変更するオペレーションを示す情報を含む論理設計の１つ以上のコンポーネントを特定する情報を受け取るように構成される。１つの局面において、受取部はさらに、論理設計を通って流れる情報のフローを制御するが論理設計を通って流れる情報の形を変更しないオペレーションを示す情報を含む、論理設計の１つ以上のコンポーネントを特定する情報を受け取るように構成される。

【００１３】

１つの局面において、受取部はさらに、ターゲットデータストアに格納されるデータの１つ以上の属性を有するターゲットコンポーネントを示す情報を含む、論理設計の１つ以上のコンポーネントを特定する情報を受け取るように構成される。１つの局面において、生成部は、下流のコンポーネントに属性のリストをエクスポートするように構成されるエクスポート部を含む。１つの局面において、受取部はさらに、１つ以上のリレーションシップの導入による論理設計における変更を受け取るように構成され、判定部はさらに、更新された等価なデータフローモデルを判定するように構成される。

【００１４】

一実施形態において、データマッピングの生成を促進するシステムは、論理設計のコンポーネントとしてエンティティリレーションシップのセットを特定する情報を受け取るための手段と、エンティティリレーションシップのセットに基づいて等価なデータフローモデルを判定するための手段と、論理フロー設計において等価なデータフローモデルを示す情報を生成するための手段とを含む。１つの局面において、システムは、データソースの属性同士の間のリレーションシップを宣言する情報に基づいて、エンティティリレーションシップのセットを表わすデータセットの１つ以上の属性を導き出すための手段をさらに含む。

【００１５】
別の局面において、システムは、論理設計を通って流れる情報の形を変更するオペレーションを示す情報を含む論理設計の１つ以上のコンポーネントを特定する情報を受け取るための手段をさらに含む。別の局面において、システムは、論理設計を通って流れる情報のフローを制御するが論理設計を通って流れる情報の形を変更しないオペレーションを示す情報を含む、論理設計の１つ以上のコンポーネントを特定する情報を受け取るための手段をさらに含む。
【００１６】
別の局面において、システムは、ターゲットデータストアに格納されるデータの１つ以上の属性を有するターゲットコンポーネントを示す情報を含む、論理設計の１つ以上のコンポーネントを特定する情報を受け取るための手段をさらに含む。別の局面において、論理フロー設計において等価なデータフローモデルを示す情報を生成するための手段は、下流のコンポーネントに属性のリストをエクスポートするための手段を含む。別の局面において、システムは、１つ以上のリレーションシップの導入による論理設計における変更を受け取るための手段と、更新された等価なデータフローモデルを判定するための手段とをさらに含む。
【００１７】
この開示の主題の性質および均等物（ならびに提供される如何なる固有または明白な利点および改良）のさらなる理解は、この開示の残りの部分、任意の添付の図面および請求の範囲への参照により、上記のセクションに加えて実現されるべきである。
【００１８】
この開示において発見されるイノベーション、実施形態および／または例を適切に記載および説明するために、１つ以上の添付の図面に対して参照がなされ得る。１つ以上の添付の図面を説明するよう用いられる付加的な詳細または例は、特許請求される発明のいずれか、ここで記載される実施形態および／または例のいずれか、またはこの開示において示されるいずれかのイノベーションの現在理解されている最良の形態の範囲への限定と見なされるべきではない。
【図面の簡単な説明】
【００１９】
【図１】本発明の実施形態を組み込み得るシステムの簡略図である。
【図２】本発明の実施形態に従ったデータ統合システムのブロック図である。
【図３】本発明の実施形態に従った、データ統合システムを実現するために使用され得るハードウェア／ソフトウェアスタックの簡略ブロック図である。
【図４】データ統合シナリオが本発明のさまざまな実施形態において作成され得るさまざまな異種混合のデータソースを有する環境のブロック図である。
【図５Ａ】データ統合システムによって実行され得る従来のデータ統合処理における簡略化されたデータフローを示す図である。
【図５Ｂ】データ統合システムによって実行され得る従来のデータ統合処理における簡略化されたデータフローを示す図である。
【図６Ａ】本発明の実施形態に従った、データ統合システムによって実行され得る次世代データ統合処理における簡略化されたデータフローを示す図である。
【図６Ｂ】本発明の実施形態に従った、データ統合システムによって実行され得る次世代データ統合処理における簡略化されたデータフローを示す図である。
【図７】本発明に従った一実施形態におけるＯＤＩスタジオとデータ統合システムのリポジトリとの間の相互作用の簡略ブロック図である。
【図８】本発明の実施形態に従った、データ統合シナリオを作成するための方法のフローチャートを示す図である。
【図９】本発明の実施形態に従った、データ統合シナリオを作成するためのユーザインターフェイスのスクリーンショットの図である。
【図１０】本発明の実施形態に従った、マッピングを作成するための方法のフローチャー

トを示す図である。
【図１１】本発明の実施形態に従った、データ統合シナリオにおいてマッピング情報を提供するためのユーザインターフェイスのスクリーンショットの図である。
【図１２】本発明の実施形態に従った、データ統合シナリオにおいてフロー情報を提供するためのユーザインターフェイスのスクリーンショットの図である。
【図１３】本発明の実施形態に従った、パッケージを作成するための方法のフローチャートを示す図である。
【図１４】本発明の実施形態に従った、データ統合シナリオにおいてパッケージシーケンス情報を提供するためのユーザインターフェイスのスクリーンショットの図である。
【図１５】本発明の実施形態に従った、データ統合シナリオを展開するための方法のフローチャートを示す図である。
【図１６】本発明に従った一実施形態における組み合わされたフローベースおよびエンティティベースのマッピングの簡略ブロック図である。
【図１７】本発明の実施形態に従った、組み合わされたフローベースおよびエンティティベースのマッピングを生成するための方法のフローチャートを示す図である。
【図１８】本発明に従った一実施形態におけるデータセットビューを伴う図１６のマッピングの簡略ブロック図である。
【図１９Ａ】本発明に従った一実施形態における組み合わされたフローベースおよびエンティティベースのマッピングについての論理設計の簡略ブロック図である。
【図１９Ｂ】本発明に従った一実施形態における組み合わされたフローベースおよびエンティティベースのマッピングについての物理設計の簡略ブロック図である。
【図２０】本発明の実施形態に従った、組み合わされたフローベースおよびエンティティベースのマッピングの物理設計を生成するための方法のフローチャートを示す図である。
【図２１】静的なＥ－Ｒモデルと動的なＥＴＬモデルとの間のリレーションシップを示す図である。
【図２２】一実施形態における自動的な変換システムのトップレベルの設計チャートを提供する図である。
【図２３Ａ】一般的なＥ－Ｒ表記法での３ウェイリレーションシップを示す図である。
【図２３Ｂ】一般的なＥ－Ｒ表記法での３ウェイリレーションシップを示す図である。
【図２４Ａ】一般的なＥ－Ｒ表記法での３ウェイリレーションシップに対する等価なものを示す図である。
【図２４Ｂ】一般的なＥ－Ｒ表記法での３ウェイリレーションシップに対する等価なものを示す図である。
【図２５】バイナリリレーションシップの連なりを使用して３ウェイリレーションシップに対する等価なものを示す図である。
【図２６】標準的なＥ－Ｒ表記法を使用して３ウェイリレーションシップを示す図である。
【図２７】図２６におけるエンティティについて作成される各テーブルにおけるロウを示す図である。
【図２８】図２８Ａは、一実施形態におけるＥ－Ｒ表記法での３ウェイリレーションシップを示す図であり、図２８Ｂは、一実施形態における３つのエンティティから生じるデータを有するデータフローを示す図である。
【図２９】さまざまなデータベースモデリング方法およびそれらのセマンティックコンテンツの間でリレーションシップをレイアウトするダイアグラムを示す図である。
【図３０】本発明の実施形態を実施するために使用され得るコンピュータシステムの簡略ブロック図である。
【図３１】本発明の実施形態に従ったデータマッピングの生成を促進するためのシステムの簡略ブロック図である。
【発明を実施するための形態】
【００２０】

発明の詳細な説明

イントロダクション

さまざまな実施形態において、ユーザはデータ統合システムによって、プラットホームおよび技術に依存しない論理設計を作成し得る。ユーザは、どのようにデータがソースとターゲットとの間に流れることをユーザが望むかをハイレベルで規定する論理設計を作成し得る。ユーザのインフラストラクチャを考慮して、ツールが論理設計を分析し、物理的設計を作成し得る。論理設計は、設計における各ソースおよびターゲットに対応する複数のコンポーネントと、ジョインまたはフィルタのようなオペレーションと、アクセスポイントとを含み得る。物理的設計に転送された際の各コンポーネントは、データに対してオペレーションを行なうようコードを生成する。存在する技術（たとえばＳＱＬサーバ、オラクル、Ｈａｄｏｏｐなど）と使用される言語（ＳＱＬ、ｐｉｇなど）とに依存して、各コンポーネントによって生成されるコードは異なり得る。

【００２１】

１つの局面において、データ統合システムのユーザは、始めから終わりまで、論理設計における各コンポーネントにて、すべてのデータ属性を特定する必要はない。データ統合システムは、論理設計を通って流れる情報を完全に宣言する必要性を回避する、プロジェクタおよびセレクタタイプのような複数のコンポーネントタイプを提供する。データ統合システムは、所定のコンポーネントタイプによって表わされるオペレーションにて何の属性が必要とされるか決定し得る。これは設計およびメンテナンスの両方を簡素化する。

【００２２】

図１は、この開示において発見されるイノベーション、実施形態および／または例のいずれかの実施形態を組み込み得るか、または、実施形態に組み込まれ得るシステム１００の簡略図である。図１は、本発明を組み込む実施形態を単に例示しており、請求の範囲において記載される本発明の範囲を限定しない。当業者は、他の変形例、修正例および代替例を認識するであろう。

【００２３】

一実施形態において、システム１００は１つ以上のユーザコンピュータ１１０（たとえばコンピュータ１１０Ａ、１１０Ｂおよび１１０Ｃ）を含む。ユーザコンピュータ１１０は、（任意の適切な種類のマイクロソフト社のＷｉｎｄｏｗｓ（登録商標）および／またはアップル社のＭａｃｉｎｔｏｓｈ（登録商標）オペレーティングシステムを実行するパーソナルコンピュータおよび／もしくはラップトップコンピュータを単に例示として含む）汎用のパーソナルコンピュータ、ならびに／または、さまざまな商業的に利用可能なＵＮＩＸ（登録商標）もしくはＵＮＩＸライクなオペレーティングシステムのいずれかを実行するワークステーションコンピュータであり得る。これらのユーザコンピュータ１１０はさらに、本発明の方法を行なうように構成される１つ以上のアプリケーションと、１つ以上のオフィスアプリケーション、データベースクライアントおよび／またはサーバアプリケーションならびにウェブブラウザアプリケーションとを含むさまざまなアプリケーションのうちのいずれかを有し得る。

【００２４】

代替的には、ユーザコンピュータ１１０は、ネットワーク（たとえば以下に記載される通信ネットワーク１２０）を介して通信することができ、ならびに／または、ウェブページもしくは他のタイプの電子文書を表示およびナビゲートすることができるシンクライアントコンピュータ、インターネットが有効化された携帯電話、および／または携帯情報端末といった任意の他の電子デバイスであり得る。例示的なシステム１００は３つのユーザコンピュータを有するよう示されているが、任意の数のユーザコンピュータまたはデバイスがサポートされ得る。

【００２５】

本発明のある実施形態は、通信ネットワーク１２０を含み得るネットワーク化された環境において動作する。通信ネットワーク１２０は、ＴＣＰ／ＩＰ、ＳＮＡ、ＩＰＸ、およびＡｐｐｌｅＴａｌｋなどを含むがこれらに限定されないさまざまな商業的に利用可能な

ネットワークプロトコルのいずれかを用いるデータ通信をサポートし得る、当業者が精通している任意のタイプのネットワークであり得る。単に例示として、通信ネットワーク１２０は、イーサネット（登録商標）ネットワークおよび／もしくはトークンリングネットワークなどを含むがこれらに限定されないローカルエリアネットワーク（「ＬＡＮ」）、ワイドエリアネットワーク、仮想プライベートネットワーク（ＶＰＮ：virtual private network）を含むがこれに限定されない仮想ネットワーク、インターネット、イントラネット、エクストラネット、公衆交換電話ネットワーク（「ＰＳＴＮ：public switched telephone network」）、赤外線ネットワーク、ＩＥＥＥ８０２．１１スイートのプロトコル、当該技術において公知であるＢｌｕｅｔｏｏｔｈ（登録商標）プロトコル、および／もしくは任意の他の無線プロトコルのいずれかの下で動作するネットワークを含むがこれに限定されないワイヤレスネットワーク、ならびに／または、これらおよび／もしくは他のネットワークの任意の組合せであり得る。

【００２６】

本発明の実施形態は、１つ以上のサーバコンピュータ１３０（たとえばコンピュータ１３０Ａおよび１３０Ｂ）を含み得る。サーバコンピュータ１３０の各々は、上で論じられたオペレーティングシステムのいずれかを含むがこれらに限定されないオペレーティングシステムと、任意の商業的に利用可能なサーバオペレーティングシステムとを有するよう構成され得る。サーバコンピュータ１３０の各々はさらに、１つ以上のクライアント（たとえばユーザコンピュータ１１０）および／または他のサーバ（たとえばサーバコンピュータ１３０）にサービスを提供するように構成され得る１つ以上のアプリケーションを実行し得る。

【００２７】

単に例示として、サーバコンピュータ１３０の１つは、単に例示としてウェブページまたは他の電子文書についてユーザコンピュータ１１０からの要求を処理するよう使用され得るウェブサーバであり得る。ウェブサーバはさらに、ＨＴＴＰサーバ、ＦＴＰサーバ、ＣＧＩサーバ、データベースサーバ、およびＪａｖａ（登録商標）サーバなどを含むさまざまなサーバアプリケーションを実行し得る。本発明のいくつかの実施形態において、ウェブサーバは、本発明の方法を実行するよう、ユーザコンピュータ１１０の１つ以上の上でウェブブラウザ内で動作され得るウェブページを取り扱うように構成され得る。

【００２８】

いくつかの実施形態において、サーバコンピュータ１３０は、ユーザコンピュータ１１０および／または他のサーバコンピュータ１３０の１つ以上の上で実行される、クライアントによってアクセス可能な１つ以上のアプリケーションを含み得る１つ以上のファイルおよび／またはアプリケーションサーバを含み得る。単に例示として、サーバコンピュータ１３０の１つ以上は、ユーザコンピュータ１１０および／または他のサーバコンピュータ１３０に応答して、（いくつかの場合において本発明の方法を行なうように構成され得る）ウェブアプリケーションを含むがこれらに限定されないプログラムまたはスクリプトを実行可能な１つ以上の汎用コンピュータであり得る。

【００２９】

単に例示として、ウェブアプリケーションは、Ｊａｖａ、ＣもしくはＣ＋＋のような任意のプログラミング言語、および／または、Ｐｅｒｌ、Ｐｙｔｈｏｎ、もしくはＴＣＬのような任意のスクリプト言語、および、任意のプログラミング／スクリプト言語の組合せで記述された１つ以上のスクリプトまたはプログラムとして実現され得る。アプリケーションサーバはさらに、オラクル、マイクロソフト、およびＩＢＭなどから商業的に利用可能なものを含むがこれらに限定されないデータベースサーバを含み得、当該データベースサーバは、ユーザコンピュータ１１０の１つおよび／またはサーバコンピュータ１３０の別の１つの上で実行されるデータベースクライアントからの要求を処理し得る。

【００３０】

いくつかの実施形態において、アプリケーションサーバは、本発明の実施形態に従って情報を表示するために動的にウェブページを作成し得る。アプリケーションサーバによっ

て提供されるデータは、（たとえばＨＴＭＬ、ＸＭＬ、Ｊａｖａｓｃｒｉｐｔ、ＡＪＡＸなどを含む）ウェブページとしてフォーマットされ得るか、および／または、（たとえば上述したような）ウェブサーバを介してユーザコンピュータ１１０の１つへ転送され得る。同様に、ウェブサーバは、ユーザコンピュータ１１０の１つからウェブページ要求および／もしくは入力データを受け取り得、ならびに／または、アプリケーションサーバへウェブページ要求および／もしくは入力データを転送する。

【００３１】

さらに別の実施形態に従うと、サーバコンピュータ１３０の１つ以上は、ファイルサーバとして機能し得、ならびに／または、ユーザコンピュータ１１０の１つおよび／もしくはサーバコンピュータ１３０の別の１つ上で実行されるアプリケーションによって組み込まれる、本発明の方法を実現するのに必要なファイルの１つ以上を含み得る。代替的には、当業者が理解するように、ファイルサーバは、すべての必要なファイルを含み得、ユーザコンピュータ１１０および／またはサーバコンピュータ１３０の１つ以上によってそのようなアプリケーションが遠隔から呼び出されることを可能にする。なお、本願明細書におけるさまざまなサーバ（たとえばアプリケーションサーバ、データベースサーバ、ウェブサーバ、ファイルサーバなど）について記載される機能は、インプリメンテーションに特有のニーズおよびパラメータに依存して、単一のサーバおよび／または複数の特殊化されるサーバによって実行され得る。

【００３２】

ある実施形態において、システム１００は、１つ以上のデータベース１４０（たとえばデータベース１４０Ａおよび１４０Ｂ）を含み得る。データベース１４０の位置は任意である。すなわち、単に例示として、データベース１４０Ａは、サーバコンピュータ１３０Ａ（および／またはユーザコンピュータ１１０の１つ以上）に対してローカルな（および／または存在している）記憶媒体上に存在し得る。代替的には、データベース１４０Ｂは、（たとえば通信ネットワーク１２０を介して）ユーザコンピュータ１１０およびサーバコンピュータ１３０の１つ以上と通信し得る限り、ユーザコンピュータ１１０およびサーバコンピュータ１３０のいずれかまたはすべてからリモートであり得る。実施形態の特定の集合では、データベース１４０は、当業者が精通しているストレージエリアネットワーク（ＳＡＮ）に存在し得る。（同様に、ユーザコンピュータ１１０およびサーバコンピュータ１３０に起因する機能を実行するための任意の必要なファイルが適切なように、それぞれのコンピュータ上にローカルにおよび／またはリモートに格納され得る。）実施形態の１つの集合において、データベース１４０の１つ以上は、ＳＱＬフォーマットのコマンドに応答してデータを格納、更新および抽出するよう適合されるリレーショナルデータベースであり得る。たとえば、データベース１４０は、上述したように、データベースサーバによって制御および／または維持され得る。

【００３３】

データ統合の概略

図２は、本発明のある実施形態に従ったデータ統合システム２００の簡略ブロック図である。図２は、この開示において示される１つ以上の発明のさまざまな実施形態または実現例を組み込み得るデータ統合システム２００の簡略図である。図２は、本願明細書において開示される発明の実施形態または実現例を単に例示しているだけであり、請求の範囲に記載されるような任意の発明の範囲を限定するべきでない。当業者は、この開示および本願明細書において示される教示を通じて、図において示される実施形態または実現例に対する他の変形例、修正例および／または代替例を認識し得る。

【００３４】

この実施形態において、データ統合システム２００は、情報ソース２０２、情報統合部２０４、および情報宛先部２０６を含む。一般に、情報ソース２０２から情報統合部２０４に情報が流れ、これにより、情報は、情報宛先部２０６によって、消費され、利用可能になり、または別の態様で使用され得る。データフローは一方向または双方向であり得る。いくつかの実施形態において、１つ以上のデータフローがデータ統合システム２００に

存在し得る。

【００３５】

情報ソース２０２は、データを提供するよう構成される１つ以上のハードウェアおよび／またはソフトウェア要素を示す。情報ソース２０２は、データへの直接または間接のアクセスを提供し得る。この実施形態において、情報ソース２０２は、１つ以上のアプリケーション２０８と１つ以上のリポジトリ２１０とを含む。

【００３６】

アプリケーション２０８は、デスクトップアプリケーション、ホストされたアプリケーション、ウェブベースアプリケーション、またはクラウドベースアプリケーションのような従来のアプリケーションを示す。アプリケーション２０８は、１つ以上の所定の目的のために、データを受け取り、処理し、かつ維持するように構成され得る。アプリケーション２０８のいくつかの例は、カスタマーリレーションシップマネジメント（ＣＲＭ：customer relationship management）アプリケーション、金融サービスアプリケーション、管理およびリスクコンプライアンスアプリケーション、人的資本マネージメント（ＨＣＭ：human capital management）調達アプリケーション、サプライチェーンマネジメントアプリケーション、または、プロジェクトもしくはポートフォリオマネージメントアプリケーションなどを含む。アプリケーション２０８は、当該技術において公知のようなさまざま人間が読むことが可能でありマシンが読み取り可能なフォーマットでアプリケーションデータを操作およびエクスポートするために構成される機能を含み得る。アプリケーション２０８はさらに、リポジトリ２１０におけるデータにアクセスするとともに、リポジトリ２１０にデータを格納し得る。

【００３７】

リポジトリ２１０は、データへのアクセスを提供するように構成されるハードウェアおよび／またはソフトウェア要素を示す。リポジトリ２１０は、データの論理的および／または物理的なパーティショニングを提供し得る。リポジトリ２１０は、リポーティングおよびデータ分析をさらに提供し得る。リポジトリ２１０のいくつかの例は、データベース、データウェアハウス、またはクラウドストレージなどを含む。リポジトリは、１つ以上のアプリケーション２０８からのデータを統合することにより作成される中央レポジトリを含み得る。リポジトリ２１０に格納されたデータは、オペレーショナルシステムからアップロードされ得る。データは、ソースにおいて利用可能になる前に付加的なオペレーションを介して渡され得る。

【００３８】

情報統合部２０４は、データ統合サービスを提供するように構成される１つ以上のハードウェアおよび／またはソフトウェア要素を示す。直接または間接のデータ統合サービスが情報統合部２０４において提供され得る。この実施形態において、情報統合部２０４は、データ移行部２１２、データウェアハウジング部２１４、マスターデータマネージメント部２１６、データ同期部２１８、連結部２２０、およびリアルタイムメッセージ部２２２を含む。情報統合部２０４は、データ統合機能を提供する１つ以上のモジュール、サービス、または、ここで示された要素以外の付加的な要素を含み得るということが理解されるであろう。

【００３９】

データ移行部２１２は、データ移行を提供するように構成される１つ以上のハードウェアおよび／またはソフトウェア要素を示す。一般に、データ移行部２１２は、ストレージタイプ、フォーマット、またはシステム同士の間でデータを転送するための１つ以上のプロセスを提供する。データ移行部２１２は通常、移行を達成するようマニュアルまたはプログラムオプションを提供する。データ移行プロシージャにおいて、１つのシステム上のデータまたは１つのシステムによって提供されるデータは、データ抽出およびデータロードのための設計を提供する別のシステムにマッピングされる。データ移行は、１つ以上のフェーズを含み得、当該１つ以上のフェーズとしてはたとえば、第１のシステムのデータ形式を第２のシステムのフォーマットおよび要件に関連させる１つ以上の設計が作成され

る設計フェーズと、データが第１のシステムから読み出されるデータ抽出フェーズと、データクリーニングフェーズと、データが第２のシステムに書き込まれるデータローディングフェーズとがある。いくつかの実施形態において、データ移行は、データが上記のフェーズのうちのいずれかにおいて正確に処理されるかどうか判定するよう、データ検証フェーズを含み得る。

【００４０】

データウェアハウジング部２１４は、リポーティングおよびデータ分析のために使用されるデータベースを提供するように構成される１つ以上のハードウェアおよび／またはソフトウェア要素を示す。データウェアハウスは典型的に、１つ以上の異種のソースからのデータを統合することにより作成されるデータの中央レポジトリとして閲覧される。データウェアハウジング部２１４は、現在のデータの格納と履歴データの格納とを含み得る。データウェアハウジング部２１４は、典型的な抽出、変換、ロード（ＥＴＬ：extract，transform，load）ベースのデータウェアハウスを含み得、これにより、ステージング層、データ統合層およびアクセス層が主な機能を収容する。一例において、ステージング層またはステージングデータベースは、１つ以上の異なるソースデータシステムの各々から抽出される生データを格納する。統合層は、ステージング層からのデータを変換し、オペレーショナルデータストア（ＯＤＳ：operational data store）データベースにこの変換されたデータをしばしば格納することによって、異なるデータセットを統合する。その後、統合データは、しばしばデータウェアハウスデータベースと称されるさらに別のデータベースに移される。当該データは、（しばしば次元と称される）階層グループと、ファクト（fact）およびアグリゲートファクト（aggregate fact）とへ配され得る。アクセス層は、ユーザまたは他のシステムがデータを抽出するのを補助するために提供され得る。データウェアハウスはデータマートへ細分され得、これにより、各データマートはウェアハウスからのデータのサブセットを格納する。いくつかの実施形態において、データウェアハウジング部２１４は、ビジネスインテリジェンスツールと、データを抽出、変換およびリポジトリへロードするツールと、メタデータを管理および抽出するツールとを含み得る。

【００４１】

マスターデータマネージメント部２１６は、データのマスターコピーを管理するように構成される１つ以上のハードウェアおよび／またはソフトウェア要素を示す。マスターデータマネージメント部２１６は、一貫してマスターデータを規定および管理するプロセス、ガバナンス、ポリシー、スタンダードおよびツールのセットを含み得る。マスターデータマネージメント部２１６は、マスターデータの信頼すべきソースを作成するために、重複を除去し、データを標準化し、ルールを組み込んでシステムに誤ったデータが入ることを除去するための機能を含み得る。マスターデータマネージメント部２１６は、データを収集、集合、マッチング、統合、品質保証、持続、組織の全体にわたって配分するためのプロセスを提供し得、これにより、情報の進行中のメンテナンスおよびアプリケーション使用において一貫性および制御を保証する。

【００４２】

データ同期部２１８は、データを同期させるように構成される１つ以上のハードウェアおよび／またはソフトウェア要素を示す。データ同期部２１８は、ソースからターゲットおよびその逆のデータの間の一貫性を確立することを提供し得る。データ同期部２１８はさらに、時間にわたるデータの連続的なハーモナイゼーションを提供し得る。

【００４３】

連結部２２０は、構成要素ソースからのデータの閲覧を統合するように構成される１つ以上のハードウェアおよび／またはソフトウェア要素を示す。連結部２２０は、複数の自律データベースシステムを単一の連結データベースへとトランスピアレントにマッピングし得る。構成要素データベースは、コンピュータネットワークを介して相互に接続されてもよく、地理的に分散されてもよい。連結部２２０は、いくつかの異なるデータベースをマージすることに対する代替案を提供する。連結データベースまたは仮想データベースはたとえば、すべての構成要素データベースの合成を提供し得る。連結部２２０は、異なる

構成要素データベースにおいて実際のデータ統合を提供しなくてもよく、閲覧においてのみ提供してもよい。

【００４４】

連結部２２０は、構成要素データベースが異種混合であっても、単一のクエリーによって、ユーザおよびクライアントが複数の非連続のデータベースにおいてデータを格納および抽出することを可能にする均一なユーザインターフェイスを提供する機能を含み得る。連結部２２０は、関連する構成要素データソースへの提出のためのサブクエリーにクエリーを分解するとともにサブクエリーの結果セットを合成する機能を含み得る。連結部２２０は、サブクエリーへの１つ以上のラッパー（wrapper）を、それらを適切なクエリー言語に変換するよう含み得る。いくつかの実施形態において、連結部２２０は、エクスポートスキーマおよびアクセスオペレーションの発行を通じてデータを連結部の他のメンバーに利用可能にする自律コンポーネントの集合である。

【００４５】

リアルタイムメッセージ部２２２は、リアルタイム制約（たとえばイベントからシステム応答までのオペレーショナルデッドライン）に従ってメッセージサービスを提供するように構成される１つ以上のハードウェアおよび／またはソフトウェア要素を示す。リアルタイムメッセージ部２２２は、厳密な時間制約内における動作または応答を保証する機能を含み得る。一例において、リアルタイムメッセージ部２２２には、１つのデータベースからいくつかのオーダおよび消費者データを取得し、ファイル中に保持されるいくつかの従業員データとそれを組合せ、その後、当該統合データをマイクロソフトＳＱＬサーバ２０００データベースにロードするタスクが課され得る。オーダは、到達する際に分析される必要があるので、リアルタイムメッセージ部２２２は、可能な限りリアルタイムに近い状態でターゲットデータベースにオーダを渡し、可能な限りワークロードを小さく維持するよう新しくかつ変更されたデータのみを抽出し得る。

【００４６】

情報宛先部２０６は、データを格納または消費するように構成される１つ以上のハードウェアおよび／またはソフトウェア要素を示す。この実施形態において、情報宛先部２０６は、データへの直接または間接のアクセスを提供し得る。この実施形態において、情報宛先部２０６は、１つ以上のアプリケーション２２４と１つ以上のリポジトリ２２６とを含む。

【００４７】

アプリケーション２２４は、デスクトップアプリケーション、ホストされたアプリケーション、ウェブベースアプリケーション、またはクラウドベースアプリケーションのような従来のアプリケーションを示す。アプリケーション２２４は、１つ以上の所定の目的のために、データを受け取り、処理し、かつ維持するように構成され得る。アプリケーション２２４のいくつかの例は、カスタマーリレーションシップマネジメント（ＣＲＭ：customer relationship management）アプリケーション、金融サービスアプリケーション、管理およびリスクコンプライアンスアプリケーション、人的資本マネージメント（ＨＣＭ：human capital management）調達アプリケーション、サプライチェーンマネジメントアプリケーション、または、プロジェクトもしくはポートフォリオマネージメントアプリケーションなどを含む。アプリケーション２２４は、当該技術において公知のようなさまざま人間が読むことが可能でありマシンが読み取り可能なフォーマットでアプリケーションデータを操作およびインポートするために構成される機能を含み得る。アプリケーション２２４はさらに、リポジトリ２２６おけるデータにアクセスするとともに、リポジトリ２２６にデータを格納し得る。

【００４８】

リポジトリ２２６は、データへのアクセスを提供するように構成されるハードウェアおよび／またはソフトウェア要素を示す。リポジトリ２２６は、データの論理的および／または物理的なパーティショニングを提供し得る。リポジトリ２２６は、リポーティングおよびデータ分析をさらに提供し得る。リポジトリ２２６のいくつかの例は、データベース

、データウェアハウス、またはクラウドストレージなどを含む。リポジトリは、１つ以上のアプリケーション２２６からのデータを統合することにより作成される中央レポジトリを含み得る。リポジトリ２２６に格納されたデータは、情報統合部２０４を通じてアップロードまたはインポートされ得る。データは、宛先にて利用可能になる前に付加的なオペレーションを介して渡され得る。

【００４９】

データ統合システム

図３は、本発明の実施形態に従ったデータ統合システム２００を実現するために使用され得るハードウェア／ソフトウェアスタックの簡略ブロック図である。図３は、本願明細書において開示される発明の実施形態または実現例を単に例示しているだけであり、請求の範囲に記載されるような任意の発明の範囲を限定するべきでない。当業者は、この開示および本願明細書において示される教示を通じて、図において示される実施形態または実現例に対する他の変形例、修正例および／または代替例を認識し得る。この実施形態に従ったデータ統合システム２００において発見されるコンポーネントの一例は、カリフォルニア州レッドウッドショアズのオラクル社によって提供される製品のオラクルフュージョンミドルウェア（ORACLE FUSION Middleware）ファミリーのメンバーであるオラクルデータインテグレータ（ORACLE DATA INTEGRATOR）を含んでもよい。オラクルデータインテグレータは、セットベースのデータ統合タスクを実行するために１つ以上のデータベースを使用するＪａｖａベースのアプリケーションである。さらにオラクルデータインテグレータは、データを抽出し、変換されたデータをウェブサービスおよびメッセージを通じて提供し、サービス指向のアーキテクチャにおいてイベントに応答および作成する統合プロセスを作成し得る。オラクルデータインテグレータは、従来のＥＴＬ［抽出－変換－ロード（extract-transform-load）］アーキテクチャではなく、ＥＬＴ［抽出－ロードおよび変換（extract-Load and Transform）］アーキテクチャに基づく。オラクルデータインテグレータのためのユーザマニュアルのコピーが、この開示に添付されており、本願明細書においてすべての目的のために参照により援用される。

【００５０】

さまざまな実施形態において、データ統合システム２００は、データ変換および統合プロセスの規定への新しい宣言型設計アプローチ（declarative design approach）を提供し、より速くより簡易な開発およびメンテナンスが得られる。したがって、データ統合システム２００は、インプリメンテーションの詳細から宣言型ルールを分離する。データ統合システム２００はさらに、データ変換および検証プロセスの実行のためのユニークなＥ－ＬＴアーキテクチャ（抽出－ロード変換（Extract-Load Transform））を提供する。実施形態におけるこのアーキテクチャは、スタンドアロンのＥＴＬサーバおよびプロプラエタリエンジンの必要性を除去する。その代りに、いくつかの実施形態において、データ統合システム２００は、ＲＤＢＭＳエンジンの固有のパワーをレバレッジする。

【００５１】

いくつかの実施形態において、データ統合システム２００は、オラクルフュージョンミドルウェアプラットフォームのような１つ以上のミドルウェアソフトウェアパッケージに統合し、ミドルウェアスタックのコンポーネントになる。図３に示されるように、データ統合システム２００は、ＪａｖａＥＥアプリケーションとしてランタイムコンポーネントを提供し得る。

【００５２】

この例において、データ統合システム２００の１つのコンポーネントはリポジトリ３０２である。リポジトリ３０２は、ＩＴインフラストラクチャに関する構成情報、すべてのアプリケーションのメタデータ、プロジェクト、シナリオおよび実行ログを格納するように構成されるハードウェアおよび／またはソフトウェア要素を示す。いくつかの局面において、たとえばデベロップメント、ＱＡ、ユーザ、アクセプタンスおよびプロダクションといった、リポジトリ３０２の複数のインスタンスは、ＩＴインフラストラクチャにおいて共存し得る。リポジトリ３０２は、メタデータおよびシナリオを交換するいくつかの分

離された環境を可能にするように構成される（たとえば、デベロップメント環境、テスト環境、メンテナンス環境、およびプロダクション環境）。リポジトリ３０２はさらに、オブジェクトがアーカイブされバージョン番号を割り当てられるバージョン制御システムとして動作するように構成される。

【００５３】

この例において、リポジトリ３０２は、少なくとも１つのマスターリポジトリ３０４および１つ以上のワークリポジトリ３０６から形成される。データ統合システム２００内での使用のために開発または構成されるオブジェクトは、これらのリポジトリタイプの１つに格納され得る。一般に、マスターリポジトリ３０４は、ユーザ、プロファイルおよび権限を含むセキュリティ情報と、技術、サーバ定義、スキーマ、コンテキスト、および言語などを含むトポロジー情報と、またバージョン付けおよびアーカイブされたオブジェクトといった情報を格納する。１つ以上のワークリポジトリ３０６は、実際の開発されたオブジェクトを含み得る。

【００５４】

いくつかのワークリポジトリは、（たとえば別個の環境を有するか、または、特定のバージョニングライフサイクルにマッチするよう）データ統合システム２００において共存し得る。１つ以上のワークリポジトリ３０６は、スキーマ定義、データストア構造およびメタデータ、フィールドおよびカラム定義、データ品質制約、相互参照、ならびにデータリネージなどを含む、モデルについての情報を格納する。１つ以上のワークリポジトリ３０６はさらに、ビジネスルール、パッケージ、プロシージャ、フォルダ、ナレッジモジュール、および変数などを含むプロジェクトと、シナリオ、スケジューリング情報およびログを含むシナリオ実行とを格納し得る。いくつかの局面において、１つ以上のワークリポジトリ３０６は、（典型的にプロダクション目的のために）実行情報のみを含み得、実行リポジトリとして指定され得る。

【００５５】

さまざまな実施形態において、リポジトリ３０２は、１つ以上のＥＴＬプロジェクトを格納する。ＥＴＬプロジェクトは、ソースまたはターゲットにおけるデータのデータ属性をモデリングする１つ以上のデータモデルを規定またはそうでなければ特定する。ＥＴＬプロジェクトはさらに、データを移動および変換するために、データ品質制御と、マッピングを規定することとを提供する。データの完全性制御はデータの全体的な一貫性を保証する。アプリケーションデータは、特定のソースまたはターゲットによって課された制約および宣言型ルールに必ずしも有効ではない。たとえば、オーダが顧客を伴わないことが分かった、または、オーダラインが製品を伴わないことが分かったなどである。データ統合システム２００は、これらの制約違反を検出し、再利用または報告目的のためにそれらを格納するよう作業環境を提供する。

【００５６】

データ統合システム２００のいくつかの実施形態において、スタティック制御およびフロー制御という２つの異なるタイプの制御が存在する。スタティック制御は、アプリケーションデータの完全性を検証するために使用されるルールの存在を示す。これらのルール（制約と称される）のうちのいくつかは、（一次キー、基準制約などを使用して）データサーバにおいて既に実現されている場合がある。データ統合システム２００は、付加的な制約の定義およびチェックを、ソースにおいてそれらを直接的に宣言することなく可能にする。フロー制御は、自身の宣言型ルールを実現する変換および統合プロセスのターゲットに関する。フロー制御は、ターゲットにデータをロードする前に、これらの制約に従ってアプリケーションの入力データを検証する。フロー制御プロシージャは一般にマッピングと称される。

【００５７】

ＥＴＬプロジェクトは、ランタイム環境における実行のために展開され得るパッケージへと自動化され得る。したがって、パッケージにおける異なるステップ（マッピングおよびプロシージャなど）の実行をシーケンス処理するとともに、これらのステップの各々に

ついて既存のコードを含んでいるプロダクションシナリオを作り出すことによって、データ統合フローの自動化が達成される。パッケージは典型的に、実行ダイアグラムへ組織されるステップのシーケンスから構成される。パッケージは、プロダクションのためのシナリオを生成するよう使用されるメインオブジェクトである。それらは、データ統合ワークフローを表わしており、たとえば、データストアまたはモデルに対するリバースエンジニアリングプロセスを開始し、アドミニストレータに電子メールを送信し、ファイルをダウンロードし、それを解凍し、マッピングが実行されなければならない順序を規定し、変化するパラメータで実行コマンドに対して繰り返すループを規定するジョブを実行し得る。

【００５８】

シナリオは、ソースコンポーネント（マッピング、パッケージ、プロシージャ、変数）をプロダクションに配置するよう設計される。このコンポーネントについて、シナリオがコード（ＳＱＬ、シェルなど）の発生から得られる。生成されると、ソースコンポーネントのコードが凍結され、シナリオがワークリポジトリ３０６の１つ以上のようなリポジトリ３０２の内部に格納される。シナリオは、エクスポートされ、その後、異なるプロダクション環境へインポートされ得る。

【００５９】

さまざまな実施形態において、データ統合システム２００は、Ｊａｖａグラフィカルモジュールおよびスケジューリングエージェントによってアクセスされるモジュールの態様でリポジトリ３０２のまわりに組織される。グラフィカルモジュールは、リポジトリ３０２に格納される１つ以上の統合プロセスを設計および構築するよう使用され得る。アドミニストレータ、デベロッパおよびオペレータは、リポジトリ３０２にアクセスするためにデベロップメントスタジオを使用し得る。エージェントは、リポジトリ３０２に格納される統合プロセスに関連付けられる統合タスクのセットをスケジューリングおよび調整するために使用され得る。たとえばランタイムにおいて、デスクトップ、ウェブサービス上に展開されるか、または、ソースと通信するエージェントは、１つ以上の統合プロセスの実行を調整する。エージェントは、マスターリポジトリ３０４に格納されるコードを抽出し、さまざまなソースおよびターゲットシステムに接続し、全体的なデータ統合プロセスまたはシナリオを取りまとめ得る。

【００６０】

この実施形態において、データ統合システム２００は、上で論じたグラフィカルモジュールおよび／またはエージェントの１つ以上を含み得るデスクトップ３０８を含む。デスクトップ３０８は、パーソナルコンピュータ、ラップトップ、ネットブック、およびタブレットなどのような１つ以上のデスクトップまたはワークステーションコンピューティングデバイスを示す。デスクトップ３０８はＪａｖａ仮想マシン（ＪＶＭ）３１０およびオラクルデータインテグレータ（ＯＤＩ：Oracle Data Integrator）スタジオ３１２を含む。Ｊａｖａ仮想マシン（ＪＶＭ）３１０は、Ｊａｖａバイトコードを実行し得る仮想マシンである。ＪＶＭ３１０はほとんどの場合、既存のオペレーティングシステム上で実行するよう実現されるが、ハードウェア上で直接的に実行するよう実現され得る。ＪＶＭ３１０は、Ｊａｖａバイトコードが実行され得るランタイム環境を提供し、ランタイムウェブサービス（ＷＳ）３１４およびエージェント３１６のような機能を可能にする。ＪＶＭ３１０は、Ｊａｖａクラスライブラリと、Ｊａｖａアプリケーションプログラミングインターフェイス（ＡＰＩ：application programming interface）を実現する（Ｊａｖａバイトコードでの）スタンダードクラスライブラリのセットと、Ｊａｖａランタイム環境（ＪＲＥ：Java Runtime Environment）を形成する他の要素とを含み得る。

【００６１】

エージェント３１６は、リポジトリ３０２に格納された１つ以上の統合プロセスに関連付けられる統合タスクのセットをスケジューリングおよび調整するように構成される。たとえば、ランタイムにおいて、エージェントは、統合プロセスの実行を調整する。エージェントは、マスターリポジトリ３０４に格納されるコードを抽出し、さまざまなソースおよびターゲットシステムに接続し、全体的なデータ統合プロセスまたはシナリオを取りま

とめ得る。

【００６２】

図３を再び参照して、ＯＤＩスタジオ３１２はデータ統合プロジェクトを設計するよう構成されるハードウェアおよび／またはソフトウェア要素を含む。この例において、ＯＤＩスタジオ３１２は、データ統合プロジェクトを作成および管理するために使用される４つのグラフィカルモジュールまたはナビゲータ、すなわち、デザイナーモジュール３１８、オペレータモジュール３２０、トポロジーモジュール３２２およびセキュリティモジュール３２４を含む。デザイナーモジュール３１８は、データストア（テーブル、ファイルおよびウェブサービスなど）、データマッピング、およびパッケージ（マッピングを含む統合ステップのセット）を規定するように構成されるモジュールである。さまざまな実施形態において、デザイナーモジュール３１８は、データ変換およびデータ完全性について宣言型ルールを規定する。したがって、プロジェクトデベロップメントがデザイナーモジュール３１８において発生する。さらに、デザイナーモジュール３１８において、データベースおよびアプリケーションメタデータがインポートおよび規定される。デザイナーモジュール３１８は、一実施形態において、メタデータおよびルールを使用して、プロダクションについてデータ統合シナリオまたはロードプランを生成する。一般に、デザイナーモジュール３１８は、データ完全性チェックを設計するように使用されるとともに、たとえば既存のアプリケーションまたはデータベースの自動リバースエンジニアリング、変換および統合マッピングのグラフィカルデベロップメントおよびメンテナンス、マッピングにおけるデータフローの可視化、自動ドキュメンテーション生成、および生成されたコードのカスタマイゼーションのような変換を構築するように使用される。

【００６３】

オペレータモジュール３２０は、プロダクション統合ジョブを閲覧および管理するように構成されるモジュールである。したがって、オペレータモジュール３２０は、プロダクションにおけるデータ統合プロセスを管理および監視し、エラーカウントを有する実行ログ、処理されたロウの数、実行統計、および実行される実際のコードなどを示し得る。設計時において、デベロッパはさらに、デザイナーモジュール３１８に関連して、デバッグ目的のためにオペレータモジュール３２０を使用し得る。

【００６４】

トポロジーモジュール３２２は、データソースおよびエージェントへの接続を作成および管理するように構成されるモジュールである。トポロジーモジュール３２２は、インフラストラクチャの物理的および論理的アーキテクチャを規定する。インフラストラクチャまたはプロジェクトアドミニストレータは、トポロジーモジュール３２２を通じて、サーバと、データベーススキーマおよびカタログと、エージェントとをマスターリポジトリに登録し得る。セキュリティモジュール３２４は、ユーザおよびそれらのリポジトリ権限を管理するように構成されるモジュールである。

【００６５】

一般に、ユーザまたはプロセスは、ソースおよびターゲット３２６について１つ以上のデータ統合プロセスを有するデータ統合プロジェクトを作成するよう、デザイナーモジュール３１８と相互作用する。各データ統合プロセスは、少なくとも１つのデータ統合タスクを含む。いくつかの実施形態において、データ統合タスクは、データのどのビットが変換され他のビットと組み合わされるかと、当該データが実際にどのように抽出およびロードされるかの技術的な詳細とを示すビジネスルールのセットによって規定される。好ましい実施形態において、データ統合タスクは、データマッピングを構築するために宣言型のアプローチを使用して特定される。マッピングは、ターゲットと称される、１つのデータストアにポピュレートするオブジェクトであり、１つ以上の他のデータストアからのデータはソースとして既知である。一般に、ソースデータストアにおけるカラムは、マッピングを通じてターゲットデータストアにおけるカラムにリンクされる。マッピングは、パッケージステップとしてパッケージに加えられ得る。上で論じたように、パッケージはデータ統合ジョブを規定する。パッケージは、プロジェクトの下に作成され、各々がマッピン

グまたはプロシージャであり得るステップの組織されたシーケンスから構成される。パッケージは、１つの入口点および複数の出口点を有し得る。

【００６６】

いくつかの実施形態において、新しいマッピングを作成する場合、デベロッパまたは技術的なビジネスユーザは、どのデータが統合されるかと、どのビジネスルールを使用するべきかとをまず規定するよう、デザイナー３１８と相互作用する。たとえば、デベロッパは、どのテーブルが連結されるべきか、どのフィルタが適用されるべきか、データを変換するためにどのＳＱＬエクスプレッションが使用されるべきかを特定し得る。使用されるＳＱＬの特定の言語は、コードが実行されるべきデータベースプラットホームによって決定される。その後、別個のステップにおいて、技術スタッフは、デザイナー３１８と相互に作用して、このデータを抽出、組合せ、次いで統合するのに最も効率的な方法を選ぶ。たとえば、技術スタッフは、インクリメンタルロード（incremental load）、バルクローティングユーティリティ（bulk-loading utility）、スローリー・チェンジング・ディメンション（slowly changing dimension）、および変更データキャプチャ（changed-data capture）といった、データベースに特有のツールおよび設計技術を使用し得る。

【００６７】

この実施形態において、マッピングはソースおよびターゲット３２６について作成され得る。ソースおよびターゲット３２６は、１つ以上のレガシーアプリケーション３２８と、１つ以上のファイル／ＸＭＬ文書３３０と、１つ以上のアプリケーション３３２と、１つ以上のデータウェアハウス（ＤＷ）と、ビジネスインテリジェンス（ＢＩ）ツールおよびアプリケーションと、エンタープライズプロセスマネージメント（ＥＰＭ：enterprise process management）ツールおよびアプリケーション３３４と、（ランタイムウェブサービス３４０およびエージェント３４２を含む）１つ以上のＪＶＭ３３６とを含み得る。

【００６８】

図４は、本発明のさまざまな実施形態においてデータ統合シナリオが作成され得るさまざまな異種混合のデータソースを有する環境４００のブロック図である。この例において、環境４００はＯＤＩスタジオ３１２およびリポジトリ３０２を含む。リポジトリ３０２は、統合シナリオ４００を生成するのに必要とされるメタデータのすべてを含む。ユーザまたはプロセスは、データ完全性制御４０２および宣言型ルール４０４を使用して統合シナリオ４００を作成するようＯＤＩスタジオ３１２と相互作用する。

【００６９】

オーダアプリケーション４０６は、顧客のオーダをトラッキングするためのアプリケーションを示す。「オーダアプリケーション」データモデルは、オーダアプリケーション４０６に格納されたデータと、任意のデータ完全性制御または条件とを表わすよう作成される。たとえば、「オーダアプリケーション」データモデルは、ハイパーストラクチャードクエリランゲッジ（ＨＳＱＬ：Hyper Structured Query Language）インターフェイスに基づき得、ＳＲＣ＿ＣＩＴＹ、ＳＲＣ＿ＣＵＳＴＯＭＥＲ、ＳＲＣ＿ＯＲＤＥＲＳ、ＳＲＣ＿ＯＲＤＥＲ＿ＬＩＮＥＳ、ＳＲＣ＿ＰＲＯＤＵＣＴおよびＳＲＣ＿ＲＥＧＩＯＮという５つのデータストアを含む。

【００７０】

パラメータファイル４０８は、販売員のリストおよび年齢のセグメンテーションから年齢の範囲を含むプロダクションシステムから発行されるフラットファイル（たとえばＡＳＣＩＩ）を示す。この例において、「パラメータ」データモデルが、フラットファイルにおけるデータを表わすために作成される。たとえば、「パラメータ」データモデルは、ファイルインターフェイスに基づき得、ＳＲＣ＿ＳＡＬＥＳ＿ＰＥＲＳＯＮおよびＳＲＣ＿ＡＧＥ＿ＧＲＯＵＰという２つのデータストアを含み得る。

【００７１】

セールスアドミニストレーションアプリケーション４１０はセールスをトラッキングするためのアプリケーションを示す。セールスアドミニストレーションアプリケーション４１０は、オーダアプリケーション４０６およびパラメータファイル４０８からのデータの

変換でポピュレートされたデータウェアハウスであり得る。「セールスアドミニストレーション」データモデルは、セールスアドミニストレーションアプリケーション４１０に格納されるデータと、任意のデータ完全性制御または条件または変換とを表わすよう作成される。たとえば、「セールスアドミニストレーション」データモデルはハイパーストラクチャードクエリラングエッジ（ＨＳＱＬ）インターフェイスに基づき得、ＴＲＧ＿ＣＩＴＹ、ＴＲＧ＿ＣＯＵＮＴＲＹ、ＴＲＧ＿ＣＵＳＴＯＭＥＲ、ＴＲＧ＿ＰＲＯＤＵＣＴ、ＴＲＧ＿ＰＲＯＤ＿ＦＡＭＩＬＹ、ＴＲＧ＿ＲＥＧＩＯＮ、およびＴＲＧ＿ＳＡＬＥという６つのデータストアを含み得る。

【００７２】

図５Ａおよび図５Ｂは、データ統合システム２００によって実行され得る従来のデータ統合処理における簡素化されたデータフローを示す。この例において、オーダアプリケーション４０６、パラメータファイル４０８、１つ以上の他の随意または付加的なソースからのデータは、セールスアドミニストレーションアプリケーション４１０にターゲットとされる従来のＥＴＬプロセスを通じて流れる。データ変換は別個のＥＴＬサーバ５００において発生する。シナリオは専用のリソースまたはプロプラエタリリソースを必要とし、パフォーマンスがより貧弱になり、高コストを引き起こす。

【００７３】

図６Ａおよび図６Ｂは、本発明の実施形態に従った、データ統合システム２００によって実行され得る次世代データ統合処理における簡素化されたデータフローを示す。この例において、オーダアプリケーション４０６、パラメータファイル４０８、１つ以上の他の随意または付加的なソースからのデータは、セールスアドミニストレーションアプリケーション４１０にターゲットとされるＥ－ＬＴプロセスを通じて流れる。データ変換は既存のリソースをレバレッジし、パフォーマンスおよび効率がより高くなる。上述したように、先のＥＴＬシステムは、データ変換を実行するために専用および／またはプロプラエタリインフラストラクチャを必要とした。これは、部分的には、未知のユーザインフラストラクチャに対応するためになされた。たとえば、どのタイプのデータベースが使用されているか知ることがなければ、先行のＥＴＬシステムは、どの変換オペレーションが所与のシステムにおいて利用可能であるか予想することができなかった。しかしながらこれは、如何なる専用および／またはプロプラエタリインフラストラクチャなしで適切なデータ変換を実行することができるユーザの既存データベースおよびサーバのようなリソースが十分に利用されないことになる。

【００７４】

実施形態に従うと、本発明は、ユーザの特定のニーズに従ってユーザがデータ統合プロセスをカスタマイズすることを可能にすることによってユーザの既存のインフラストラクチャをレバレッジする。たとえば、データ統合プランが設計されると、データ統合プランは、実行単位と称される、単一のシステムによって実行可能である個別の部分に分割され得る。ひとたびデータ統合プランが複数の実行単位に分割されると、ユーザのインフラストラクチャおよびシステムリソースに基づいて、ユーザには物理的なプランが提示され得る。このプランはさらにユーザによって、どのユーザシステムがどの実行単位を実行するかを変更するようカスタマイズされ得る。たとえば、ジョインオペレーションが第１のデータベース上で実行されるプランがユーザには提示され得、ユーザは、第２のデータベースにジョインオペレーションを移動させることによって当該プランをカスタマイズし得る。

【００７５】

図６Ｂに示されるように、これにより、先行のＥＴＬシステムを特徴付けたスタンドアロン変換サーバに依存しない抽出－ロード－変換（Ｅ－ＬＴ：extract-load-transform）アーキテクチャが得られる。代わりに、上述したように、データ変換はユーザの既存のインフラストラクチャ上で実行され得る。Ｅ－ＬＴアーキテクチャは、プロプラエタリ変換サーバを取得および維持することに関連付けられるコストを低減しつつユーザにさらに大きなフレキシビリティを提供する。

【００７６】
図３を再び参照すると、エージェントは統合プロセスに関連付けられる統合タスクのセットをスケジューリングおよび調整するために使用され得る。たとえば、ランタイムにおいて、エージェントは、統合プロセスの実行を調整する。エージェントは、マスターリポジトリ３０４に格納されるコードを抽出し得、さまざまなソースおよびターゲットシステムに接続し得、全体的なデータ統合プロセスまたはシナリオを取りまとめる。さまざまな実施形態において、２つのタイプのエージェントが存在する。一例において、スタンドアロンのエージェントがエージェント３１６のようなデスクトップ３０８上にインストールされる。別の例において、アプリケーションサーバエージェントは（オラクルＷｅｂＬｏｇｉｃサーバの上で展開されるＪａｖａＥＥエージェントのように）アプリケーションサーバ３２６上で展開され得、高いアベイラビリテイ要件についてのクラスタリングのようなアプリケーションサーバレイヤーフィーチャから利益を得ることができる。さらに別の例において、エージェントは、エージェント３４２のように、ソースおよびターゲット３２６上で展開され得る。

【００７７】
この実施形態において、データ統合システム２００は、上で論じられるエージェントの１つ以上を含み得るアプリケーションサーバ３４４を含む。アプリケーションサーバ３４４は、１つ以上のアプリケーションサーバ、ウェブサーバ、またはホストされたアプリケーションを示す。この例において、アプリケーションサーバ３４４は、ＦＭＷコンソール３４６、サーブレットコンテナ３４８、ウェブサービスコンテナ３５０、およびデータソース接続プール３５２を含む。

【００７８】
ＦＭＷコンソール３４６は、サーブレットコンテナ３４８、ウェブサービスコンテナ３５０およびデータソース接続プール３３４に関係する情報のような、アプリケーションサーバ３４４の局面を管理するように構成される１つ以上のハードウェアおよび／またはソフトウェア要素を示す。たとえば、ＦＭＷコンソール３４６は、オラクルＷｅｂＬｏｇｉｃサーバドメインを管理するために使用されるブラウザベースのグラフィカルユーザインターフェイスであり得る。ＦＭＷコンソール３４６は、ＷｅｂＬｏｇｉｃサーバインスタンスを構成、開始、停止し、ＷｅｂＬｏｇｉｃサーバクラスタを構成し、データベース接続性（ＪＤＢＣ）およびメッセージング（ＪＭＳ）のようなＷｅｂＬｏｇｉｃサーバサービスを構成し、ユーザ、グループおよび役割を作成および管理することを含むセキュリティパラメータを構成し、ＪａｖａＥＥアプリケーションを構成および展開し、サーバおよびアプリケーションのパフォーマンスを監視し、サーバおよびドメインのログファイルを閲覧し、アプリケーション展開記述子を閲覧し、選択されたランタイムアプリケーション展開記述子要素を編集する機能を含み得る。いくつかの実施形態において、ＦＭＷコンソール３４６は、プロダクションにおけるデータ統合プロセスへのアクセスをＦＭＷコンソール３４６に提供するＯＤＩプラグイン３５４を含み、エラーカウントを有する実行ログ、処理されたロウの数、実行統計、および実行される実際のコードなどを示し得る。

【００７９】
サーブレットコンテナ３４８は、アプリケーションサーバ３４４の性能を拡張するように構成される１つ以上のハードウェアおよび／またはソフトウェア要素を示す。サーブレットは、ＨＴＭＬフォームから提出されたデータを処理するまたは格納するためにほとんどの場合に使用され、データベースクエリの結果のような動的なコンテンツを提供し、適切な顧客のショッピングカートへ品物を充填するといった、ステートレスＨＴＴＰプロトコルに存在しない状態情報を管理する。サーブレットは典型的に、Ｊａｖａクラスが要求に応答し得るプロトコルである、ＪａｖａサーブレットＡＰＩに準拠するＪａｖａＥＥにおけるＪａｖａクラスである。サーブレットを展開および実行するために、サーブレットコンテナ３４８は、サーブレットと相互作用するウェブサーバのコンポーネントとして使用される。したがって、サーブレットコンテナ３４８は、ウェブサービスコンテナ３５０のパブリックウェブサービス３５６およびデータサービス３５８によって提供される機能

と、データソース接続プール３５２によって提供されるデータプールへのアクセスとを拡張し得る。サーブレットコンテナ３４８はさらに、サーブレットのライフサイクルを管理し、特定のサーブレットにＵＲＬをマッピングし、ＵＲＬリクエスタが正しいアクセス権を有することを保証することを担う。

【００８０】

この例において、サーブレットコンテナ３４８は、ＯＤＩ　ＳＤＫ３６２に関連付けられるＪａｖａＥＥアプリケーション３６０と、ＯＤＩコンソール３６４と、ＪａｖａＥＥエージェント３６８に関連付けられるランタイムウェブサービス３６６とを含む。ＯＤＩ　ＳＤＫ３６２は、データ統合およびＥＴＬ設計のためのソフトウェア開発キット（ＳＤＫ）を提供する。ＯＤＩ　ＳＤＫ３６２は、一般的でありかつ非常に反復的である作業の自動化を可能にし、ユーザが反復のタスクをスクリプトにすることを可能にする。

【００８１】

ＯＤＩコンソール３６４は、リポジトリ３０２へのウェブアクセスを提供するＪａｖａエンタープライズエディション（ＪａｖａＥＥ：Java Enterprise Edition）アプリケーションである。ＯＤＩコンソール３６４は、プロジェクト、モデルおよび実行ログを含むＤｅｓｉｇｎ－Ｔｉｍｅオブジェクトをユーザがブラウズすることを可能にするように構成される。ＯＤＩコンソール３６４は、ユーザがフローマップを閲覧し、すべてのデータのソースを追跡し、データを構築するのに使用される変換を理解するためにフィールドレベルにさらにドリルダウンすることを可能にし得る。さらに、エンドユーザは、ＯＤＩコンソール３６４を通じてシナリオ実行を開始および監視し得る。１つの局面において、ＯＤＩコンソール３６４は、データサーバ、物理および論理スキーマのようなトポロジーオブジェクトを閲覧および編集し、かつ、リポジトリ３０２を管理する能力をアドミニストレータに提供する。

【００８２】

データシナリオ設計および開発

上で論じたように、シナリオは、ソースコンポーネント（マッピング、パッケージ、プロシージャ、変数）をプロダクションに配置するよう設計される。このコンポーネントについて、シナリオがコード（ＳＱＬ、シェルなど）の発生から得られる。シナリオは、エクスポートされ、その後、異なるプロダクション環境へインポートされ得る。

【００８３】

図７は、本発明に従った一実施形態におけるＯＤＩスタジオとデータ統合システムのリポジトリとの間の相互作用の簡略ブロック図である。図７に示される実施形態において、図３のＯＤＩスタジオ３１２は、プロダクションについてデータ統合シナリオ７００を生成するようメタデータおよびルールを使用する。一般に、デザイナーモジュール３１８は、データ完全性チェックを設計するように使用されるとともに、たとえば既存のアプリケーションまたはデータベースの自動リバースエンジニアリング、変換および統合インターフェイスのグラフィカルデベロップメントおよびメンテナンス、インターフェイスにおけるデータフローの可視化、自動ドキュメンテーション生成、および生成されたコードのカスタマイゼーションのような変換を構築するように使用される。

【００８４】

図８は、本発明の実施形態に従った、データ統合シナリオを作成するための方法８００のフローチャートを示す。図８に示された方法８００の実現または処理は、コンピュータシステムまたは情報処理デバイスのようなロジックマシンの中央処理装置（ＣＰＵまたはプロセッサ）によって実行される際にソフトウェア（たとえば命令またはコードモジュール）によって実行され得るか、電子デバイスまたは特定用途向け集積回路のハードウェアコンポーネントによって実行され得るか、または、ソフトウェアおよびハードウェア要素の組合せによって実行され得る。図８に示される方法８００は、ステップ８１０において開始する。

【００８５】

さまざまな実施形態において、ユーザはＯＤＩスタジオ３１２のデザイナーモジュール

３１８とのセッションを開始し、リポジトリ３０２へ接続し得る。ユーザは１つ以上のユーザインターフェイスフィーチャと相互作用して、新しいデータ統合プロジェクトを作成するか、または、たとえばマスターリポジトリ３０４に格納される既存のデータ統合プロジェクトから選択し得る。一般に、デザイナーモジュール３１８は、メタデータを管理し、データ完全性チェックを設計し、変換を構築するよう使用される。さまざまな実施形態において、デザイナーモジュール３１８を通じて取り扱われる主なオブジェクトはモデルおよびプロジェクトである。データモデルは、データソースまたはターゲットにおけるメタデータのすべてを含む（たとえばテーブル、カラム、制約、記述、相互参照など）。プロジェクトは、ソースまたはターゲット（たとえばマッピング、プロシージャ、変数など）についてロードおよび変換ルールのすべてを含む。

【００８６】

ステップ８２０において、１つ以上のデータモデルが作成される。ステップ８３０において、１つ以上のプロジェクトが作成される。図９は、本発明の実施形態に従った、データ統合シナリオを作成するためのユーザインターフェイスのスクリーンショットである。この例において、ナビゲーションパネル９１０は、情報を表示し、データモデルとの相互作用のための機能を含む。ナビゲーションパネル９２０は情報を表示し、プロジェクトとの相互作用のための機能を含む。上で論じたように、ユーザはデータモデルを作成するだけでなく、データモデルにおけるデータについて任意のデータ完全性チェックを開発し得る。さらに、ユーザは、ソースからのデータをターゲットにロードするフローにおいてデータについてのデータ完全性および変換を提供する、プロジェクトについてのインターフェイス、プロシージャ、変数を特定し得る。ステップ８４０では、１つ以上のデータ統合シナリオが生成される。図８はステップ８５０で終了する。

【００８７】

図１０は、本発明の実施形態に従った、マッピングを作成するための方法１０００のフローチャートを示す。図１０に示された方法１０００の実現または処理は、コンピュータシステムまたは情報処理デバイスのようなロジックマシンの中央処理装置（ＣＰＵまたはプロセッサ）によって実行される際にソフトウェア（たとえば命令またはコードモジュール）によって実行され得るか、電子デバイスまたは特定用途向け集積回路のハードウェアコンポーネントによって実行され得るか、または、ソフトウェアおよびハードウェア要素の組合せによって実行され得る。図１０に示される方法１０００は、ステップ１０１０において開始する。

【００８８】

ステップ１０２０において、ターゲットデータストア情報が受け取られる。たとえば、ユーザは、ターゲットデータストア情報を提供するよう、デザイナーモジュール３１８の１つ以上のユーザインターフェイスフィーチャと相互作用し得る。一実施形態において、ユーザは、選択されたデータモデルおよび任意の関連付けられる変換またはデータ完全性チェックの局面を視覚的に表わすマッピングまたはフローパネル上にナビゲーションパネル９１０から、１つ以上のデータモデルを含むターゲットデータストア情報をドラッグアンドドロップし得る。

【００８９】

ステップ１０３０では、ソースデータストア情報が受け取られる。たとえば、ユーザは、ソースデータストア情報を提供するよう、デザイナーモジュール３１８の１つ以上のユーザインターフェイスフィーチャと相互作用し得る。一実施形態において、ユーザは、選択されたデータモデルおよび任意の関連付けられる変換またはデータ完全性チェックの局面を視覚的に表わすターゲットデータストア情報の同じマッピングまたはフローパネル上にナビゲーションパネル９１０から、１つ以上のデータモデルを含むソースデータストア情報をドラッグアンドドロップし得る。

【００９０】

さまざまな実施形態において、ソースデータストア情報およびターゲットデータストア情報は、１つ以上のデータモデルおよび随意にオペレーションから形成され得る。オペレ

ーションのいくつかの例は１つ以上のデータセットオペレーション（たとえばユニオン、ジョイン、インターセクションなど）、データ変換、データフィルタオペレーション、制約、記述、相互参照、または完全性チェックなどを含み得る。さらに別の実施形態において、これらのオペレーションのうちのいくつかは、あらかじめ構成されるとともに、デザイナーモジュール３１８において視覚的に表わされ得る。他の実施形態において、カスタムオペレーションが提供され得、オペレーションを実現するロジックおよびマッピングなどをユーザが特定することを可能にする。

【００９１】

ステップ１０４０において、マッピング情報が受け取られる。たとえば、ユーザは、ターゲットデータストア情報にソースデータストア情報をマッピングするよう、デザイナーモジュール３１８の１つ以上のユーザインターフェイスフィーチャと相互作用し得る。一実施形態において、ユーザは、ソースデータストア情報におけるデータ要素の属性をターゲットデータストア情報におけるデータ要素の属性に視覚的に接続し得る。これは、ソースデータストア情報およびターゲットデータストア情報におけるテーブルのカラム名をマッチングすることにより行われ得る。さらに別の実施形態において、１つ以上の自動マッピング技術がマッピング情報を提供するために使用され得る。

【００９２】

図１１は、本発明の実施形態に従った、データ統合シナリオにおいてマッピング情報を提供するためのユーザインターフェイスのスクリーンショットである。この例において、パネル１１１０におけるソースデータストア情報の属性が、パネル１１２０におけるターゲットデータストア情報の属性にマッピングされる。

【００９３】

図１０を再び参照して、ステップ１０５０において、データロードストラテジーが受け取られる。データロードストラテジーは、ソースデータストア情報からの実際のデータが抽出フェーズの間にどのようにロードされるべきであるかについての情報を含む。データロードストラテジーは、デザイナー３１８のフロータブにおいて規定され得る。いくつかの実施形態において、データロードストラテジーは、マッピングの構成に依存してフローについて自動的に計算され得る。

【００９４】

たとえば、１つ以上のナレッジモジュールが当該フローのために提案され得る。ナレッジモジュール（ＫＭ）は、異なる技術にわたって再使用可能な変換およびＥＬＴ（抽出、ロード、および変換）ストラテジーを実現するコンポーネントである。１つの局面において、ナレッジモジュール（ＫＭ）はコードテンプレートである。各ＫＭは、全体のデータ統合プロセスにおいて個々のタスクに専用であり得る。ＫＭにおけるコードは、ほとんど置換法で実行されるであろう形で現われ、多くの異なる統合ジョブによって一般的に使用されることを可能にする。生成および実行されるコードは、デザイナーモジュール３１８において規定される宣言型ルールおよびメタデータに由来する。この一例は、オラクルデータベース１０ｇから変更データキャプチャを通じてデータを抽出し、オラクルデータベース１１ｇにおけるパーティショニングされたファクトテーブルに変換データをロードするか、または、マイクロソフトＳＱＬサーバデータベースからのタイムスタンプベースの抽出を作成し、このデータをテラデータエンタープライズデータウェアハウス（Teradata enterprise data warehouse）にロードすることである。

【００９５】

ＫＭの能力はそれらの再使用可能性とフレキシビリティとにあり、たとえば、あるロードストラテジーが１つのファクトテーブルのために開発され得、その後、当該ロードストラテジーが他のすべてのファクトテーブルに適用され得る。１つの局面において、所与のＫＭを使用するすべてのマッピングは、ＫＭに対してなされる任意の変化を引き継ぐ。いくつかの実施形態において、統合ナレッジモジュール（ＩＫＭ：integration knowledge module）、ロードナレッジモジュール（ＬＫＭ：loading knowledge module）、およびチェックナレッジモジュールＣＫＭ（check knowledge module）といったように、５つの異

なるタイプのＫＭが提供され、それらの各々はソースからターゲットまで変換処理における１つのフェーズをカバーする。

【００９６】

図４を参照して、ユーザは、環境４００においてＳＲＣ＿ＡＧＥ＿ＧＲＯＵＰ、ＳＲＣ＿ＳＡＬＥＳ＿ＰＥＲＳＯＮファイルおよびＳＲＣ＿ＣＵＳＴＯＭＥＲテーブルからデータを抽出する方法を規定し得る。ロードストラテジーを規定するために、ユーザは、ＳＲＣ＿ＡＧＥ＿ＧＲＯＵＰファイルのロードに対応するソースセットを選択し、ＳＱＬへのＬＫＭファイルを選択し得、ファイルからＳＱＬまでのフローを実現する。１つの局面において、ＬＫＭはリモートサーバからステージングエリアにソースデータロードすることを担う。

【００９７】

ステップ１０６０では、データ統合ストラテジーが受け取られる。ローディングフェーズを規定した後、ユーザは、ロードされたデータのターゲットへの統合に適合するべきストラテジーを規定する。統合ストラテジーを規定するために、ユーザは、ターゲットオブジェクトを選択し、ＩＫＭ　ＳＱＬインクリメンタルアップデート（IKM SQL Incremental Update）を選択し得る。ＩＫＭは、最終の変換されたデータをターゲットに書き込むことを担う。ＩＫＭは、開始されると、リモートサーバのためのすべてのローディングフェーズは、たとえばすべてのリモートソースデータセットがＬＫＭによってステージングエリアにロードされたといったように、既にそれらのタスクを行なったか、または、ソースデータストアがステージングエリアと同じデータサーバ上に存在するとみなす。

【００９８】

ステップ１０７０では、データ制御ストラテジーが受け取られる。一般に、ＣＫＭは、データセットのレコードが規定された制約と一貫していることをチェックすることを担う。ＣＫＭは、データ完全性を維持するために使用され得、全体的なデータ品質イニシアチブに参加する。ＣＫＭは２つの態様で使用され得る。第１に、既存データの一貫性をチェックするために使用され得る。これは任意のデータストア上またはインターフェイス内で行われ得る。この場合、チェックされたデータは、現在データストアに存在するデータである。第２の場合において、ターゲットデータストアにおけるデータが、ロードされた後でチェックされる。この場合、ＣＫＭは、ターゲットへの書き込みの前に、結果得られたフロー上のターゲットデータストアの制約をシミュレートする。

【００９９】

図１２は、本発明の実施形態に従った、データ統合シナリオにおいてフロー情報を提供するためのユーザインターフェイスのスクリーンショットである。

【０１００】

ステップ１０８０においてインターフェイスが生成される。図１０はステップ１０９０で終了する。

【０１０１】

データ統合シナリオパッケージおよび展開

上で論じたように、パッケージにおいて異なるステップ（マッピングおよびプロシージャなど）の実行をシーケンス処理するとともに、これらのステップの各々について既存のコードを含んでいるプロダクションシナリオを作り出すことによって、データ統合システム２００においてデータ統合フローの自動化が達成され得る。パッケージは、実行ダイアグラムへ組織されるステップのシーケンスから構成される。パッケージは、プロダクションのためのシナリオを生成するよう使用されるメインオブジェクトである。シナリオは、ソースコンポーネント（マッピング、パッケージ、プロシージャ、変数）をプロダクションに配置するよう設計される。このコンポーネントについて、シナリオがコード（ＳＱＬ、シェルなど）の発生から得られる。シナリオは、エクスポートされ、その後、異なるプロダクション環境へインポートされ得る。

【０１０２】

図１３は、本発明の実施形態に従った、パッケージを作成するための方法のフローチャ

ートを示す。図１３に示された方法１３００の実現または処理は、コンピュータシステムまたは情報処理デバイスのようなロジックマシンの中央処理装置（ＣＰＵまたはプロセッサ）によって実行される際にソフトウェア（たとえば命令またはコードモジュール）によって実行され得るか、電子デバイスまたは特定用途向け集積回路のハードウェアコンポーネントによって実行され得るか、または、ソフトウェアおよびハードウェア要素の組合せによって実行され得る。図１３に示される方法１３００は、ステップ１３１０において開始する。

【０１０３】

ステップ１３２０において、パッケージステップ情報が受け取られる。パッケージステップ情報は、ステップ、要素、プロパティ、およびコンポーネントなどを識別する情報を含む。一例において、ユーザは、パッケージについて１つ以上のステップを作成、識別、またはそうでなければ特定するよう、デザイナーモジュール３１８の１つ以上のユーザインターフェイスフィーチャと相互作用し得る。一実施形態において、１つ以上のコンポーネントが選択され、図に配置される。これらのコンポーネントはパッケージにおけるステップとして現われる。

【０１０４】

ステップ１３３０において、パッケージステップシーケンス情報が受け取られる。パッケージステップシーケンス情報は、ステップについてのオーダリング、および従属性などを識別する情報を含む。ひとたびステップが作成されると、ステップがデータ処理チェーンへと順に並べられるかまたは並べ替えられる。一例において、ユーザは、パッケージの１つ以上のステップについてシーケンシングまたはオーダリングを提供するよう、デザイナーモジュール３１８の１つ以上のユーザインターフェイスフィーチャと相互作用し得る。データ処理チェーンは、第１のステップとして規定されたユニークステップを含み得る。一般に、各ステップは、成功または失敗のような１つ以上の終結状態を有する。失敗または成功のようないくつかの状態におけるステップの後には別のステップまたはパッケージの終了が続き得る。１つの局面において、失敗のようないくつかの状態の場合、シーケンス情報は多くの再試行を規定し得る。別の局面において、パッケージはいくつかの可能な終結ステップのみを有し得る。

【０１０５】

図１４は、本発明の実施形態に従った、データ統合シナリオにおいてパッケージシーケンス情報を提供するためのユーザインターフェイスのスクリーンショットである。

【０１０６】

ステップ１３４０において、パッケージが生成される。図１３はステップ１３５０で終了する。

【０１０７】

上で論じたように、データ統合フローの自動化は、パッケージにおける異なるステップ（マッピングおよびプロシージャなど）の実行をシーケンス処理することにより達成され得る。次いで、パッケージのステップの各々について既存のコードを含むプロダクションシナリオのために、パッケージが作り出され得る。さまざまな実施形態において、パッケージはプロダクション環境において自動的に実行されるよう展開される。

【０１０８】

図１５は、本発明の実施形態に従ったデータ統合シナリオを展開するための方法１５００のフローチャートを示す。図１５に示された方法１５００の実現または処理は、コンピュータシステムまたは情報処理デバイスのようなロジックマシンの中央処理装置（ＣＰＵまたはプロセッサ）によって実行される際にソフトウェア（たとえば命令またはコードモジュール）によって実行され得るか、電子デバイスまたは特定用途向け集積回路のハードウェアコンポーネントによって実行され得るか、または、ソフトウェアおよびハードウェア要素の組合せによって実行され得る。図１５に示される方法１５００は、ステップ１５１０において開始する。

【０１０９】

ステップ１５２０において、統合シナリオが抽出される。一実施形態において、パッケージがリポジトリ３０２から抽出される。ステップ１５３０において、統合シナリオは１つ以上のエージェントに展開される。ステップ１５４０において、統合シナリオは１つ以上のエージェントによって実行される。１つの局面において、統合シナリオは、たとえばＯＤＩスタジオ３１２から、コマンドラインから、またはウェブサービスからといったようにいくつかの態様で実行され得る。たとえば、シナリオ実行は、上に論じられるようにオペレータモジュール３２０などを介して閲覧および監視され得る。図１５はステップ１５５０で終了する。

【０１１０】

組み合わされたフローベースのＥＴＬおよびエンティティリレーションシップベースのＥＴＬ

ほとんどのデータ統合システムにおいて、マッピングは、マップの部分を形成するすべての入力および出力属性の明示的な定義を必要とする。典型的なフローベースのＥＴＬツールにおいて、コネクタが属性レベルに形成される。これにより、非常に簡潔なマッピングモデルが得られる。しかしながら、これによりさらに、巨大な数のオブジェクトが生成されるとともに、属性レベルコネクタの数によりマップを構築および維持することが煩雑になる。

【０１１１】

さまざまな実施形態において、データ統合システム２００は、マッピングの設計およびメンテナンスを容易にするための１つ以上の技術を組み込む。コンポーネントは、すべての入力および出力属性を特定する必要なく、単純に既存の設計に加えられ得、コンポーネントレベルのコネクタがリルートされることが可能になる。１つの局面において、データセットとフロー指向の設計との組合せは、変更と共に複雑性を扱うよう提供される。エンティティリレーションシップは、設計の論理ビュー内で特定され得、これにより、データストア、ジョイン、フィルタおよびルックアップが、一般にマップへの変更を必要とすることなく、追加または除去されることが可能になる。

【０１１２】

本願明細書において一般に使用されるようなデータセットは、データストアのグループからのデータフローを表わす。いくつかのデータセットは、ユニオンおよびインターセクトのようなセットベースのオペレータのようなオペレーションを使用してインターフェイスターゲットデータストアへマージされ得る。さまざまな実施形態では、設計の論理ビューにおいて、データセットは追加、除去、配列され得る。したがって、データ統合システム２００は、ユーザが単一のビューにおいてフローベースのＥＴＬとエンティティリレーションシップベースのＥＴＬとを組み合わせることを可能にする。したがって、データ統合システム２００は、マッピングの設計およびメンテナンスを非常に容易にする。データ統合システム２００はさらに、既存の設計へのコンポーネントの追加、典型的には単に必要なレベルコネクタをリルートすることを簡易にする。

【０１１３】

図１６は、本発明に従った一実施形態における組み合わされたフローベースおよびエンティティベースのマッピング１６００の簡略ブロック図である。この例において、マッピング１６００は、データソースＳＲＣ＿ＥＭＰを表わすコンポーネント１６１０と、データセットＤＡＴＡＳＥＴを表わすデータセット１６２０と、データターゲットＴＧＴ＿ＥＭＰＤＥＰＴを表わすコンポーネント１６３０とを含む。データターゲットＴＧＴ＿ＥＭＰＤＥＰＴを更新するために、データソースＳＲＣ＿ＥＭＰおよびＤＡＴＡＳＥＴについてジョインが必要とされる。入力としてコンポーネント１６１０およびデータセット１６２０に接続するとともに出力としてコンポーネント１６３０に接続するＪＯＩＮを表わすコンポーネント１６４０がマッピング１６００に加えられる。コンポーネント１６４０は（ＳＲＣ＿ＥＭＰ．ＤＥＰＴＮＯ＝ＤＡＴＡＳＥＴ．ＤＥＰＴＮＯ）といったジョインエクスプレッションを提供するように構成される。

【０１１４】

従来のデータ統合システムにおいて、マッピング１６００は、ＪＯＩＮを表わすコンポーネント１６４０の部分を形成するすべての入力および出力属性の明示的な定義を必要とする。対照的に、さまざまな実施形態において、マップデベロッパは、コンポーネント１６４０を通って流れているためコンポーネント１６３０に可視である、コンポーネント１６１０によって表わされるデータソースＳＲＣ＿ＥＭＰの属性およびデータセット１６２０によって表わされるＤＡＴＡＳＥＴの属性からデータターゲットＴＧＴ＿ＥＭＰＤＥＰＴのカラムがどのように直接的にポピュレートされるかを提供するようデータセット１６２０におけるエンティティリレーションシップを規定し得る。

【０１１５】

図１７は、本発明の実施形態に従った、組み合わされたフローベースおよびエンティティベースのマッピングを生成するための方法１７００のフローチャートを示す。図１７に示される方法１７００の実現または処理は、コンピュータシステムまたは情報処理デバイスのようなロジックマシンの中央処理装置（ＣＰＵまたはプロセッサ）によって実行される際にソフトウェア（たとえば命令またはコードモジュール）によって実行され得るか、電子デバイスまたは特定用途向け集積回路のハードウェアコンポーネントによって実行され得るか、または、ソフトウェアおよびハードウェア要素の組合せによって実行され得る。図１７に示される方法１７００は、ステップ１７１０において開始する。

【０１１６】

ステップ１７２０において、１つ以上のコンポーネントが受け取られる。上で論じたように、いくつかのタイプのコンポーネントは、マップを通って流れるデータの形に影響する一方、他のタイプのコンポーネントは、データのフローを制御するがフローの形を根本的に変更しない。ステップ１７３０において、１つ以上のデータセットが受け取られる。たとえば、マップデザイナーは、設計からのデータセットを追加、編集、または除去し得る。マップデザイナーは、データセットにおけるさまざまな属性間のエンティティリレーションシップを特定するためにリレーションシップエディタと相互作用し得る。１つの局面において、データ統合システム２００は、設計の下流のコンポーネントに晒されることになる属性を判定するために、規定されたエンティティリレーションシップを抽出するように構成される。ステップ１７４０において、コンポーネントおよびデータセットに基づいてマップが生成される。さまざまな実施形態において、コンポーネントおよびデータセットへの変更を反映するために、設計の論理ビューおよび物理ビューが更新され得る。さまざまな局面において、データ統合システム２００は、フローのデータセットビューにおいてリレーションシップを導き出すことに基づいて物理設計を自動的に生成する。図１７はステップ１７５０で終了する。

【０１１７】

データ統合システム２００はさらに、既存の設計へのコンポーネントおよび他のデータセットを追加することを簡易にし、典型的にはレベルコネクタをリルートすることを必要とするのみである。たとえば、フィルタコンポーネントが設計に加えられると、コンポーネントレベルコネクタの変更は、ある下流のコンポーネントの属性のアサインメントにおける変更を必要としない。別の例において、別のデータセットを追加することによって、マップデザイナーは、マップの設計ビューの内部からエンティティリレーションシップを直接的に特定または宣言することが可能になる。

【０１１８】

図１８は、本発明に従った一実施形態における、データセットビューを有するマッピング１６００の簡略ブロック図である。この例において、コンポーネント１６２０は１つ以上のエンティティ１８１０、１８２０および１８３０を含む。マッピング１６００にエンティティリレーションシップを追加するためには、ユーザは、リレーションシップ１８４０のようなエンティティ属性同士の間のリレーションシップを追加または規定する必要があるだけである。さまざまな実施形態において、そのような変更は、マッピング１６００における如何なる下流のアサインメントへの変更も必要としない。なぜならば、１つ以上のエンティティリレーションシップから得られる出力属性は、設計ビューにおいて提供さ

れる情報から直接的に導き出され得るからである。従来のフローツールでは、カラムレベルにおけるすべてのものは、新しいデータセットの導入によって再リンクされる必要がある。

【０１１９】

図１９Ａおよび図１９Ｂは、本発明に従った一実施形態における組み合わされたフローベースおよびエンティティベースのマッピングについての論理および物理設計の簡略ブロック図である。この例において、図１９Ａのビュー１９１０は、データソースを表わすコンポーネントＡ、Ｂ、およびＣと、論理設計のフロービューにおいてデーターゲットを表わすコンポーネントＴとを含む。コンポーネントＡ、ＢおよびＣは、論理設計のデータセットビューにおいてエンティティリレーションシップを記述するデータセットとして表わされる。したがって、データセットビューにおいてマップクリエイターによって規定されたエンティティリレーションシップから記述されるコンポーネントＴのような下流のコンポーネントから閲覧される際、データセットは、属性の宣言されたセットを有する。コンポーネントＪ１およびＪ２は、データセットビューにおいてコンポーネントの属性間の論理オペレーションを表わす。

【０１２０】

この例において、図１９Ｂのビュー１９２０は、データソースを表わすコンポーネントＡ、Ｂ、およびＣと、物理設計のフロービューにおいてデーターゲットを表わすコンポーネントＴとを含む。属性のセットは、データセットビューにおいて規定されるとともに物理設計を作成するために使用されるエンティティリレーションシップから導き出される。

【０１２１】

図２０は、本発明の実施形態に従った、組み合わされたフローベースおよびエンティティベースのマッピングの物理設計を生成するための方法２０００のフローチャートを示す。図２０に示された方法２０００の実現または処理は、コンピュータシステムまたは情報処理デバイスのようなロジックマシンの中央処理装置（ＣＰＵまたはプロセッサ）によって実行される際にソフトウェア（たとえば命令またはコードモジュール）によって実行され得るか、電子デバイスまたは特定用途向け集積回路のハードウェアコンポーネントによって実行され得るか、または、ソフトウェアおよびハードウェア要素の組合せによって実行され得る。図２０に示される方法２０００は、ステップ２０１０において開始する。

【０１２２】

ステップ２０２０において、コンポーネント定義が受け取られる。たとえば、コンポーネント定義はルール、オペレーション、プロシージャ、変数、およびシーケンスなどを含み得る。ステップ２０３０において、データセット定義が受け取られる。たとえば、マップデザイナーは、論理設計のフロービュー内においてエンティティリレーションシップを追加または編集し得る。ステップ２０４０において、フロー設計からリレーションシップ情報を導き出すことに基づいて物理設計が生成される。図２０はステップ２０５０で終了する。

【０１２３】

したがって、データ統合システム２００はユーザがプラットホームおよび技術に依存しない論理設計を作成することを可能にする。ユーザは、どのようにデータがソースとターゲットとの間に流れることをユーザが望むかをハイレベルで規定する論理設計を作成し得る。ユーザのインフラストラクチャを考慮して、ツールが論理設計を分析し、物理的設計を作成し得る。論理設計は、設計における各ソースおよびターゲットに対応する複数のコンポーネントと、ジョインまたはフィルタのようなオペレーションと、アクセスポイントとを含み得る。物理的設計に転送された際の各コンポーネントは、データに対してオペレーションを行なうようコードを生成する。存在する技術（たとえばＳＱＬサーバ、オラクル、Ｈａｄｏｏｐなど）と使用される言語（ＳＱＬ、ｐｉｇなど）とに依存して、各コンポーネントによって生成されるコードは異なり得る。

【０１２４】

したがって、データ統合システム２００のユーザは、論理設計においてデータセットコンポーネントをあらかじめ規定する必要はない。データ統合システム２００は、マップデザイナーが論理設計のデータセットビューにおいてエンティティリレーションシップを宣言することを可能にするツールを提供する。データ統合システム２００は、所定のコンポーネントタイプによって表わされるオペレーションにてどの属性が必要かを決定することができる。これは設計およびメンテナンスの両方を簡素化する。

【０１２５】

エンティティリレーショナルモデリング

リレーショナルデータベース設計は、エンティティリレーショナルモデリングまたはＥ－Ｒモデリングに基づいている。従来、Ｅ－Ｒ設計は、問題ドメインの静的な構成を記述するために使用されている。データストアからデータを抽出してそれらを「マッサージ」してある構造にするといったより動的な局面は、一般に異なる問題と考えられる。１９９０年代中盤以降、これらのいわゆる「ＥＴＬツール」に対して着実な取り組みがなされてきた。ＥＴＬツールは、一般にＥＴＬモデルと称される、動的データフローに関する仕様を人間のデザイナーが作成するのを支援することが可能である。

【０１２６】

図２１は、静的なＥ－Ｒモデルと動的なＥＴＬモデルとの間のリレーションシップを示す図である。１つの興味深い疑問は、ＥＴＬ設計プロセスにおいて、示される人的要因を除去することが可能かどうかである。代替的には、別の態様でこの疑問を表すと、人間の介入なしで動的なデータフローモデルが自動的に形成され得るようにＥ－Ｒモデルは十分なオペレーショナル情報を含んでいるか？ということである。

【０１２７】

Ｅ－Ｒモデルを使用してＥＴＬ設計プロセスを自動化することから多くの恩恵が存在する。１つのそのような恩恵はＥＴＬデザイナーの生産性である。Ｅ－Ｒモデルは、ＥＴＬプロセスより容易に補正され得る。Ｅ－Ｒモデルはさらに、データベースエンジニアが理解する標準的な表記法を有する。しかしながら、如何なるＥＴＬツールについても同じことは言えない。すべてでなければ大部分は、デザイナーの側での急な学習曲線を必要とする。別の恩恵は、変更に対するより良好な適応性である。Ｅ－Ｒモデルが終了する場合、「仲介者（middleman）」なしで、ＥＴＬプロセスも終了する。

【０１２８】

さまざまな実施形態において、Ｅ－ＲモデルからＥＴＬモデルへの自動的な変換を提供するよう技術が開示される。これは、データベースエンジニアがＥ－Ｒダイアグラムを読むと、データフローモデルが彼の頭において通常構築されるという所見に基づく。この暗黙のデータフローモデル（silent data flow model）を使用して、エンジニアはＥ－Ｒモデルを理解し得、他者とコミュニケーションすることができる。エンジニアはさらに、このモデルに基づいてソフトウェアを作成する。Ｅ－Ｒモデルが複雑になる場合、この現象はより明らかである。したがって、発明者は、すべてのＥ－Ｒモデルにおいて１つ以上の隠されたデータフローモデルが存在し得ることを認識している。１つの局面において、自動的な変換システムの作成をガイドする際に正確であると証明されたＥ－Ｒモデルについて等価なデータフローモデルが提供される。

【０１２９】

図２２は、一実施形態における自動変換システム２２００のトップレベルの設計チャートを提供する図である。図２２は、本願明細書において開示される発明の実施形態または実現例を単に例示し得るだけであり、請求の範囲に記載されるような任意の発明の範囲を限定するべきでない。当業者は、この開示および本願明細書において示される教示を通じて、図において示される実施形態または実現例に対する他の変形例、修正例および／または代替例を認識し得る。

【０１３０】

図２２に示されるように、Ｅ－Ｒモデルは、「ユーザディレクティブ（user directive）」のセットとともに、自動変換システム２２００に入力として提供される。その後、自

動変換システム２２００は、ＥＴＬの目的のために等価なデータフローモデルを作成する。本願明細書において使用されるように、「ユーザディレクティブ」は、ユーザがデータフローモデルの計算を考慮に入れることを期待する要件のセットである。たとえば、ユーザは、論理の考慮、性能の考慮、またはセキュリティの考慮などにより、バイナリリレーションシップの連なりについて特定のオーダを要求し得、指定されたマシン／位置についてリレーションシップを処理することを要求し得る。

【０１３１】

本願明細書において使用されるような「等価なデータフロー」モデルは、Ｅ－Ｒモデルについてセマンティックモデルを表わす。セマンティックモデルは、論理モデルが何を意味するかを明白に規定するために使用される。セマンティックモデルは、自然言語、集合論表記法、代数方程式、数学論理またはアルゴリズム表記法（一般にオペレーショナルセマンティックとして公知）といった非常に異なる態様で表現され得る。さまざまな実施形態において、Ｅ－Ｒモデルについてのセマンティックモデルは、「ＣＦＯモデル」と称されるオペレーショナルセマンティックモデルである。１つの局面において、オペレーショナルセマンティックフォーマットで意味を規定することは、二重の恩恵を提供する。第１に、オペレーショナルセマンティックモデルは既に、データフローモデルと一致するステップバイステップフォームにあるという恩恵である。第２に、オペレーショナルセマンティックモデルは、他のフォーマルなセマンティックモデルと比較して、人間にとって理解するのが容易であり、自然言語の説明より正確であるという恩恵である。

【０１３２】

Ｅ－Ｒモデル（またはダイアグラム）におけるバイナリリレーションシップは、単純にＥＴＬモデルにおけるジョインにマッピングされ得る。しかしながら、マルチウェイリレーションシップは何らかの作業を必要とする。なぜならば、それについては一般的な誤認（misconception）があるからである。図２３Ａおよび図２３Ｂは、２つの一般的なＥ－Ｒ表記法（E-R notation）での３ウェイリレーションシップを示す。図２３Ａを参照して、モデル２３１０は、標準的なＥ－Ｒ表記法を使用して描かれるかまたは別の態様で表わされる。この例において、モデル２３１０は３つのエンティティＰＥＴ，ＰＥＴ＿ＴＹＰＥ，ＰＥＴ＿ＯＷＮＥＲを含んでおり、「Ｐｅｔ－ｏｆ－Ｔｙｐｅ－ａｎｄ－Ｏｗｎｅｒ」と呼ばれる３ウェイリレーションシップにおいて関係付けられている。図２３Ａの直観的理解は、これらの３つのエンティティが同時に相互作用し得るということである。

【０１３３】

しかしながら、実際には、標準的なＥ－Ｒ表記法は使用されない。代わりに、いわゆる「カラスの足（Crow's Feet）」表記法を見ることがより一般的である。これらの２つの表記法の間の違いは単に表面的である。図２３Ｂを参照して、モデル２３２０はカラスの足表記法を使用して描かれるかまたは別の態様で表わされる。この例においても、モデル２３２０は、ＰＥＴ，ＰＥＴ＿ＴＹＰＥ，ＰＥＴ＿ＯＷＮＥＲの３つのエンティティを含んでおり、真ん中におけるコーナー線を有するボックスとして示される「Ｐｅｔ－ｏｆ－Ｔｙｐｅ－ａｎｄ－Ｏｗｎｅｒ」と呼ばれる３ウェイリレーションシップにおいて関係付けられている（関連エンティティ（associative entity）とも称される）。関連エンティティは、３つの他のエンティティを一緒に同時に結ぶよう作成される。

【０１３４】

１つの一般的な誤認は、マルチウェイリレーションシップをバイナリリレーションシップの連なりと同等視する誤りである。図２４Ａおよび図２４Ｂは、２つの一般的なＥ－Ｒ表記法における３ウェイリレーションシップに対する等価なものを示す。図２４Ａを参照して、モデル２４１０は、標準的なＥ－Ｒ表記法での２つのバイナリリレーションシップを有する、図２３Ａのモデル２３１０に等価なものとして描かれるかまたは別の態様で表わされる。図２４Ｂを参照して、モデル２４２０は、カラスの足表記法での等価なモデルを有する、モデル２３２０に等価なものとして描かれるかまたは別の態様で表わされる。

【０１３５】

これらのモデルの両方は、２つのバイナリリレーションが常に同時に保持することを必

要としないという同じ問題を共有する。たとえば、「ペットＡ」と称するＰＥＴにおけるインスタンスは、「ＰＴ　Ａ」と称するＰＥＴ＿ＴＹＰＥにおけるインスタンスに関係し得るが、「ペットＡ」がさらにＰＥＴ＿ＯＷＮＥＲからのインスタンスに関係しなければならないということは必要とされない。

【０１３６】

しかしながら、ペットは同時に２つのバイナリリレーションシップに参加しなければならないという事実をモデル化することが可能である。図２５は、バイナリリレーションシップの連なりを使用して３ウェイリレーションシップに対する等価なものを示す。この例において、図２４Ｂとは異なり、モデル２５００は、関連エンティティとしてＰＥＴエンティティを表わす。関連エンティティにおける各インスタンスは、例外なくすべての他の接続されたエンティティに関する。図２５は、バイナリリレーションシップの連なりのように思われ得るが、実際には隠れた図２３Ｂにおける３ウェイリレーションシップであり、ＰＥＴエンティティは、図２３Ｂにおいて示される関連エンティティを吸収する。

【０１３７】

１つの局面において、ＰＥＴエンティティが関連エンティティを吸収することができる２つの特殊な場合が存在する。第１に、１つの可能性は、各ＰＥＴインスタンスが１つ以下のリレーションシップインスタンスに参加するということである。第２に、別の可能性は、ＰＥＴが弱いエンティティかどうかである（弱いエンティティの形式定義は、自身の一次キー（primary key）を有さないエンティティである）。ＰＥＴが強いエンティティであると仮定すると、それ自身の一次キーは、ペットを識別するためにのみ使用されなければならない。リレーションシップインスタンスを識別するためにもそれを使用することができない。たとえば、ペットインスタンスが１つより多いリレーションシップインスタンスに参加する場合には、強いＰＥＴエンティティに一次キー違反が存在することになる。他方、ＰＥＴが弱いエンティティである場合、その（ユニークでない）部分キー(partial key)は、３リレーションシップのキー（部分的またはユニークのいずれか）と組み合わされ得る。この場合、ＰＥＴは３リレーションシップを吸収し得る。

【０１３８】

したがって、付加的な仮定（additional assumption）を作成することなく、図２３Ｂは、バイナリリレーションシップの連なりに類似するよう変形され得ない。したがって、図２３Ｂにおいて示される一般的な３リレーションシップに焦点を合わせる。

【０１３９】

図２６は、標準的なＥ－Ｒ表記法を使用して３ウェイリレーションシップを示す。当該例についてのスキーマは、ＰＥＴ，ＰＥＴ＿ＴＹＰＥ，ＰＥＴ＿ＯＷＮＥＲエンティティを含み、これに加えて、それらについての１つ以上の３リレーションシップを含む。いくつかの付加的な情報も提供される。この例において、モデル２６００は、Ｅ－Ｒダイアグラムにおいてカーディナリティ範囲（cardinality range）０．．ｍによって示されるように、３リレーションシップの随意の参加者としてＰＥＴを表わす。他の２つのエンティティは両方ともリレーションシップの完全な参加者である。

【０１４０】

ｐ，ｔ，ｏがそれぞれＰＥＴ、ＰＥＴ＿ＴＹＰＥおよびＰＥＴ＿ＯＷＮＥＲのインスタンスであるとする。「Ｐｅｔ－ｏｆ－Ｔｙｐｅ－ａｎｄ－Ｏｗｎｅｒ」リレーションシップにおける可能性のあるインスタンスは以下のとおりである。

・(p, t, o)

・(<missing>, t, o)

ここで、＜ｍｉｓｓｉｎｇ＞は、エンティティからの値の欠如を表わす。これらの候補タプルが有効なリレーションシップインスタンスであるかどうかは、

PET.type_id = PET_TYPE.id and PET.owner_id = PET_OWNER.id

と規定される３ウェイジョイン条件によって判定される。

【０１４１】

なお、＜ｍｉｓｓｉｎｇ＞という値は、任意の他の値と一致することが可能である。そ

のため、この例において、タプル（＜ｍｉｓｓｉｎｇ＞，ｔ，ｏ）は、
<missing> = PET_TYPE.id and <missing> = PET_OWNER.id
という条件が真と評価されるので、リレーションシップの有効なインスタンスである。
【０１４２】
これらのエンティティについての３つの例示的なテーブルは、以下のステートメントによって作成される。
【０１４３】
【数１】

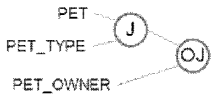
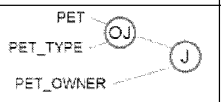
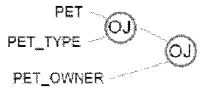
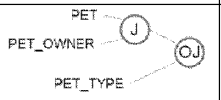
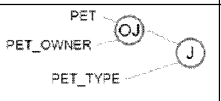
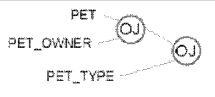
```
create table PET (
 id   number,
 name varchar2(30),
 tid  number,
 oid  number);

create table PET_TYPE (
 id   number,
 name varchar2(30));

create table PET_OWNER (
 id   number,
 name varchar2(30));
```

【０１４４】
図２７は、各テーブルにおけるロウを示す。示されるように、各エンティティは１つのインスタンスのみを有する。１つの局面において、ユーザは、
PET.type_id = PET_TYPE.id and PET.owner_id = PET_OWNER.id
という３ウェイジョイン条件を入力し得る。
【０１４５】
ユーザはさらに、随意のものとして、エンティティＰＥＴをマークし得る。これは、図２６に示されるＥ－Ｒモデルを入力することと等価である。１つの困難は、図２６の意味を最も良くキャプチャするためにどのようにＳＱＬステートメントを生成するかである。さまざまな実施形態において、良好なマルチウェイジョインインプリメンテーションを提供することになるシンタックスに関して判定がなされる。以下の例において、ＡＮＳＩジョインシンタックスが使用される。
【０１４６】
各ＡＮＳＩジョインはペアワイズ（pair-wise）であるので、３つのテーブルを結合するためには、２つのジョインが必要である。さらに、ＰＥＴは随意のエンティティであるので、２つのジョインの少なくとも１つはアウタージョインでなければならない。更に、どの２つのテーブルが最初に結合されるかも考慮するべきファクタである。これらの考慮

をすべて一緒にすると、ＡＮＳＩシンタックスを使用するマルチウェイジョインについて９つの可能なインプリメンテーションに対応する９つの順列が発生する。これらの場合は、データフローチャートを使用して描かれ、それらのＳＱＬステートメントおよび結果とともに以下の表１に示される。

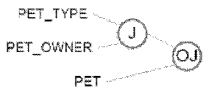
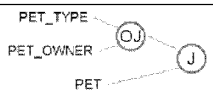
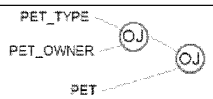
【０１４７】

【表１】

	フローチャート	SQL	結果
1		select p.name "pet", t.name "type", o.name "owner" from PET p join PET_TYPE t on (p.tid = t.id) right outer join PET_OWNER o on (p.oid = o.id)	pet type owner -------- -------- ---------- Jeff (コメント: Type が無効化されています。これは良い結果ではありません。)
2		select p.name "pet", t.name "type", o.name "owner" from PET p right outer join PET_TYPE t on (p.tid = t.id) join PET_OWNER o on (p.oid = o.id)	ロウが選択されていません
3		select p.name "pet", t.name "type", o.name "owner" from PET p right outer join PET_TYPE t on (p.tid = t.id) right outer join PET_OWNER o on (p.oid = o.id)	pet type owner -------- -------- ---------- Jeff (コメント: Type が無効化されています。これは良い結果ではありません。)
4		select p.name "pet", t.name "type", o.name "owner" from PET p join PET_OWNER o on (p.oid = o.id) right outer join PET_TYPE t on (p.tid = t.id)	pet type owner -------- -------- ---------- Cat (コメント: Owner が無効化されています。これは良い結果ではありません。)
5		select p.name "pet", t.name "type", o.name "owner" from PET p right outer join PET_OWNER o on (p.oid = o.id) join PET_TYPE t on (p.tid = t.id)	ロウが選択されていません
6		select p.name "pet", t.name "type", o.name "owner" from PET p right outer join PET_OWNER o on (p.oid = o.id) right outer join PET_TYPE t on (p.tid = t.id)	pet type owner -------- -------- ---------- Cat (コメント: Owner が無効化されています。これは良い結果ではありません。)

【０１４８】

【表２】

7		select p.name "pet", t.name "type", o.name "owner" from PET_TYPE t join PET_OWNER o on (1 = 1) left outer join PET p on (p.tid = t.id and p.oid = o.id)	pet type owner -------- -------- ---------- Cat Jeff (コメント: これは期待した結果です。 なお PET_TYPE と PET_OWNER との間には直接的なジョイン条件は存在しません. デフォルトで真になります。)
8		select p.name "pet", t.name "type", o.name "owner" from PET_TYPE t full outer join PET_OWNER o on (1 = 1) join PET p on (p.tid = t.id and p.oid = o.id)	ロウが選択されていません
9		select p.name "pet", t.name "type", o.name "owner" from PET_TYPE t full outer join PET_OWNER o on (1 = 1) left outer join PET p on (p.tid = t.id and p.oid = o.id)	pet type owner -------- -------- ---------- Cat Jeff (コメント: 生成された SQL はインプリメンテーション#7 におけるものと等価です)

【０１４９】

すべての可能なインプリメンテーションの検討から、インプリメンテーション＃７が３ウェイリレーションシップについての期待と一致するように思われる。したがって一般に、マルチウェイリレーションシップは、バイナリリレーションシップの連なりと等価ではない。しかしながら、さまざまな実施形態において、マルチウェイリレーションシップは、バイナリジョインを使用して実現され得る。したがって、１つの局面では、正しいデータフローインプリメンテーションを生成する際の使用のために、（カジュアルな）人間のユーザにとって理解可能でありつつ正確であるモデルが作成される。

【０１５０】

上で論じたように、等価なデータフローモデルは、「オペレーショナルセマンティックモデル」のカテゴリーにフィットする。システムの意味／意図を明白に記述するオペレーティングセマンティックモデルが作成されている。しかしながら、Ｅ－Ｒモデルを等価に表わすためのものは、本願明細書において論じられるような新たな機会を提供する。

【０１５１】

図２８Ａおよび図２８Ｂは、一実施形態において、Ｅ－Ｒ表記法における３ウェイリレーションシップと、３つのエンティティから生じるデータを有するデータフローとを示す。図２８ＡのＰＥＴの例を使用して、図２８Ｂは、３つのエンティティから生じるデータを有するデータフローを記載する。各エンティティは、タプルのセットを提供する。各タプルはカラム／属性のリストから構成される。すべてのタプルは、以下に規定されるコネクトフェーズ、フィルタフェーズ、およびアウトプットフェーズという３つのステージを経る。

【０１５２】

コネクトフェーズ：すべての入力エンティティのカルテシアン積を実行する。エンティティが随意のエンティティ（以下に規定）である場合、カルテシアン積が行なわれる前に、値＜ｍｉｓｓｉｎｇ＞のすべてのカラムを有する特別なタプルが、エンティティの追加のメンバーとして最初に加えられる。

【０１５３】

フィルタフェーズ：フィルタフェーズにおいて、コネクトフェーズからのすべてのタプルは以下の３つのグループへ分類される。

・グループＦ（リレーションシップ条件を達成しないタプルを含む）

・グループＳ１（たとえば、如何なる＜ｍｉｓｓｉｎｇ＞値も比較することなく、PET.tid = PET_TYPE.id and PET.oid = PET_OWNER.id

というリレーションシップ条件を満たすタプルを含む）
・グループＳ２（リレーションシップ条件を満たすすべての他のタプルを含むが、補足値＜ｍｉｓｓｉｎｇ＞が比較において使用される）。
【０１５４】
直観的に、グループＳ１は、ストレートな成功を成し遂げたロウを含む。グループＳ２は、随意のエンティティからの無視できるｍｉｓｓｉｎｇ値により、ジョイン条件をパスした。
【０１５５】
アウトプットフェーズ：以下のルールを使用して、タプルのセットである最終結果を出力する。
・グループＦからのすべてのタプルが廃棄される。
・グループＳ１からのすべてのタプルが最終結果セットに含まれる。
・グループＳ２からのタプルは、最終結果に対して重要な寄与を有する場合のみ、最終結果セットに含まれる。
【０１５６】
タプルは、結果セットにおけるタプルのうちの１つに一致する場合、最終結果に対して重要な寄与をしないと考えられる。２つのタプルがマッチするかどうかチェックする際に、＜ｍｉｓｓｉｎｇ＞値が任意の他の値とマッチするとみなす。たとえば、以下の２つのタプルはマッチする。
('ABC', 123) vs (<missing>, 123)
直観的に、最終のアウトプットフェーズは、グループＳ１およびＳ２におけるタプルに対して重複排除（deduplication）を実行する。
【０１５７】
図２７における例示的なデータを使用して、コネクトフェーズの結果は、
(pet_100, pet_type_1, pet_owner_10)
(<missing>, pet_type_1, pet_owner_10)
という２つのタプルを含む。
【０１５８】
ここで、ｉｄ＝１００を有するＰＥＴテーブルにおけるロウを表わすためにｐｅｔ＿１００が使用される。なお、ＰＥＴは随意のエンティティであるので、値＜ｍｉｓｓｉｎｇ＞は「有効な」ペットとして扱われる。
【０１５９】
第２のフェーズにおいて、
PET.tid = PET_TYPE.id and PET.oid = PET_OWNER.id
というマルチウェイジョイン条件が評価される。そして、
(<missing>, pet_type_1, pet_owner_10)
というタプルのみが当該条件を満たす。
【０１６０】
最終フェーズは、如何なる重複排除も行なう必要がないので、この例については重要でないことである。
【０１６１】
テーブル１におけるインプリメンテーション＃７が正しい結果を返すことができる理由は、インプリメンテーション＃７が、ジョイン条件を評価し始める前にすべてのテーブルのカルテシアン積を行なうからである。インプリメンテーション＃７は、規定されたオペレーショナルセマンティックモデルに一貫している唯一のインプリメンテーションである。ロウがデータフローにおいてフィルタリングされる前にカルテシアン積演算が良好に完了することを確実にすることにより、マルチウェイリレーションシップに固有の同時性プロパティが、潜在的に破壊的なバイナリジョインから保護される。
【０１６２】
いくつかの実施形態において、ユーザは、コネクトフェーズオペレーションについての

必要性を示す線をエンティティ間に描くことによりモデルを視覚的に作成することが可能であり得る。たとえば、ユーザが単にＰＥＴとＰＥＴ＿ＯＷＮＥＲとの間の接続を描いたが、
PET.tid = PET_TYPE.id and PET.oid = PET_OWNER.id
といったようにリレーションシップ条件を入力したとする。

【０１６３】

上記の３ウェイリレーションシップ条件を見ると、接続は、自動的に判定され、ＰＥＴとＰＥＴ＿ＴＹＰＥとの間で作成され得る。これは、コネクトフェーズオペレーションが、含まれるすべてのエンティティのカルテシアン積を必要とするからである。この導き出されたジョインに関するジョイン条件は、カルテシアン積を達成する場合のみ、１＝１である。

【０１６４】

さらに、ユーザが、ＰＥＴ＿ＯＷＮＥＲからＰＥＴ＿ＴＹＰＥまで付加的な線を描いて、３つのエンティティの間で円を形成したとする。１つの局面において、円を作成した新しい線は、リレーションシップにおけるすべてのエンティティが十分に接続されているので無視され得る。人間のユーザであれば、線は「バイナリリレーションシップ」を意味すると考えるかもしれないが、図２８Ｂのオペレーショナルセマンティックモデルにずっと従い続けることによって、線は、カルテシアン積を使用してエンティティをともに接続することを意味するだけである。

【０１６５】

エンティティが接続された後、ダイアグラムはバイナリジョインのツリーに変換され得、当該ツリーにおいて、ＰＥＴおよびＰＥＴ＿ＴＹＰＥについてのジョインノードは１＝１条件を保持する。また、マルチウェイジョイン条件は最後のジョインノードに対して遅延される。全プロセスにおいて、ジョイン条件は偽られず、すべてのテーブルからのすべてのロウが相互作用する機会を有することを保証するよう最大限に遅延される。

【０１６６】

対照的に、ジョイン条件が（シンタックス上許される）２つの部分へ分割され、２つのジョインノードに２つのサブ条件を割り当てた場合、オペレーショナルセマンティックモデルは違反されることになる。なぜならば、コネクトフェーズが完了する前にフィルタフェーズが開始されるからである。

【０１６７】

したがって、オペレーショナルセマンティックモデルは、詳細でステップバイステップの態様で特定されるので、任意の既存のプログラミング言語を使用してプログラムインプリメンテーションに容易に変換され得る。それを実現するのに単にＳＱＬを使用する必要はない。

【０１６８】

図２９は、さまざまなデータベースモデリング方法およびそれらのセマンティックコンテンツの間でリレーションシップをレイアウトするダイアグラムを示す。この例において、Ｅ－Ｒモデルはまだ、セマンティックモデルからの支援を必要としている。ＣＦＯモデルは、特にマルチウェイリレーションシップについて、Ｅ－Ｒにおいて曖昧性を除去するための１つのそのようなセマンティックモデルである。図２９に示されるように、同じ目的のために、オブジェクト指向モデルが使用され得る。Ｅ－Ｒを正確にＯＯモデルに変換するための多くの特許が存在する。しかし、ＯＯモデルは、データフローモデルと統合する能力を欠いている。データフローモデルは、ソースからターゲットまでのデータのステップバイステップオペレーションを明示的に説明している。ＯＯモデルは、その目的には記述的すぎる。ＣＦＯモデルは、オートマトンに類似しており、データフローモデルとの統合に本質的に好適である。

【０１６９】

結論

図３０は、本発明の実施形態を実施するために使用され得るコンピュータシステム３０

００の簡略ブロック図である。図３０に示されるように、コンピュータシステム３０００は、バスサブシステム３０２０を介して多くの周辺機器と通信するプロセッサ３０１０を含む。これらの周辺機器は、メモリサブシステム３０４０およびファイルストレージサブシステム３０５０を含むストレージサブシステム３０３０と、入力デバイス３０６０と、出力デバイス３０７０と、ネットワークインターフェイスサブシステム３０８０とを含み得る。

【０１７０】

バスサブシステム３０２０は、コンピュータシステム３０００のさまざまなコンポーネントおよびサブシステムを意図されるように互いと通信させるためのメカニズムを提供する。バスサブシステム３０２０は単一のバスとして概略的に示されるが、バスサブシステムの代替的な実施形態は複数のバスを利用してもよい。

【０１７１】

ストレージサブシステム３０３０は、本発明の機能を提供する基本的なプログラミングおよびデータ構造を格納するように構成され得る。本発明の機能を提供するソフトウェア（コードモジュールまたは命令）は、ストレージサブシステム３０３０に格納され得る。これらのソフトウェアモジュールまたは命令は、プロセッサ３０１０によって実行され得る。ストレージサブシステム３０３０は、さらに本発明に従って使用されるデータを格納するためのリポジトリを提供し得る。ストレージサブシステム３０３０は、メモリサブシステム３０４０とファイル／ディスクストレージサブシステム３０５０とを含み得る。

【０１７２】

メモリサブシステム３０４０は、プログラム実行の間に命令およびデータの格納のためのメインランダムアクセスメモリ（ＲＡＭ）３０４２と、固定された命令が格納されるリードオンリメモリ（ＲＯＭ）３０４４とを含む多くのメモリを含み得る。ファイルストレージサブシステム３０５０は、プログラムおよびデータファイルのための持続性（不揮発性）ストレージを提供しており、ハードディスクドライブ、関連するリムーバブル媒体を有するフロッピー（登録商標）ディスクドライブ、コンパクトディスクリードオンリメモリ（ＣＤ－ＲＯＭ）ドライブ、ＤＶＤ、オプティカルドライブ、リムーバブル媒体カートリッジ、および他の同様のストレージ媒体を含み得る。

【０１７３】

入力デバイス３０６０は、キーボードと、マウス、トラックボール、タッチパッドまたはグラフィックスタブレットのようなポインティングデバイスと、スキャナと、バーコードスキャナと、ディスプレイに組み込まれるタッチスクリーンと、音声認識システム、マイクロホンのような音声入力デバイスと、他のタイプの入力デバイスとを含み得る。一般に、「入力デバイス」という用語の使用は、コンピュータシステム３０００に情報を入力するためのすべての可能なタイプのデバイスおよびメカニズムを含むように意図される。

【０１７４】

出力デバイス３０７０は、ディスプレイサブシステム、プリンタ、ファックスマシン、または音声出力デバイスなどのノンビジュアルディスプレイを含み得る。ディスプレイサブシステムは、陰極線管（ＣＲＴ）、液晶ディスプレイ（ＬＣＤ）のようなフラットパネルデバイス、または投射デバイスであり得る。一般に、「出力デバイス」という用語の使用は、コンピュータシステム３０００から情報を出力するためのすべての可能なタイプのデバイスおよびメカニズムを含むように意図される。

【０１７５】

ネットワークインターフェイスサブシステム３０８０は、他のコンピュータシステム、デバイス、および通信ネットワーク３０９０のようなネットワークにインターフェイスを提供する。ネットワークインターフェイスサブシステム３０８０は、コンピュータシステム３０００からデータを受け取るとともに他のシステムにデータを送信するためのインターフェイスとして機能する。通信ネットワーク３０９０のいくつかの例は、プライベートネットワーク、パブリックネットワーク、専用回線、インターネット、イーサネットネットワーク、トークンリングネットワーク、および光ファイバーネットワークなどである。

【０１７６】
コンピュータシステム３０００は、パーソナルコンピュータ、ポータブルコンピュータ、ワークステーション、ネットワークコンピュータ、メインフレーム、キオスクまたは任意の他のデータ処理システムを含むさまざまなタイプのものであり得る。コンピュータおよびネットワークの絶えず変化する性質により、コンピュータシステムの好ましい実施形態を説明する目的のために、図３０に示されるコンピュータシステム３０００の説明は特定の例としてのみ意図される。図３０に示されたシステムよりも多くまたはより少ないコンポーネントを有する多くの構成が可能である。
【０１７７】
図３１は、本発明の実施形態に従ったデータマッピングの生成を促進するためのデータ統合システム３１００の簡略ブロック図である。本発明の原理を実行するために、データ統合システム３１００のブロックは、ハードウェア、ソフトウェアまたはハードウェアおよびソフトウェアの組合せによって実現され得る。当業者であれば、図３１に記載されるブロックは、上述されるように、本発明の原理を実現するために、組み合されてもよく、またはサブブロックへと分離されてもよいということが理解される。したがって、本願明細書における記載は、本願明細書において記載される機能ブロックの任意の可能な組合せ、分離、またはさらなる定義をサポートし得る。
【０１７８】
図３１に示されるように、受取部３１１０、判定部３２２０および生成部３１３０を含むデータ統合システム３１００が示される。随意に、データ統合システム３１００はさらに、導出部３１４０およびエクスポート部３１５０を含み得る。
【０１７９】
一実施形態において、受取部３１１０は、論理設計のコンポーネントとしてエンティティリレーションシップのセットを特定する情報を受け取るように構成される。判定部３１２０は、エンティティリレーションシップのセットに基づいて、等価なデータフローモデルを判定するように構成される。生成部３１３０は、論理フロー設計において等価なデータフローモデルを示す情報を生成するように構成される。
【０１８０】
実施形態の１つの局面において、導出部３１４０は、データソースの属性同士の間のリレーションシップを宣言する情報に基づき、エンティティリレーションシップのセットを表わすデータセットの１つ以上の属性を導き出すように構成される。実施形態の１つの局面において、受取部３１１０はさらに、論理設計を通って流れる情報の形を変更するオペレーションを示す情報を含む論理設計の１つ以上のコンポーネントを特定する情報を受け取るように構成される。
【０１８１】
実施形態の１つの局面において、受取部３１１０はさらに、論理設計を通って流れる情報のフローを制御するが論理設計を通って流れる情報の形を変更しないオペレーションを示す情報を含む論理設計の１つ以上のコンポーネントを特定する情報を受け取るように構成される。実施形態の１つの局面において、受取部３１１０はさらに、ターゲットデータストアに格納されるデータの１つ以上の属性を有するターゲットコンポーネントを示す情報を含む論理設計の１つ以上のコンポーネントを特定する情報を受け取るように構成される。
【０１８２】
実施形態の１つの局面において、生成部３１３０は、下流のコンポーネントに属性のリストをエクスポートするように構成されるエクスポート部３１５０を含む。実施形態の１つの局面において、受取部３１１０はさらに、１つ以上のリレーションシップの導入による論理設計における変更を受け取るように構成され、判定部３１２０はさらに、更新された等価なデータフローモデルを判定するように構成される。
【０１８３】
本発明の具体的な実施形態を記載してきたが、さまざまな修正例、変更例、代替的な構

成、および均等例も本発明の範囲内に含まれる。記載された発明は、ある特定のデータ処理環境内のオペレーションに制限されず、複数のデータ処理環境において自由に作用する。さらに、特定の一連のトランザクションおよびステップを使用して本発明が記載されたが、本発明の範囲は記載された一連のトランザクションおよびステップに限定されるわけではないということは当業者に明らかであるはずである。

【０１８４】

さらに、本発明をハードウェアおよびソフトウェアの特定の組合せを用いて説明したが、ハードウェアおよびソフトウェアの他の組合せも本発明の範囲内であると認識されるべきである。本発明は、ハードウェアのみで、またはソフトウェアのみで、またはその組合せを使用して実現され得る。

【０１８５】

したがって、明細書および図面は、限定的な態様ではなく例示的な態様であるとみなされるべきである。しかしながら、添付の特許請求の範囲に記載されるより広い本発明の精神および範囲から逸脱することがなければ、追加、削減、削除、ならびに、他の修正および変更もなされてもよいということが明らかであろう。

【０１８６】

その教示がこの開示内に示され得る１つ以上の発明のいずれかのさまざまな実施形態がソフトウェア、ファームウェア、ハードウェアまたはその組合せにおけるロジックの形で実現され得る。ロジックは、この開示において示された発明のさまざまな実施形態において開示され得るステップのセットを実行するために、ロジックマシンの中央処理装置（ＣＰＵまたはプロセッサ）を指示するように適合される命令のセットとして、マシンアクセス可能なメモリ、マシン読み取り可能な物品、有形的なコンピュータ読取可能媒体、コンピュータ読取可能記憶媒体、または他のコンピュータ／マシン読取可能媒体に格納され得る。ロジックは、この開示に示される発明のさまざまな実施形態における方法またはプロセスを行なうよう実行される際に、コードモジュールがコンピュータシステムまたは情報処理デバイスのプロセッサにより作動状態になると、ソフトウェアプログラムまたはコンピュータプログラムプロダクトの一部を形成し得る。本願明細書において提供されるこの開示および教示に基づいて、示された発明の１つ以上のさまざまな実施形態の開示されたオペレーションまたは機能のいずれかをソフトウェア、ファームウェア、ハードウェアまたはその組合せで実現するための他の態様、変形例、修正例、代替例および／または方法を当業者は理解するであろう。

【０１８７】

その教示がこの開示に示され得るそれらの発明のいずれか１つの開示された例、実現例およびさまざまな実施形態は、当業者にこの開示の教示を妥当な明瞭さで伝えるために単に例示的である。これらの実現例および実施形態は例示的な図または特定の図を参照して記載され得る際に、記載される方法および／または特定の構造のさまざまな修正例または適合例は当業者に明らかになり得る。本願明細書において発見されるこの開示およびこれらの教示に依存し、かつ、当該教示によって技術を進歩させたすべてのそのような修正例、適合例または変形例は、その教示がこの開示内に示され得る１つ以上の発明の範囲内に存在すると考えられるべきである。したがって、開示内に示された発明が具体的に示される実施形態にまったく限定されないということが理解されるので、この記載および図は限定的な意味で考えられるべきでない。

【０１８８】

したがって、上記の記載および如何なる添付の図面、説明および図は、例示的であるが限定的ではないように意図される。したがって、この開示に示される如何なる発明の範囲も、上記の記載および図に示されるそれらの実施形態を単純に参照してではなく、それらの完全な範囲または均等物とともに係属中の請求項を参照して決定されるべきである。

【図１】

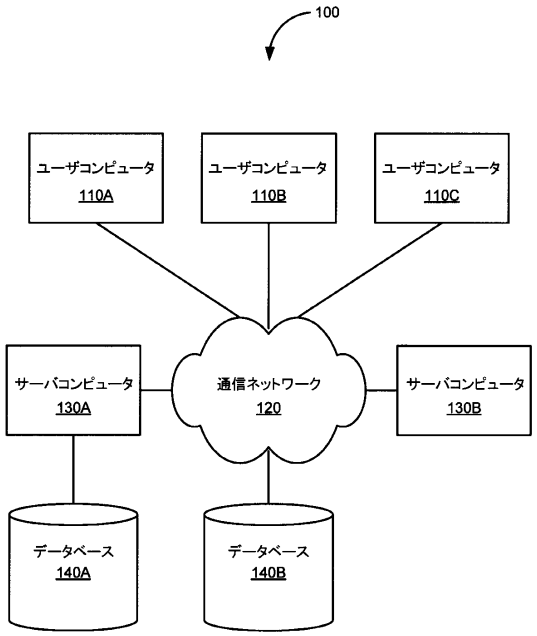


FIG. 1

【図２】

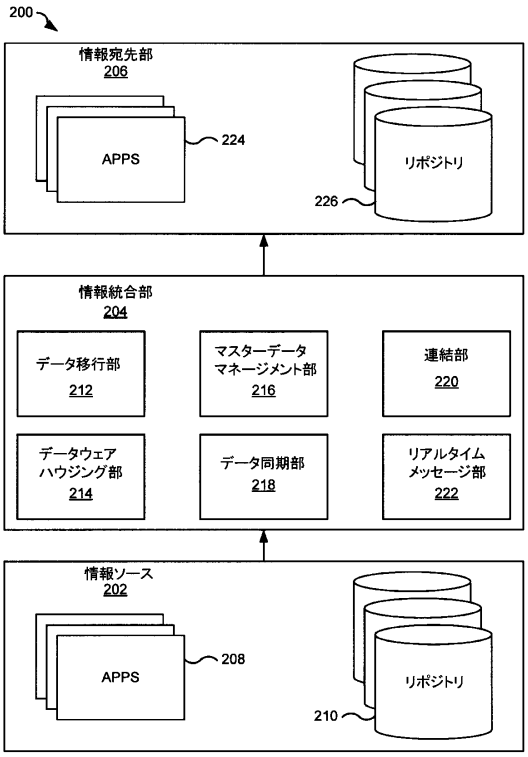


FIG. 2

【図３】

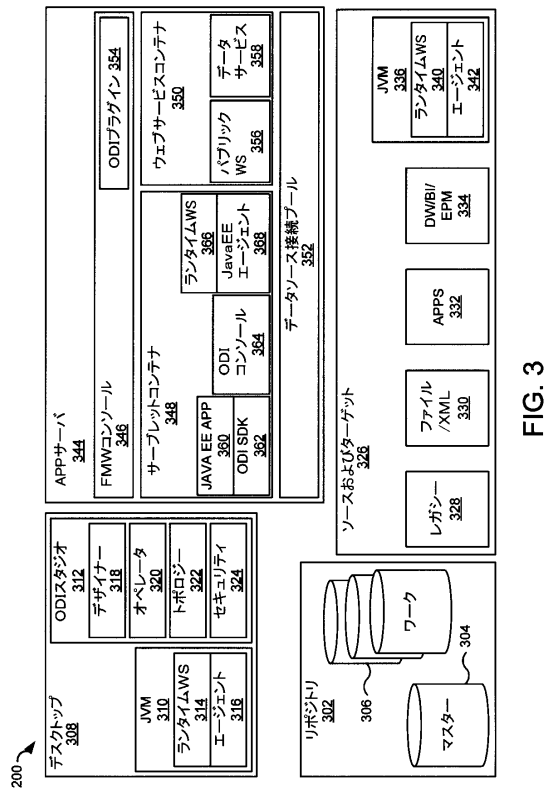


FIG. 3

【図４】

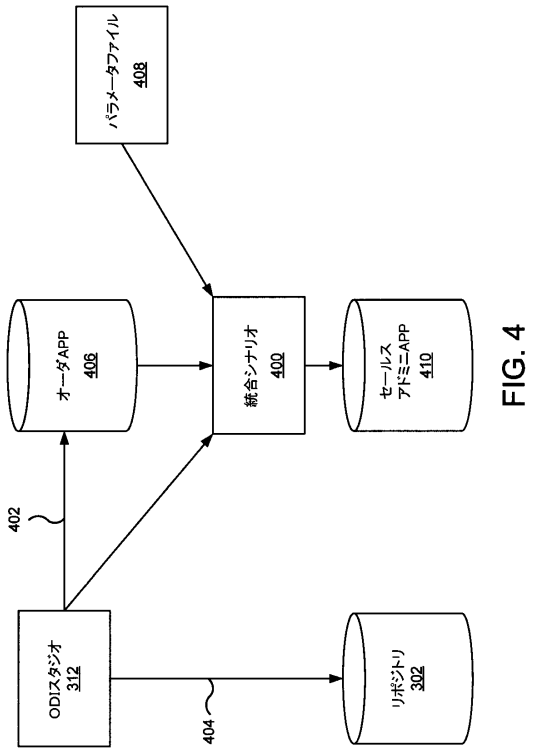


FIG. 4

【図５Ａ】

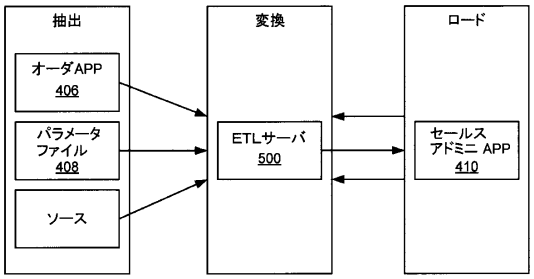


FIG. 5A

【図６Ａ】

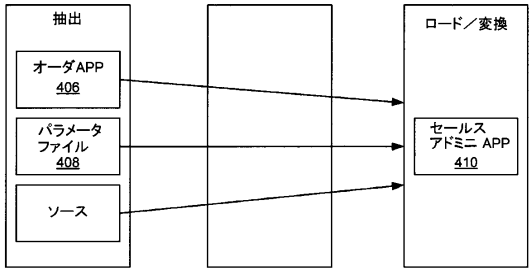


FIG. 6A

【図５Ｂ】

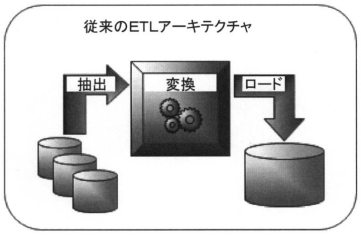


FIG. 5B

【図６Ｂ】

FIG. 6B

【図７】

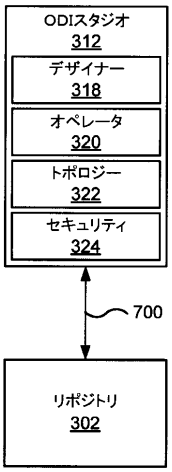


FIG. 7

【図８】

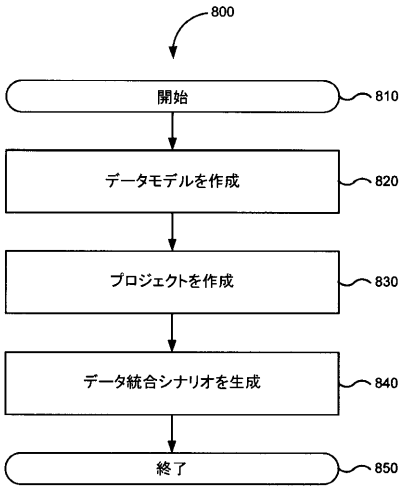


FIG. 8

【図９】

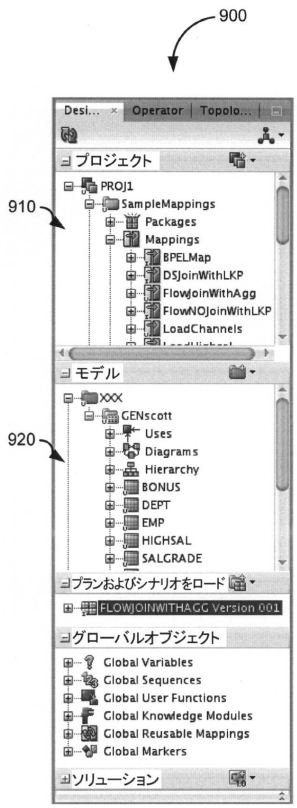


FIG. 9

【図１０】

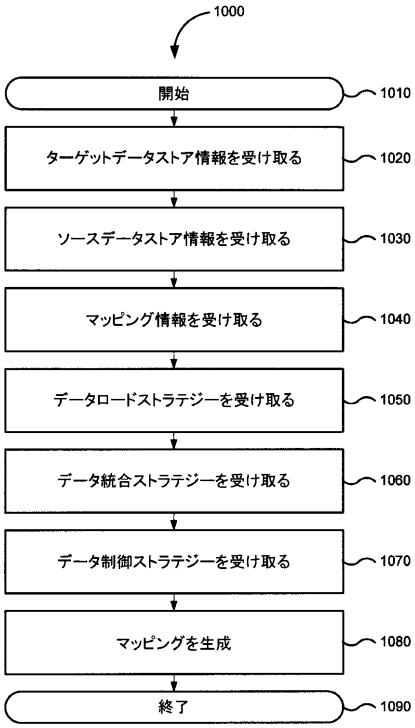


FIG. 10

【図１１】

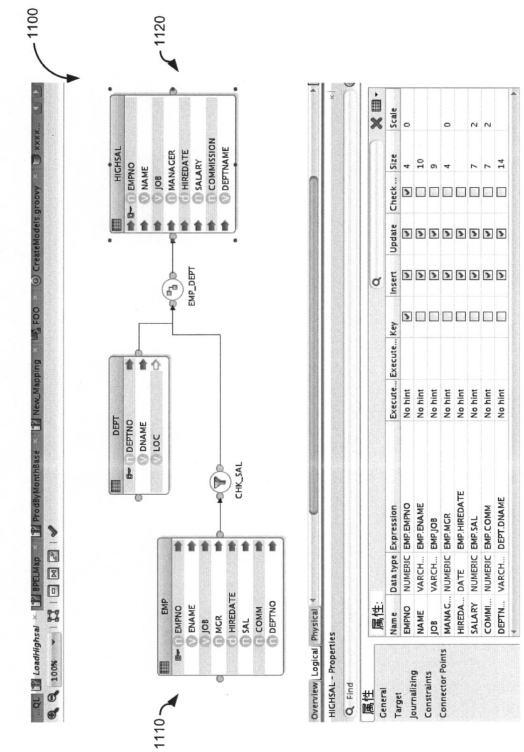


FIG. 11

【図１２】

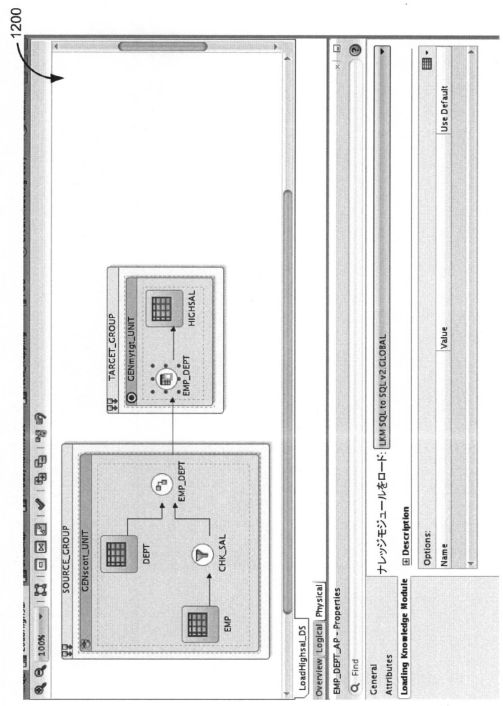


FIG. 12

【図１３】

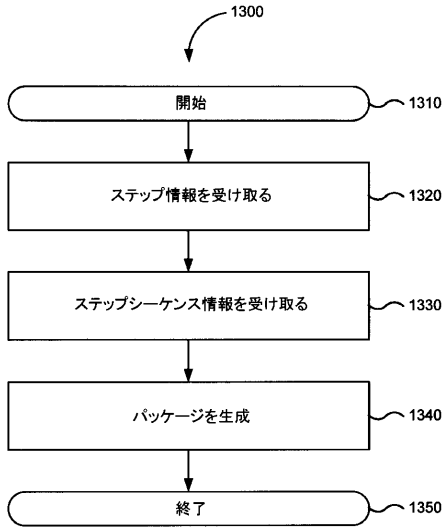


FIG. 13

【図１４】

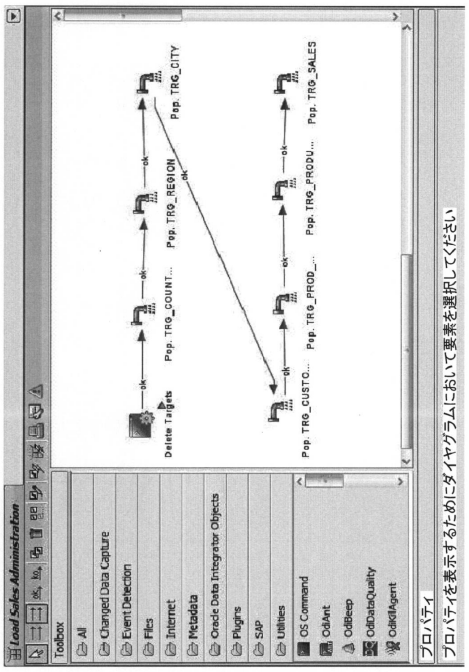


FIG. 14

【図１５】

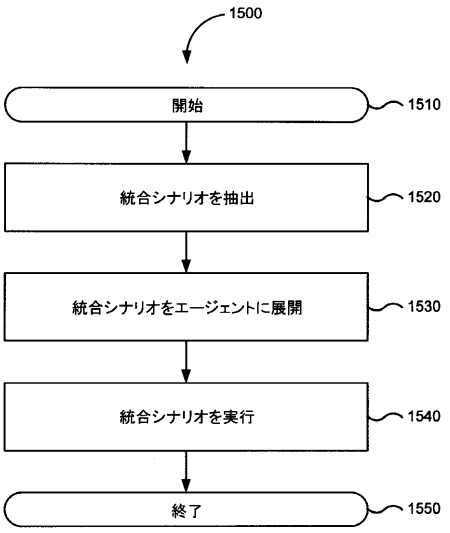


FIG. 15

【図１６】

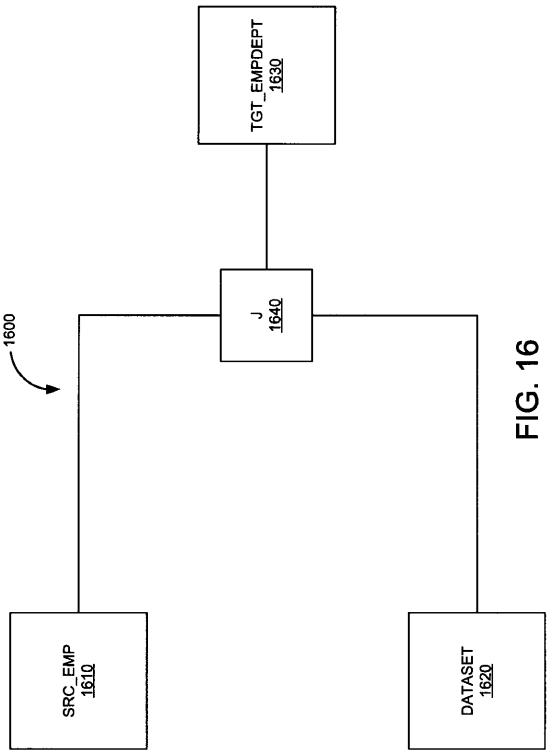


FIG. 16

【図１７】

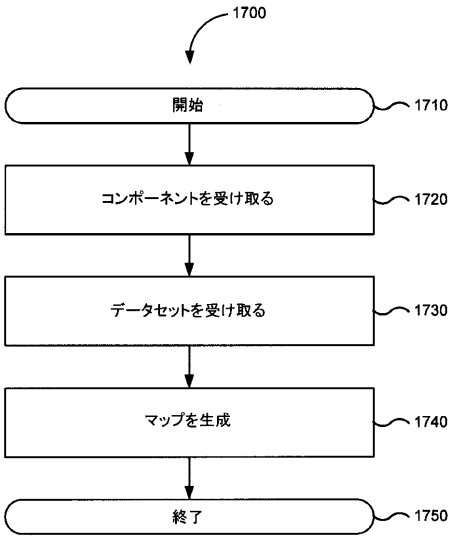


FIG. 17

【図１８】

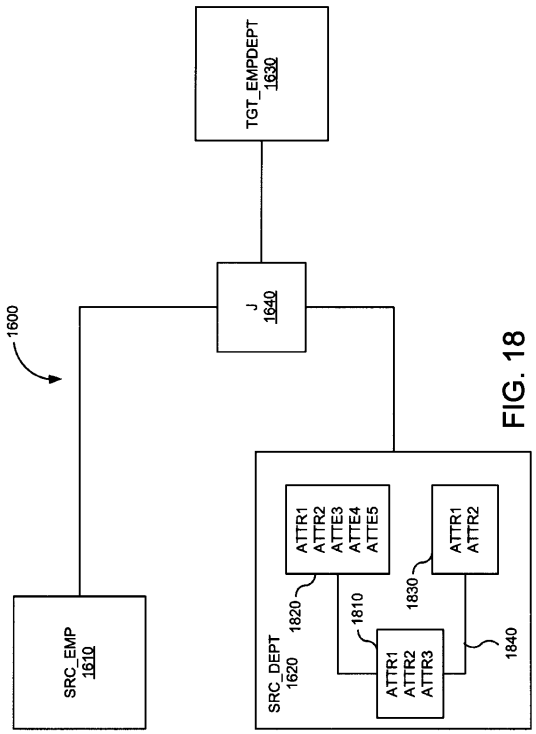


FIG. 18

【図１９Ａ】

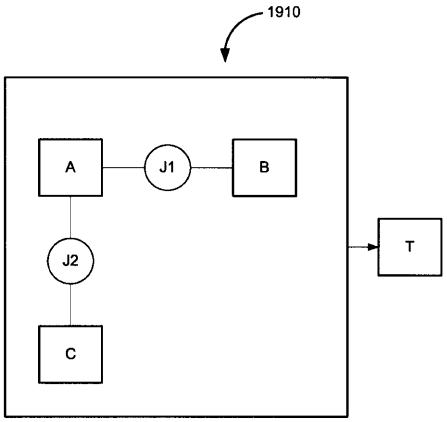


FIG. 19A

【図１９Ｂ】

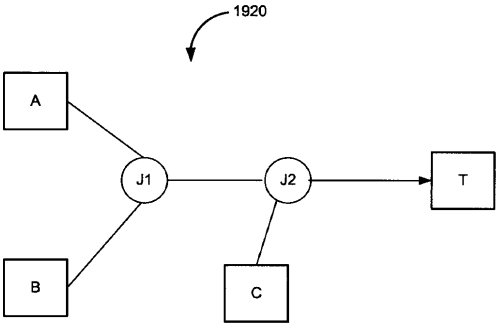


FIG. 19B

【図２０】

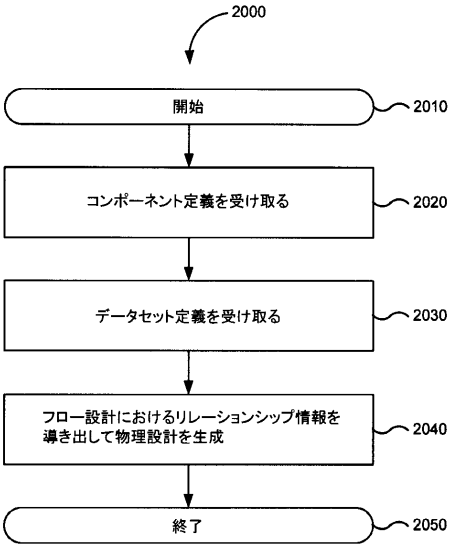


FIG. 20

【図２１】

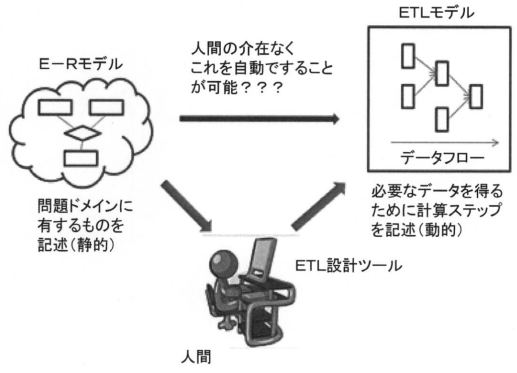


FIG. 21

【図２２】

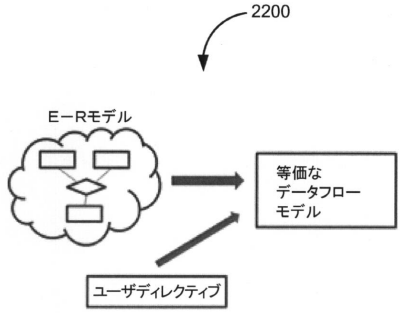
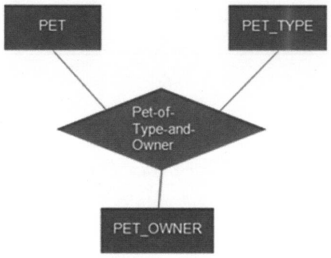


FIG. 22

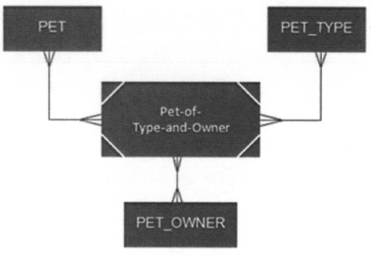
【図２３Ａ】



標準的なE−R表記法

FIG. 23A

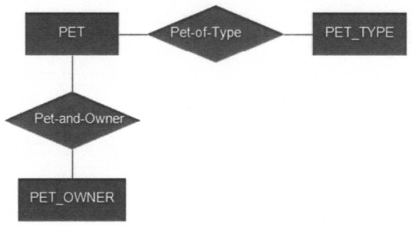
【図２３Ｂ】



カラスの足表記法

FIG. 23B

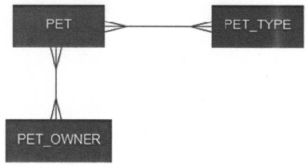
【図２４Ａ】



標準的なＥ－Ｒ表記法

FIG. 24A

【図２４Ｂ】



カラスの足表記法

FIG. 24B

【図２５】

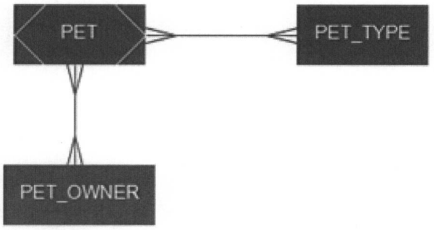


FIG. 25

【図２６】

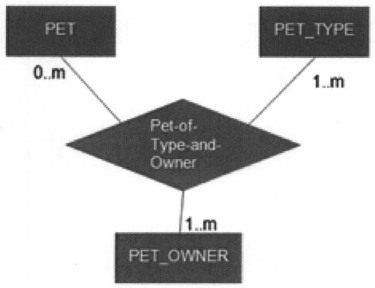


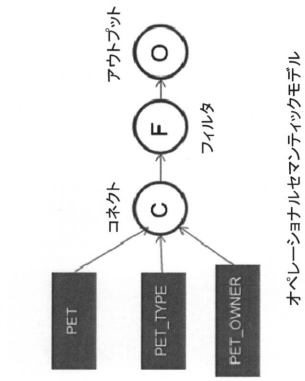
FIG. 26

【図２７】

```
SQL> select * from PET;

        ID NAME                                TID        OID
---------- ---------- ---------- ----------
       100 Stray Dog                          10

SQL> select * from PET_TYPE;

        ID NAME
---------- ----------
         1 Cat

SQL> select * from PET_OWNER;

        ID NAME
---------- ----------
        10 Jeff
```

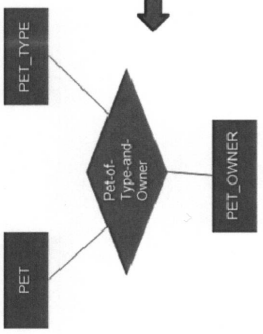
FIG. 27

【図２８】



オペレーショナルセマンティックモデル

FIG. 28B



エンティティリレーションシップ

FIG. 28A

【図２９】

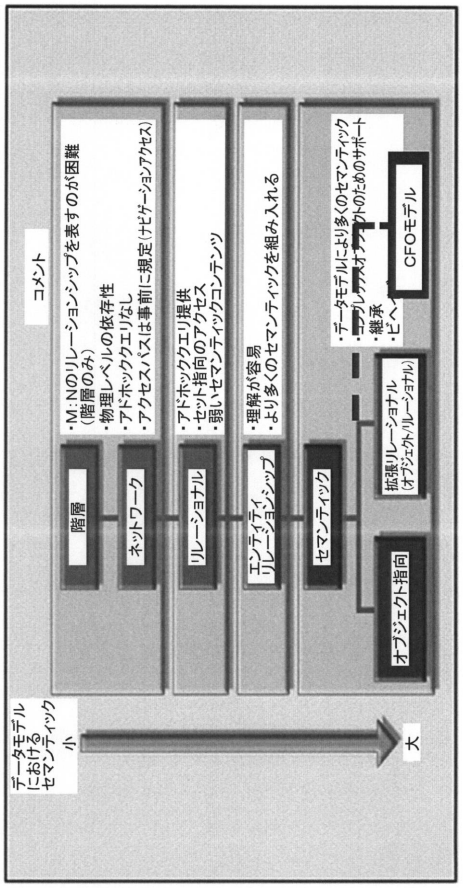


FIG. 29

【図３０】

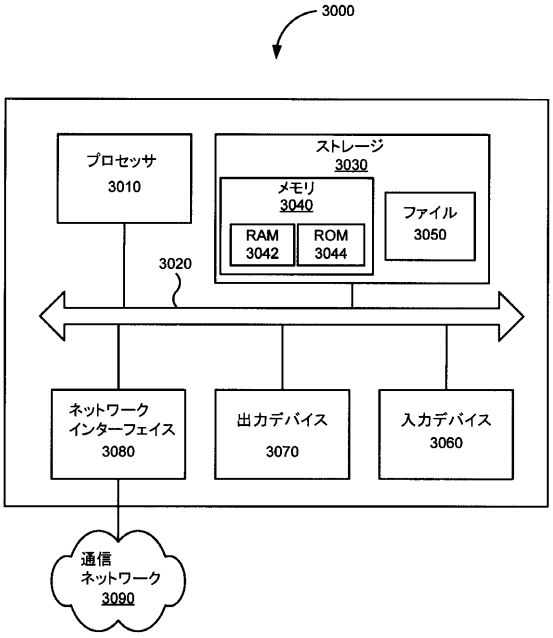


FIG. 30

【図３１】

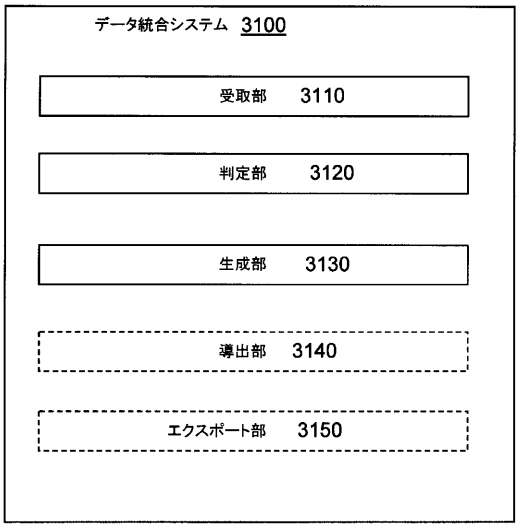


FIG. 31

フロントページの続き

(31)優先権主張番号　14/044,429
(32)優先日　平成25年10月2日(2013.10.2)
(33)優先権主張国　米国(US)

(72)発明者　ラウ，クウォクーハン・（トーマス）
アメリカ合衆国、９４０２２　カリフォルニア州、ロス・アルトス、メルセデス・アベニュ、１０７０、ナンバー・２５
(72)発明者　ゴン，ユウ・（ジェフ）
アメリカ合衆国、９４０６５　カリフォルニア州、ハーフ・ムーン・ベイ、ミュアフィールド・ロード、４

審査官　山崎　誠也

(56)参考文献　特開２００６－０３１６８７（ＪＰ，Ａ）
特開２００６－１７２０３６（ＪＰ，Ａ）
米国特許出願公開第２０１３／０１１１３７５（ＵＳ，Ａ１）
データベース技術を最大限に活用できるデータ連係基盤　～Oracle Data Integratorのご紹介～，日本オラクル株式会社，２０１０年１２月３１日，ｐ．１－５４
データ連携がこんなに簡単に!?　Oracle Data Integratorのご紹介，日本オラクル株式会社，２００９年１２月３１日，ｐ．１－４４
Laura Hofman Miquel, Getting Started with Oracle Data Integrator, Oracle Fusion Middleware, ORACLE, ２０１１年　４月３０日, p.1-1～10-4
実演!社内システムにおけるデータ連携方法ご紹介，日本オラクル株式会社，２０１１年　４月６日，ｐ．１－２９

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｑ　１０／００－９９／００
Ｇ０６Ｆ　１２／００