(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2006-244499
(P2006-244499A)
(43)公開日　平成18年9月14日(2006.9.14)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 12/00	(2006.01)	G06F	12/00	513Z	5B065
G06F 3/06	(2006.01)	G06F	3/06	301F	5B082

審査請求 未請求　請求項の数 20　ＯＬ　（全 35 頁）

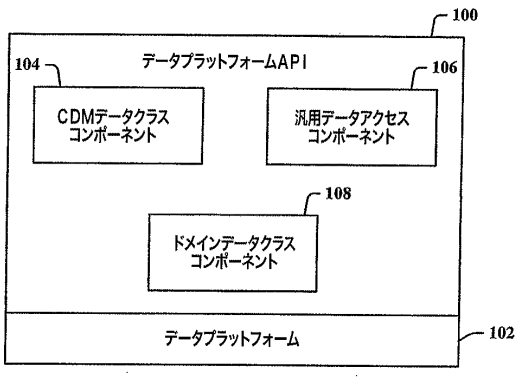
(21)出願番号　特願2006-53435 (P2006-53435)
(22)出願日　平成18年2月28日 (2006.2.28)
(31)優先権主張番号　60/657,522
(32)優先日　平成17年2月28日 (2005.2.28)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　11/195,320
(32)優先日　平成17年8月2日 (2005.8.2)
(33)優先権主張国　米国 (US)

(71)出願人　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　アーサー　ティー．ホイッテン
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内

最終頁に続く

(54)【発明の名称】共通データプラットフォームのためのストレージＡＰＩ

(57)【要約】
【課題】データプラットフォーム用のアプリケーションプログラムインターフェース（ＡＰＩ）を提供する。
【解決手段】ＡＰＩはデータストアに関連付けられたデータプラットフォームのストア、セッション、トランザクション、およびクエリサービスの少なくとも１つを提示するジェネリックデータアクセスコンポーネントを含む。ＡＰＩのデータクラスコンポーネントはデータプラットフォームのデータモデルのタイプおよび関係を提示する正規化アプリケーション非依存クラスを提供する。ＡＰＩはデータプラットフォームのドメイン固有のプロパティ、挙動を提示するアプリケーション固有＆フレームワーク固有クラスのドメインデータクラスコンポーネントを含む。データプラットフォームは異種の複数のアプリケーションフレームワークによってアクセス可能な異なるフレームワークの対応するアプリケーションがデータストアにアクセスできるようにするデータサービスを提供する。
【選択図】図１



【特許請求の範囲】
【請求項１】
データプラットフォーム用のアプリケーションプログラムインターフェース（ＡＰＩ）であって、
データストアに関連付けられている前記データプラットフォームのストア、セッション、トランザクション、およびクエリサービスのうちの少なくとも１つを提示するジェネリックデータアクセスコンポーネントと、
前記データプラットフォームのデータモデルの型および関係を提示する正規化アプリケーション非依存クラスのデータクラスコンポーネントと、
前記データプラットフォームのドメイン固有プロパティおよび挙動を提示するアプリケーション固有およびフレームワーク固有のクラスのドメインデータクラスコンポーネントと
を含むことを特徴とするアプリケーションプログラムインターフェース。
【請求項２】
前記データプラットフォームは、本質的に異なる複数のアプリケーションフレームワークによってアクセス可能な、該異なるフレームワークの対応するアプリケーションが前記データストアにアクセスできるようにするデータサービスを提供するために、前記データストアにインターフェースで接続する共通データプラットフォームであることを特徴とする請求項１に記載のインターフェース。
【請求項３】
前記ドメインデータクラスコンポーネントは、他のクラスが動作するストアを定義するドメインクラスを含むことを特徴とする請求項１に記載のインターフェース。
【請求項４】
前記データクラスコンポーネントは、セッションについてのコンテキストを提供するコンテキストクラスを含むことを特徴とする請求項１に記載のインターフェース。
【請求項５】
前記コンテキストクラスは、現在のコンテキスト内のオブジェクトへの変更をリフレッシュまたは保存するメソッドによって、識別管理、変更の追跡、および同時並行競合処理の範囲を定義することを特徴とする請求項４に記載のインターフェース。
【請求項６】
前記データクラスコンポーネントは、前記データストアに対して結合可能なオブジェクトベースのクエリを構築するために使用されるサーチャークラスを含むことを特徴とする請求項１に記載のインターフェース。
【請求項７】
検索結果集合にわたるビューを提供するビュークラスをさらに含むことを特徴とする請求項１に記載のインターフェース。
【請求項８】
スキーマのテーブルにアクセスを提供するスキーマクラスをさらに含むことを特徴とする請求項１に記載のインターフェース。
【請求項９】
前記スキーマクラスは、対象のスキーマに基づいて型付けされていないスキーマクラスから導出されている強く型付けされたスキーマクラスであることを特徴とする請求項８に記載のインターフェース。
【請求項１０】
請求項１に記載の前記インターフェースを遂行するためのコンピュータ実行可能命令を格納することを特徴とするコンピュータ可読媒体。
【請求項１１】
データプラットフォームを提示するコンピュータ実施方法であって、
データストアに関連付けられている前記データプラットフォームのストア、セッション、トランザクション、およびクエリサービスのうちの少なくとも１つを提示するステップ

と、
前記データプラットフォームのデータモデルの型および関係を提示するステップと、
前記データプラットフォームのドメイン固有のプロパティおよび挙動を提示するステップと
を含むことを特徴とするコンピュータ実施方法。
【請求項１２】
サーバ情報、認証情報、およびマッピング情報を含む前記ストアのストア情報をカプセル化するステップをさらに含むことを特徴とする請求項１１に記載の方法。
【請求項１３】
ストレージサーチャーを介してストレージビューを構築する動作をさらに含むことを特徴とする請求項１１に記載の方法。
【請求項１４】
ストレージサーチャーを介してストレージドメインのクエリを行う動作をさらに含むことを特徴とする請求項１１に記載の方法。
【請求項１５】
クライアントと１つまたは複数の前記ストアとの間の接続をカプセル化するクラスを提供する動作をさらに含むことを特徴とする請求項１１に記載の方法。
【請求項１６】
ストアのうちの少なくとも１つを提示する前記動作は、ジェネリックデータアクセスコンポーネントを介することを特徴とする請求項１１に記載の方法。
【請求項１７】
型および関係を提示する前記動作は、正規化アプリケーション非依存クラスを介することを特徴とする請求項１１に記載の方法。
【請求項１８】
ドメイン固有のプロパティを提示する前記動作は、アプリケーション固有およびフレームワーク固有のクラスのドメインデータクラスコンポーネントを介することを特徴とする請求項１１に記載の方法。
【請求項１９】
データプラットフォームＡＰＩを円滑にするシステムであって、
データストアに関連付けられている前記データプラットフォームのストア、セッション、トランザクション、およびクエリサービスのうちの少なくとも１つを提示する手段と、
前記データプラットフォームのデータモデルの型および関係を提示する手段と、
前記データプラットフォームのドメイン固有のプロパティおよび挙動を提示する手段と、
前記ストアに対して結合可能なオブジェクトベースのクエリを構築する手段と、
前記クエリの１組の検索結果にわたるビューを提供する手段と
を含むことを特徴とするシステム。
【請求項２０】
オブジェクトの強く型付けされた集まりを表す手段をさらに含むことを特徴とする請求項１９に記載のシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、共通データプラットフォームのストレージアプリケーションプログラムインターフェース（ＡＰＩ）に関する。
【背景技術】
【０００２】
製品をブラウズ(ネット閲覧)し、注文を発生するために利用される基幹業務（ＬＯＢ）アプリケーションフレームワークであろうと、人と人との会合のスケジュール管理に使用される個人情報管理（ＰＩＭ）エンドユーザアプリケーションであろうと、ほとんどすべ

てのアプリケーションにおいて、データは重要な価値あるものになってきている。アプリケーションは、アプリケーションデータに対するデータアクセス／操作とデータ管理処理の両方を実行する。一般のアプリケーション操作は、データの集まりに対してクエリ（問合せ）を行い、その結果集合をフェッチ（解読）し、データの状態を修正する一部のアプリケーションロジックを実行し、最後にそのデータを記憶媒体に残す。

【０００３】

従来、クライアント／サーバアプリケーションは、クエリアクションおよび保持アクションを、データ階層に配置されているデータベース管理システム（ＤＢＭＳ）に委ねていた。データ処理を中心とするロジックがある場合、それは、データベースシステム内のストアードプロシージャ（１回の処理で複数の手続を実行可能にするためのプログラムコード）としてコード化される。データベースシステムは、テーブルや行という形でデータに対する処理を行い、アプリケーションは、アプリケーション階層において、プログラミング言語オブジェクト（ＣｌａｓｓおよびＳｔｒｕｃｔなど）の見地からデータに対する処理を行っていた。アプリケーション階層およびデータ階層におけるデータ操作サービス（および機構）の不一致は、クライアント／サーバシステムにおいて許容可能のものであった。しかし、Ｗｅｂ技術（およびサービス指向アーキテクチャ）が出現し、アプリケーションサーバがより広範に受け入れられるようになるにつれて、アプリケーションは多階層（ｍｕｌｔｉ－ｔｉｅｒ）になってきており、さらに重要なことだが、現在では各階層にデータが存在している。

【０００４】

このような階層状のアプリケーションアーキテクチャでは、データは、複数の階層で操作される。さらに、ハードウェアのアドレス可能度が進歩し、メモリが大きくなるにつれて、より多くのデータがメモリに存在するようになってきた。また、アプリケーションは、例えばオブジェクト、ファイル、およびＸＭＬ（拡張可能なマーク付け言語）データなど、本質的に異なるタイプのデータも扱っている。

【０００５】

このようなハードウェアおよびソフトウェア環境において、プログラミング環境によく統合されたリッチなデータアクセスおよび操作サービスの必要性が高まりつつある。上記の問題に当るために取り入れられた従来の実装の１つは、データプラットフォームである。データプラットフォームは、アプリケーションプログラミング環境によく統合された、データにアプリケーションがアクセスし、操作し、管理するためのサービス（機構）の集まりを提供する。しかし、このような従来のアーキテクチャは、多くの点で不十分である。このようなデータプラットフォームのキーポイントなる要求には、複雑なオブジェクトモデリング、上質のリレーションシップ（関係）、論理的データ抽象化と物理的データ抽象化との分離、クエリの優れたデータモデルの概念（ｑｕｅｒｙ　ｒｉｃｈ　ｄａｔａ　ｍｏｄｅｌ　ｃｏｎｃｅｐｔｓ）、アクティブな通知（ａｃｔｉｖｅ　ｎｏｔｉｆｉｃａｔｉｏｎｓ）、中間階層インフラ(基盤)とのよりよい一体化などがある。

【発明の開示】

【発明が解決しようとする課題】

【０００６】

したがって、改善されたデータプラットフォームに関して技術的に要求にまだ十分に応じられていない。

【課題を解決するための手段】

【０００７】

以下は、開示した本発明のいくつかの態様の基本認識を提供するために、簡略化した要約を示している。この要約は、本発明の広範囲な概要を示すものではなく、本発明の鍵となる／重要な要素を特定するためのもの、または本発明の範囲を画定するためのものでもない。この要約は、後述するより詳細な説明の前置きとして、いくつかの概念を簡略化した形式で提示することを唯一の目的としている。

【０００８】

本明細書に開示され、請求項に記載された本発明は、その一態様において、データプラットフォームのアプリケーションプログラムインターフェース（ＡＰＩ）を備える。このＡＰＩは、データストアに関連付けられているデータプラットフォームのストア、セッション、トランザクション、およびクエリサービスのうちの少なくとも１つを提示するジェネリック（ｇｅｎｅｒｉｃ:包括的）データアクセスコンポーネント（成分、構成要素）を含む。ＡＰＩのデータクラスコンポーネントは、データプラットフォームのデータモデルのタイプおよび関係を提示する正規化（ｃａｎｏｎｉｃａｌ）、アプリケーション非依存クラスを提供する。ＡＰＩは、データプラットフォームのドメイン固有のプロパティ（属性）、および挙動を提示するアプリケーション固有の、およびフレームワーク固有のクラスのドメインデータのクラスコンポーネントを含む。データプラットフォームは、さまざまな複数のアプリケーションフレームワークによってアクセス可能な、さまざまなフレームワークの対応するアプリケーションがデータストアにアクセスできるようにするデータサービスを提供するために、データストアに対してインターフェースで接続する共通データプラットフォームであるとすることができる。

【０００９】

別の態様では、ＡＰＩは、５つのコアクラスを含む。ＴａｂｌｅＳｅｔクラスは、データモデルスキーマから生成することができ、スキーマ内で定義されたテーブルへの強く型付けされたアクセス（ｓｔｒｏｎｇｌｙ ｔｙｐｅｄ ａｃｃｅｓｓ）を提供する。ＳｔｏｒａｇｅＤｏｍａｉｎクラスは、残りのクラスが動作するストアを定義する。ＳｔｏｒａｇｅＣｏｎｔｅｘｔクラスは、セッションのコンテキストを提供する。ＳｔｏｒａｇｅＣｏｎｔｅｘｔクラスは、現在のコンテキスト内のオブジェクトへの変更をリフレッシュまたは保存するメソッドによって、識別管理、変更の追跡、および同時並行競合処理（ｃｏｎｃｕｒｒｅｎｃｙ ｃｏｎｆｌｉｃｔ ｈａｎｄｌｉｎｇ）の範囲を定義する。ＳｔｏｒａｇｅＳｅａｒｃｈｅｒクラスは、データストアに対する構成可能なオブジェクトベースのクエリを構築するために使用される。ＳｔｏｒａｇｅＶｉｅｗクラスは、１組の結果にわたるリッチなアプリケーションビューを提供する。ＳｔｏｒａｇｅＶｉｅｗクラスは、フィルタリング、ソート、スクロール、グループ分け、セクション分け、セクションの展開／折りたたみなどの操作をサポートする。

【００１０】

上記および関連の目的を達成するために、開示した本発明のいくつかの態様の例を、以下の説明および添付の図面との関連で本明細書に記載している。しかし、これらの態様は、本明細書に開示した原理を使用し得る様々な方法のほんの一部を示しているにすぎず、本発明にはこのようなすべての態様およびその均等物が含まれるものとする。他の利点および新規の特徴は、以下の詳細な説明を図面と併せ読めば明らかになる。

【発明を実施するための最良の形態】

【００１１】

次に、図面を参照して本発明を説明する。全体を通して、同等の参照番号を使用して同等の要素を指す。以下の説明では、説明上、本発明を完全に理解できるようにするために様々な特定の詳細を記載している。しかし、このような特定の詳細説明なしに、本発明を実施できることは明らかである。他の例では、本発明を説明しやすくするために、よく知られている構造および装置をブロック図の形態で示している。

【００１２】

本出願で使用する場合、「コンポーネント（成分、構成要素）」および「システム」という用語は、ハードウェア、ハードウェアとソフトウェアの組合せ、ソフトウェアそのもの、または実行中のソフトウェアのいずれかのコンピュータ関連のエンティティを指すものとする。例えば、コンポーネントは、それだけには限定されないが、プロセッサ上で稼働するプロセス、プロセッサ、ハードディスクドライブ、（光および／または磁気記憶媒体の）複数の記憶ドライブ、オブジェクト、実行可能ファイル、実行スレッド（一連のメッセージ群）、プログラム、および／またはコンピュータとすることができる。一例として、サーバ上で稼働するアプリケーションおよびサーバはいずれもコンポーネントとする

ことができる。１つまたは複数のコンポーネントがプロセスおよび／または実行スレッド内に存在する可能性があり、１つのコンポーネントを１つのコンピュータ上に配置する、および／または２つ以上のコンピュータの間に分散することができる。

【００１３】

ユーザに対して情報を展示するいくつかの方法が、スクリーンショット(ソフトの画面例)として、いくつかの図に関連して図示され、記述されているが、他の様々な代替を使用できることを当業者であれば理解されよう。「画面（ｓｃｒｅｅｎ）」、「Ｗｅｂページ」、および「ページ」という用語は、本明細書では一般に交換可能に区別なく使用される。ページまたは画面は、表示の説明として、グラフィカルユーザインターフェースとして、または（例えばパーソナルコンピュータ、ＰＤＡ(携帯情報端末）、携帯電話、または他の適応の装置のいずれであろうと）画面上に情報を表す他の方法によって格納され、かつ／または送信され、ページに表示されるレイアウトおよび情報または内容は、メモリ、データベース、または他の記憶設備に格納される。

【００１４】

新規な共通データプラットフォーム（ＣＤＰ）は、オブジェクト、およびそれらがどのように関連しているかを表す共通データモデル（ＣＤＭ）、固定記憶域、およびこのようなオブジェクトのインメモリレプリケーション（ｉｎ－ｍｅｍｏｒｙ ｒｅｐｌｉｃａｔｉｏｎ）を機能するサービスから成る。ＣＤＰは、持続性のデータをアプリケーションオブジェクトとして処理する革新的なプラットフォームを提供する。ＣＤＰは、プラットフォームの一部として定義される、基礎をなすデータモデルおよびサービスに合うように調整される新規なアプリケーションプログラミングインターフェース（ＡＰＩ）を含む。ＣＤＰの機能性は、１組のクラスを介して提示される。パブリックメンバ（メソッドおよびプロパティなど）を含むこのようなクラスの定義は、ＣＤＰ内のオブジェクトを処理するＡＰＩを含む。

【００１５】

最初に図面を参照すると、図１は、革新的な一態様によるデータプラットフォーム１０２（ＣＤＰなど）のストレージＡＰＩ１００を示している。ＡＰＩ１００は、クラス、インターフェース、および静的（スタティック）ヘルパー機能の形のデータプラットフォーム（ＣＤＰなど）を使用してアプリケーションのプログラミングインターフェースを提供する。データベースプログラミング言語の集成（Ｃ＃シーケンス演算子など）もこのＡＰ層の一部である。それをサポートして、ＡＰＩ１００は、エンティティ、リレーションシップ、拡張など、ＣＤＭ概念を提示する１組の正規化アプリケーション非依存クラスであるＣＤＭデータクラスコンポーネント１０４を含む。ジェネリックデータアクセスコンポーネント１０６は、ストア、セッション、トランザクション（ＳｔｏｒａｇｅＣｏｎｔｅｘｔなど）、クエリサービス（ＳｔｏｒａｇｅＳｅａｒｃｈｅｒなど）、およびＣＲＵＤサービス（ＳａｖｅＣｈａｎｇｅｓなど）を提示するＡＰＩ１００の一部として提供される。ＣＲＵＤ（作成、取り出し、更新、および削除）サービスは、データに適用される基本的なプロセス（処理）である。ＡＰＩ１００は、ＣＤＭに従うが、ドメイン固有のプロパティおよび挙動を提示するＣｏｎｔａｃｔ、Ｍｅｓｓａｇｅ、ＰｕｒｃｈａｓｅＯｒｄｅｒｓなどのアプリケーション／フレームワーク固有のクラスであるドメインデータクラスコンポーネント１０８も含む。

【００１６】

図２は、開示された一態様によるストレージＡＰＩを提供する方法を示している。説明を簡潔にするために、例えば、フローチャートまたはフロー図の形で本明細書に示した１つまたは複数の方法は、一連の動作として示され、表されているが、一部の動作は、本発明に従って、本明細書に示され、表されたものとは異なる順序で、および／または他の動作と同時に行うことができるので、本発明は、動作の順序によって限定されるものではないことを理解されたい。例えば、手順を、代わりに相互に関係のある一連の状態またはイベントとして、例えば状態図などで表すことができることを当業者であれば理解されよう。さらに、本発明による方法を実施するのに、ここで示したすべての動作が必要であると

は限らない。２００で、ストレージＡＰＩが受信される。２０２で、ＡＰＩは、データのクエリを行うクラスを定義する。２０４で、ＡＰＩは、データを取り出すクラスを定義する。２０６で、ＡＰＩは、データをナビゲートするクラスを定義する。２０８で、ＡＰＩは、データを修正（モディファイ）するクラスを定義する。２１０で、ＡＰＩは、データを保持するクラスを定義する。

【００１７】

図３は、ジェネリックデータアクセスコンポーネント１０６のより詳細な図を示している。ＡＰＩ３００は、使用しやすい、拡張可能な、強力な、結合可能（ｃｏｍｐｏｓａｂｌｅ）なクラスおよびメソッドへの機能性のファクタリング（ｆａｃｔｏｒｉｎｇ：因数化）を定義する。ストレージＡＰＩ３００は、共通のデータモデルに準拠して、データのクエリ、取り出し、ナビゲート、修正、およびデータに対する変更の保持を行うクラスを定義する。ストレージＡＰＩ３００は、このクエリ、ナビゲーション、および保持の機能を、ストレージＡＰＩ３００によってクエリされ、ナビゲートされ、保持される規範的データクラスで定義される機能と分離する。

【００１８】

ストレージＡＰＩ３００は、次のコアクラスから成り、図３は、ＳｔｏｒａｇｅＤｏｍａｉｎ、ＳｔｏｒａｇｅＣｏｎｔｅｘｔ、ＴａｂｌｅＳｅｔ、ＳｔｏｒａｇｅＳｅａｒｃｈｅｒ、およびＳｔｏｒａｇｅＶｉｅｗの間の関係を示している。これらのコアクラスのサポートで、追加のクラスを定義することができる。

【００１９】

ＴａｂｌｅＳｅｔ－ＴａｂｌｅＳｅｔクラスは、データモデルスキーマから生成することができ、スキーマ内で定義されたテーブルへの強く型付けされたアクセスを提供する。ＴａｂｌｅＳｅｔインスタンスは、１つまたは複数のＳｔｏｒａｇｅＣｏｎｔｅｘｔインスタンスをラップし、基礎となるＳｔｏｒａｇｅＣｏｎｔｅｘｔクラスおよび関連のＳｔｏｒａｇｅＤｏｍａｉｎクラスを使用して、オブジェクトのクエリ、ナビゲーション、および更新を行う。追加のメソッドを生成されたＴａｂｌｅＳｅｔクラスに追加して、スキーマ固有またはフレームワーク固有の機能を得ることができる。

【００２０】

ＳｔｏｒａｇｅＤｏｍａｉｎ－ほかのクラスが動作するストアを定義するクラスである。様々なタイプのストアがそれ自体の固有のＳｔｏｒａｇｅＤｏｍａｉｎクラスを実装する。ＳｔｏｒａｇｅＤｏｍａｉｎは、直接使用しても、ＴａｂｌｅＳｅｔと連結して使用しても良い。

【００２１】

ＳｔｏｒａｇｅＣｏｎｔｅｘｔ－セッションのコンテキストを提供するクラスである。ＳｔｏｒａｇｅＣｏｎｔｅｘｔクラスは、現在のコンテキスト内のオブジェクトへの変更をリフレッシュまたは保存するメソッドによって、識別管理、変更の追跡、および同時並行競合処理の範囲を定義する。ＳｔｏｒａｇｅＣｏｎｔｅｘｔクラスは、（例えばデータをリフレッシュする際、または修正を保持する際など）ストアと通信するためにＳｔｏｒａｇｅＤｏｍａｉｎクラスを使用する。ＳｔｏｒａｇｅＣｏｎｔｅｘｔは、直接使用しても、ＴａｂｌｅＳｅｔと連結して使用しても良い。

【００２２】

ＳｔｏｒａｇｅＳｅａｒｃｈｅｒ－ＳｔｏｒａｇｅＳｅａｒｃｈｅｒクラスは、データストアに対して結合可能なオブジェクトベースのクエリを構築するために使用される。ＳｔｏｒａｇｅＳｅａｒｃｈｅｒクラスは、一般にＳｔｏｒａｇｅＣｏｎｔｅｘｔ内で、ＳｔｏｒａｇｅＤｏｍａｉｎによって実行されるＳｔｏｒａｇｅＥｘｐｒｅｓｓｉｏｎクラスを生成する。ＳｔｏｒａｇｅＳｅａｒｃｈｅｒは、順方向のみに流されるやり方での結果の列挙や、上質でスクロール可能なＳｔｏｒａｇｅＶｉｅｗの構築をサポートする。

【００２３】

ＳｔｏｒａｇｅＶｉｅｗ－ＳｔｏｒａｇｅＶｉｅｗクラスは、１組の結果にわたるリッチなアプリケーションビューを提供する。ＳｔｏｒａｇｅＶｉｅｗは、フィルタリング、

ソート、スクロール、グループ分け、セクション分け、セクションの展開／折りたたみなどの操作をサポートする。

【００２４】

次に図４を参照すると、ここではデータモデルのストレージＡＰＩを提供する方法を示している。４００で、データストア上で使用するデータプラットフォーム（ＣＤＰなど）が受信される。４０２で、例えばエンティティ、リレーションシップ、拡張などＣＤＭ概念を表すベースクラスを含むＡＰＩが提供される。基礎となるデータプラットフォームの機能性を、本発明のＡＰＩで定義された共通ＣＤＭデータクラスを介して、オーバーレイアプリケーション（ｏｖｅｒｌａｙｉｎｇ　ａｐｐｌｉｃａｔｉｏｎ）およびアプリケーションフレームワークに提示することができる。４０４で、他のＡＰＩクラスが動作するデータストアを定義するクラスが提供される。４０６で、データストアに対するオブジェクトベースのクエリを構築するために使用されるクラスが提供される。４０８で、セッションコンテキストを定義し、識別管理、変更の追跡、競合処理などを含むクラスが提供される。４１０で、スキーマから生成され、スキーマのテーブルへの型付けされたアクセスを提供するクラスが提供される。４１２で、検索結果集合のビューを促進するクラスが提供される。４１４で、ＣＤＭスキーマのインスタンスによって表される特定のエンティティおよびリレーションシップを表す１組のドメイン固有クラスが定義される。

【００２５】

次の項は、共通データモデルのＡＰＩを構成するクラスおよびメンバの定義について詳述する。

【００２６】

ＳｔｏｒａｇｅＤｏｍａｉｎクラス。ＳｔｏｒａｇｅＤｏｍａｉｎクラスは、サーバ、認証、マッピングなどのストア情報をカプセル化するために使用される。ストア固有の情報を提供するために、ストアのタイプごとにストレージドメインのベースクラスが導出される。基本のＳｔｏｒａｇｅＤｏｍａｉｎタイプは、次のように定義することができる。

【００２７】

【表１】

```
public abstract class StorageDomain : IDisposable
{}
```

【００２８】

ＷｉｎＦＳＤｏｍａｉｎクラス。ＷｉｎＦＳストアに対するＳｔｏｒａｇｅＤｏｍａｉｎの例は、次のように表すことができる。

【００２９】

【表２】

```
public class WinFSDomain : StorageDomain
{
  public WinFSDomain();
  public WinFSDomain(string share);
}
```

【００３０】

ＷｉｎＦＳＤｏｍａｉｎコンストラクタは、例えばＵＮＣ（汎用命名規則）共有名により、ストア、およびストア内の範囲を指定する情報を取得することができる。あるいは、デフォルトのコンストラクタは、例えばデフォルトのストアのルートまでなど、デフォルトのストア情報を使用することができる。ＵＮＣは、ＵＮＩＸ（登録商標）コミュニティから生じた、ネットワークにおけるサーバ、プリンタ、および他のリソースを識別する標準（規格）である。ＵＮＣパスでは、コンピュータの名前の前にダブルスラッシュまたはバックスラッシュを使用する。

【００３１】

ＳｑｌＳｔｏｒａｇｅＤｏｍａｉｎクラス。リレーショナルストア（ＳＱＬデータベー

スなど）に対するＳｔｏｒａｇｅＤｏｍａｉｎの例は、次のように表すことができる。

【００３２】

【表３】

```
public class SqlStorageDomain : StorageDomain
{
  public SqlStorageDomain();
  public SqlStorageDomain(String connectionString);
  public SqlStorageDomain(SqlConnection connection, String
    mappingFile);
  public SqlStorageDomain(SqlConnection connection,
    IRelationalMapping mapping);
}
```

【００３３】

ＳｑｌＳｔｏｒａｇｅＤｏｍａｉｎコンストラクタは、例えば、接続情報およびマッピング情報を含む接続文字列、またはこのような情報を含む名前付き設定の形で接続情報を取得することができる。あるいは、コンストラクタは、標準マッピングインターフェースを実装するマッピングファイルまたはオブジェクトの形で、マッピング情報とともに接続オブジェクトを取得することができる。あるいは、デフォルトのコンストラクタは、例えば環境設定ファイル（ｃｏｎｆｉｇｕｒａｔｉｏｎ ｆｉｌｅ）からのデフォルトの接続情報またはマッピング情報を使用することができる。

【００３４】

図５は、テーブルセットタイプを提示する方法を示している。５００で、テーブルタイプのベースクラスが受信される。ＴａｂｌｅＳｅｔクラスは、テーブルセットタイプのベースクラスとして使用される。このタイプのインスタンスは、必要に応じて、アプリケーションによって直接作成し、使用することもできる。基本のＴａｂｌｅＳｅｔタイプは、以下のメンバを有する。

【００３５】

【表４】

```
public class TableSet : IDisposable
{

  public TableSet( StorageContext context, string tableSetName );
  public TableSet( StorageDomain domain, string tableSetName );
  public TableSet( StateManager manager, string tableSetName);

  public void Dispose();

  public StorageContext Context { get; }

  public string Name { get; }

  public Table<T> GetTable<T>(string propertyName);
  public object GetTableSetReference(string propertyName);

  public void SaveChanges();

}
```

【００３６】

ＴａｂｌｅＳｅｔは一般に、スキーマ内の１組のテーブルの名前で作られる。あるいは、スキーマ内の１組のテーブルを、例えばデフォルトの名前付け、環境設定ファイルなど、別の機構により決定することができる。ＴａｂｌｅＳｅｔを既存のＳｔｏｒａｇｅＣｏｎｔｅｘｔに関連付けるために、ＳｔｏｒａｇｅＣｏｎｔｅｘｔをＴａｂｌｅＳｅｔに提供することができる。あるいは、ＴａｂｌｅＳｅｔをＳｔｏｒａｇｅＤｏｍａｉｎに関連付けるために、ＳｔｏｒａｇｅＤｏｍａｉｎをＴａｂｌｅＳｅｔに提供することができる

。あるいは、さらに、ＴａｂｌｅＳｅｔに共通の状態マネージャ（管理プログラム）を提供することができる。

【００３７】

５０２で、テーブルセットに関連付けられているデータオブジェクトを保存するために、ＳａｖｅＣｈａｎｇｅｓメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。５０４で、提供された名前に基づいてスキーマ内のテーブルを表す（Ｔａｂｌｅ＜Ｔ＞など）オブジェクトを構築し、戻すために、ＧｅｔＴａｂｌｅメソッドを提供することができる。５０６で、ＴａｂｌｅＳｅｔＲｅｆｅｒｅｎｃｅを戻すためにＧｅｔＴａｂｌｅＳｅｔＲｅｆｅｒｅｎｃｅメソッドを提供することができる。

【００３８】

図６は、本発明のＡＰＩにおいてＷｉｎＦＳ機能を提供する方法を示している。６００で、ＷｉｎＦＳ機能のクラスが使用される。ＷｉｎＦＳ固有の機能を提供するために、ＷｉｎＦＳＤａｔａクラスをＴａｂｌｅＳｅｔクラスから導出することができる。ＷｉｎＦＳＤａｔａクラスは、次のメンバを有する。

【００３９】

【表５】

```
public partial WinFSData : TableSet {

  public WinFSData( StorageContext context );
  public WinFSData( StorageContext context, string tableSetName );
  public WinFSData();
  public WinFSData( string share );

  public Item GetRootItem() {}
  public Item GetItemByPath( string path ) {}

  public Table<Item> Items { get {} }
  public Table<Link> Links { get {} }
  public Table<ItemExtension> ItemExtensions { get {} }
  public Table<ItemFragment> ItemFragments { get {} }

  // Copy methods
  public Ref<Item> CopyItem(string sourceItemName, string
     destinationItemName );
  public Ref<Item> CopyItem(string sourceItemName, string
     destinationItemName,
                          CopyItemOptions options );
  public Ref<Item> CopyItem(string sourceItemName, string
     destinationItemName,
                          StorageContext destinationContext,
     CopyItemOptions options );
  public Ref<Item> CopyItem( Ref<Item> sourceItemRef,
                     Ref<Item> destinationContainerRef );
  public Ref<Item> CopyItem( Ref<Item> sourceItemRef,
                          Ref<Item> destinationContainerRef,
     CopyItemOptions options);
  public Ref<Item> CopyItem( Ref<Item> sourceItemRef,
                          Ref<Item> destinationContainerRef, string
     newNamespaceName,
                          CopyItemOptions options);
  public Ref<Item> CopyItem( Item sourceItem, Item
     destinationContainer );
  public Ref<Item> CopyItem( Item sourceItem, Item
     destinationContainer,
                           CopyItemOptions options);
  public Ref<Item> CopyItem( Item sourceItem, Item
     destinationContainer,
                          string newNamespaceName, CopyItemOptions
     options);

  // Move Methods
```

【００４０】

【表６】

```
public void MoveItem( string sourceItemName, string
   destinationItemName );
public void MoveItem( string sourceItemName, string
   destinationItemName,
                  MoveItemOptions options );
public void MoveItem( Ref<Item> sourceItemRef, Ref<Item>
   destinationContainerRef );
public void MoveItem( Ref<Item> sourceItemRef, Ref<Item>
   destinationContainerRef,
                     string newNamespaceName, MoveItemOptions
   options );
public void MoveItem( Item sourceItem, Item destinationContainer
   );
public void MoveItem( Item sourceItem, Item
   destinationContainer,
                     string newNamespaceName, MoveItemOptions
   options );

// Delete Methods
public void DeleteItem ( string itemName );
public void DeleteItem ( string itemName, ItemDeleteOptions
   options );
public void DeleteItem ( Ref<Item> itemRef );
public void DeleteItem ( Ref<Item> itemRef, ItemDeleteOptions
   options );
public void DeleteItem ( Item item );
public void DeleteItem ( Item item, ItemDeleteOptions options );
// Export Methods
public void ExportItem( string itemName, Stream stream );
public void ExportItem( string itemName, string fileName );
public void ExportItem( string itemName, string fileName,
            ExportItemOptions options );
public void ExportItem( Ref<Item> itemRef, Stream stream );
public void ExportItem( Ref<Item> itemRef, string fileName );
public void ExportItem( Ref<Item> itemRef, string fileName,
                ExportItemOptions options );
public void ExportItem( Item item, Stream stream );
public void ExportItem( Item item, string fileName );
public void ExportItem( Item item, string fileName,
   ExportItemOptions options );

// Import Methods
public void ImportItem( Stream stream, string itemName );
public void ImportItem( string fileName, string itemName );
public void ImportItem( string fileName, string itemName,
                        ImportItemOptions options );
public void ImportItem( Stream stream, Ref<Item>
   containerItemRef,
                        string namespaceName );
public void ImportItem( Stream stream, Ref<Item>
   containerItemRef,
                        string namespaceName );
public void ImportItem( string fileName, Ref<Item>
   containerItemRef,
                        string namespaceName );
public void ImportItem( string fileName, Ref<Item>
   containerItemRef,
                        string namespaceName,
   ImportItemOptions options );
public void ImportItem( Stream stream, Item containerItem,
   string uniuqeName );
```

【００４１】

【表７】

```
  public void ImportItem( string fileName, Item containerItem,
     string namespaceName );
  public void ImportItem( string fileName, Item item, string
     namespaceName,
                          ImportItemOptions options );
}
```

【００４２】

ＷｉｎＦＳＤａｔａコンストラクタは、既存のＳｔｏｒａｇｅＣｏｎｔｅｘｔで構築したり、指定された情報（ＵＮＣ共有など）またはデフォルトの情報（例えばデフォルトのストアのルート）を使用してＳｔｏｒａｇｅＣｏｎｔｅｘｔを作成したりすることができる。さらに、ＷｉｎＦＳＤａｔａクラスを特定の指名されたテーブルセットのインスタンスに関連付けるために、テーブルセット名を指定することができる。

【００４３】

６０２で、ドメインのルートを戻すためにＧｅｔＲｏｏｔＩｔｅｍメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。６０４で、そのパスが与えられるとアイテムを戻すために、ＧｅｔＩｔｅｍＢｙＰａｔｈメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。

【００４４】

６０６で、Ｉｔｅｍｓテーブル、ＩｔｅｍＥｘｔｅｎｓｉｏｎｓテーブル、およびＩｔｅｍＦｒａｇｍｅｎｔｓテーブルを表すオブジェクトを戻すために、Ｉｔｅｍｓプロパティ、ＩｔｅｍＥｘｔｅｎｓｉｏｎｓプロパティ、およびＩｔｅｍＦｒａｇｍｅｎｔｓプロパティを提供することができる。６０８で、Ｌｉｎｋｓテーブルを表すオブジェクトを戻すために、Ｌｉｎｋｓプロパティを提供することができる。６１０で、アイテムのコピー、移動、および削除を行うメソッドが提供される。指定されたアイテムをストア内の別の位置にコピーするために、ＣｏｐｙＩｔｅｍメソッドを提供することができる。指定されたアイテムをストア内に移動するために、ＭｏｖｅＩｔｅｍメソッドを提供することができる。ＤｅｌｅｔｅＩｔｅｍメソッドは、指定されたアイテムをストアから削除する。６１２で、アイテムのインポート、およびエクスポートを行うメソッドが提供される。指定されたアイテムをストアからエクスポートするために、ＥｘｐｏｒｔＩｔｅｍメソッドを提供することができる。指定されたアイテムをストアにインポートするために、ＩｍｐｏｒｔＩｔｅｍメソッドを提供することができる。ＣｏｐｙＩｔｅｍメソッド、ＭｏｖｅＩｔｅｍメソッド、ＤｅｌｅｔｅＩｔｅｍメソッド、ＥｘｐｏｒｔＩｔｅｍメソッド、ＩｍｐｏｒｔＩｔｅｍメソッドの非同期バージョンを提供することもできる。

【００４５】

図７は、ストア内のクラスを表す方法を示している。７００で、ストア内のある範囲を表すクラスが定義される。ストア内のある範囲を表すために、Ｔａｂｌｅ＜Ｔ＞クラスが使用される。Ｔａｂｌｅ＜Ｔ＞クラスは、オブジェクトをその範囲に追加し、またはそこから削除するメソッド、およびその範囲の内容にわたるＳｔｏｒａｇｅＳｅａｒｃｈｅｒを構築するメソッドを有することができる。

【００４６】

【表８】

```
public class Table<T> {
  public Table(StorageContext context, string TableName);
  public Table(StorageDomain domain, string TableName);

  public StorageContext Context { get; internal set;}
  public StorageDomain Domain { get; internal set;}

  public StorageSearcher<T> Searcher { get; }

  // Support ICollection
  bool ICollection<T>.Add(T obj);
  void ICollection<T>.Remove(T obj);
  void ICollection<T>.Clear();
  bool ICollection<T>.Contains(T t);
  public virtual int Count { get; }
  void ICollection<T>.CopyTo(T[] array, int arrayIndex);
  bool ICollection<T>.IsReadOnly { get { }}
}
```

【００４７】

Ｔａｂｌｅ＜Ｔ＞クラスは、そのスキーマでの対応するテーブルの名前とともに、ＳｔｏｒａｇｅＣｏｎｔｅｘｔまたはＳｔｏｒａｇｅＤｏｍａｉｎを指定する情報で構築することができる。７０２で、Ｔａｂｌｅ＜Ｔ＞クラスに関連付けられているＳｔｏｒａｇｅＣｏｎｔｅｘｔを戻すために、Ｃｏｎｔｅｘｔプロパティを提供することができる。７０４で、Ｔａｂｌｅ＜Ｔ＞クラスに関連付けられているＳｔｏｒａｇｅＤｏｍａｉｎを戻すために、Ｄｏｍａｉｎプロパティを提供することができる。７０６で、ストア内の対応するテーブルに対するＳｔｏｒａｇｅＳｅａｒｃｈｅｒを戻すために、Ｓｅａｒｃｈｅｒプロパティを提示することができる。７０８で、オブジェクトのアッド（追加）、リムーブ（削除・除去）、およびクリア（消去）を行うメソッドが提供される。テーブルにオブジェクトを追加するために、Ａｄｄメソッドを提示することができる。テーブルから削除すべきオブジェクトを指定するために、Ｒｅｍｏｖｅメソッドを提示することができる。テーブルをクリアするために、Ｃｌｅａｒメソッドを提示することができる。７１０で、テーブルが指定されたオブジェクトを含むかどうかを戻すために、Ｃｏｎｔａｉｎｓメソッドを提示することができる。７１２で、テーブル内のオブジェクトの総数を指定するために、Ｃｏｕｎｔメソッドを提示することができる。７１４で、オブジェクトをテーブルにコピー（複製）するメソッドが提供される。７１６で、テーブルが読み取り専用であるかどうかを提示するプロパティ（属性）が提供される。指定されたオブジェクトをテーブルにコピーするために、ＣｏｐｙＴｏメソッドを提示することができる。テーブルを追加したり削除したりすることができるかどうかを戻すために、ＩｓＲｅａｄＯｎｌｙプロパティを提示することができる。

【００４８】

図８は、クライアントと１つまたは複数のストアとの間の接続をカプセル化する方法を示している。さらに、クラスは、セッションコンテキスト、識別管理、変更の追跡、および同時並行競合処理の範囲を定義する。ＳｔｏｒａｇｅＣｏｎｔｅｘｔクラスは、クライアント(コンピュータ)と１つまたは複数のストアとの間の接続をカプセル化し、ＣＲＵＤ（作成、読み取り、更新、および削除）操作のゲートウェイである。

【００４９】

【表９】

```
public class StorageContext : IDisposable {

  public StorageContext();
  public StorageContext(StorageDomain domain);

  public object GetObjectByKey(StorageKey key);
  public StorageKey GetObjectKey(object o);
  public void SaveChanges();

  public void Refresh(RefreshMode options, IEnumerable<object>
     objects);
  public void Refresh(RefreshMode options, params object[] objects);

  public void Dispose();

  public StorageDomain Domain { get; }
  public void Add(object o);
  public void MarkForDeletion(object o);
}
```

【００５０】

ＳｔｏｒａｇｅＣｏｎｔｅｘｔは、ストア情報を提供するＳｔｏｒａｇｅＤｏｍａｉｎが与えられると構築される。あるいは、ＳｔｏｒａｇｅＣｏｎｔｅｘｔは、ＳｔｏｒａｇｅＤｏｍａｉｎ無しで構築することができ、環境設定ファイルなど、デフォルトのソースからストア情報を取得することができる。

【００５１】

８０２で、キーを介してオブジェクトを戻すメソッドが提供される。特定のキーに関連付けられているＳｔｏｒａｇｅＣｏｎｔｅｘｔ内のオブジェクトを戻すために、ＧｅｔＯｂｊｅｃｔＢｙＫｅｙメソッドを提供することができる。代わりに、このメソッドを取り出して、個別のＳｔａｔｅＭａｎａｇｅｍｅｎｔオブジェクトに入れることができる。このメソッドの非同期バージョンを提供することもできる。８０４で、ＳｔｏｒａｇｅＣｏｎｔｅｘｔ内の特定のオブジェクトに関連付けられているキーを戻すために、ＧｅｔＯｂｊｅｃｔＫｅｙメソッドを提供することができる。代わりに、このメソッドを取り出して、個別のＳｔａｔｅＭａｎａｇｅｍｅｎｔオブジェクトに入れることができる。８０６で、ＳｔｏｒａｇｅＣｏｎｔｅｘｔ内のオブジェクトへの追加、削除、または修正を保存するために、ＳａｖｅＣｈａｎｇｅｓメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。

【００５２】

８０８で、ＳｔｏｒａｇｅＣｏｎｔｅｘｔ内のオブジェクトを現在のストア値でリフレッシュするために、Ｒｅｆｒｅｓｈメソッドを提供することができる。リフレッシュする明示的な１組のオブジェクトは、例えば列挙子（ｅｎｕｍｅｒａｔｏｒ）を介して、またはパラメータとして指定することができる。修正の競合の処理方法を制御するために、追加のオプションを指定することができる。このメソッドの非同期バージョンを提供することもできる。８１０で、新しいオブジェクトをＳｔｏｒａｇｅＣｏｎｔｅｘｔに関連付けるために、Ａｄｄメソッドを提供することができる。代わりに、このメソッドを取り出して、個別のＳｔａｔｅＭａｎａｇｅｍｅｎｔオブジェクトに入れることができる。８１２で、ＳａｖｅＣｈａｎｇｅｓが呼び出されたときに削除すべきＳｔｏｒａｇｅＣｏｎｔｅｘｔ内のオブジェクトをマークするために、ＭａｒｋＦｏｒＤｅｌｅｔｉｏｎメソッドを提供することができる。代わりに、このメソッドを取り出して、個別のＳｔａｔｅＭａｎａｇｅｍｅｎｔオブジェクトに入れることができる。８１４で、ＳｔｏｒａｇｅＣｏｎｔｅｘｔに関連付けられているＳｔｏｒａｇｅＤｏｍａｉｎを戻すために、ＳｔｏｒａｇｅＤｏｍａｉｎプロパティを提供することができる。

【００５３】

図９は、ストアに対するクエリを構築する方法を示している。９００で、ストアに対するクエリを構築するベースクラスが定義される。ＳｔｏｒａｇｅＳｅａｒｃｈｅｒクラスは、ストアに対する結合可能なオブジェクトベースのクエリを構築するために使用される。ＳｔｏｒａｇｅＳｅａｒｃｈｅｒは、一般にＳｔｏｒａｇｅＣｏｎｔｅｘｔ内で、ＳｔｏｒａｇｅＤｏｍａｉｎによって実行されるＳｔｏｒａｇｅＥｘｐｒｅｓｓｉｏｎを生成する。ＳｔｏｒａｇｅＳｅａｒｃｈｅｒは、順方向のみに流されるやり方での結果の列挙や、リッチでスクロール可能なＳｔｏｒａｇｅＶｉｅｗの構築をサポートする。

【００５４】

【表１０】

```
public class StorageSearcher<T> : IStorageSearcher, IEnumerable<T>
  where T : class
{

  public StorageSearcher(string expression);
  public StorageSearcher(string expression, object[] parameters);
  public StorageSearcher(string expression, object[] parameters,
    StorageContext context);
  public StorageSearcher(StorageExpression expression);
  public StorageSearcher(StorageExpression expression,
    StorageContext context);
  public StorageSearcher(string expression, object[] parameters,
    StorageDomain store);
  public StorageSearcher(StorageExpression expression,
    StorageDomain store);

  public StorageContext Context { get; }
  public StorageDomain Domain { get; }

  public StorageExpression Expression { get; }
  Type IStorageSearcher.ResultType { get; }

  public StorageSearcher<T> BindContext(StorageContext context);
  IStorageSearcher IStorageSearcher.BindContext(StorageContext
    context);

  public StorageSearcher<T> BindParameters(IDictionary<string,
    object> parameters);
  IStorageSearcher IStorageSearcher.BindParameters(IDictionary
    parameters);

  public StorageSearcher<T> Filter(string expression, params
    object[] parameters);
  public StorageSearcher<U> FilterByType<U>() where U : T;
  public StorageSearcher<U> TreatAsType<U>();
  public StorageSearcher<T> Sort(string expression, params
    object[] parameters);
  public StorageSearcher<StorageRecord> Project(string expression,
    params object[] parameters);
  public StorageSearcher<StorageRecord> Group(string expression,
    params object[] parameters);
  public StorageSearcher<T> Union(StorageSearcher<T> searcher);
```

【００５５】

【表１１】

```
  public StorageSearcher<U> Query<U>(string expression, params
     object[] parameters);
  public StorageSearcher<U> Query<U>(StorageExpression
     expression);
  IStorageSearcher IStorageSearcher.Query(Type resultType,
     StorageExpression expression);
  IStorageSearcher IStorageSearcher.Query(Type resultType, string
     expression,
      params object[] parameters);

  public IEnumerator<T> GetEnumerator();
  IEnumerator IEnumerable.GetEnumerator();

  public T GetFirst();
  object IStorageSearcher.GetFirst();
  public int GetCount();
  public List<T> GetList();

  public StorageView<StorageViewRecord> CreateView();
  public StorageView<StorageViewRecord>
     CreateView(StorageViewDefinition definition);
  public StorageView<StorageViewRecord> CreateView(
     StorageViewDefinition definition, StorageViewOptions options);
  public StorageView<T> CreateView<T>(StorageViewDefinition
     definition, StorageViewOptions options) where T :
     StorageViewRecord {}
}
```

【００５６】

ＳｔｏｒａｇｅＳｅａｒｃｈｅｒは、ＳｔｏｒａｇｅＳｅａｒｃｈｅｒをバインドするコンテキストまたはストアを指定するために、ＳｔｏｒａｇｅＣｏｎｔｅｘｔまたはＳｔｏｒａｇｅＤｏｍａｉｎで構築することができる。さらに、ＳｔｏｒａｇｅＳｅａｒｃｈｅｒを文字列またはＳｔｏｒａｇｅＥｘｐｒｅｓｓｉｏｎオブジェクトツリーとして初期化するために、クエリ表現（ｑｕｅｒｙ　ｅｘｐｒｅｓｓｉｏｎ）を指定することができる。

【００５７】

９０２で、任意のクエリ表現をカプセル化する新しいＳｔｏｒａｇｅＳｅａｒｃｈｅｒを構築するために、Ｑｕｅｒｙメソッドを提供することができる。９０４で、フィルタメソッドが提供される。入力サーチャー（入力探索部）によって生成されるクエリ結果にわたるフィルタをカプセル化する新しいＳｔｏｒａｇｅＳｅａｒｃｈｅｒを構築するために、Ｆｉｌｔｅｒメソッドを提供することができる。入力サーチャーによって生成されるクエリ結果にわたるフィルタをカプセル化する新しいＳｔｏｒａｇｅＳｅａｒｃｈｅｒを構築するために、ＦｉｌｔｅｒＢｙＴｙｐｅメソッドを提供することができる。入力サーチャーによって生成されるクエリ結果を異なるタイプと見なす新しいＳｔｏｒａｇｅＳｅａｒｃｈｅｒを構築するために、ＴｒｅａｔＡｓＴｙｐｅメソッドを提供することができる。

【００５８】

９０６で、Ｓｏｒｔ（ソート）メソッド、Ｐｒｏｊｅｃｔ（投影）メソッド、Ｇｒｏｕｐ(グループ分け)メソッド、Ｕｎｉｏｎ（統合）メソッドが提供される。入力サーチャーによって生成される一種のクエリ結果をカプセル化する新しいＳｔｏｒａｇｅＳｅａｒｃｈｅｒを構築するために、Ｓｏｒｔメソッドを提供することができる。入力サーチャーによって生成されるクエリ結果の投影をカプセル化する新しいＳｔｏｒａｇｅＳｅａｒｃｈｅｒを構築するために、Ｐｒｏｊｅｃｔメソッドを提供することができる。入力サーチャーによって生成されるクエリ結果のグループ分けをカプセル化する新しいＳｔｏｒａｇｅＳｅａｒｃｈｅｒを構築するために、Ｇｒｏｕｐメソッドを提供することができる。２つ

の入力サーチャーによって生成されるクエリ結果の結合をカプセル化する新しいＳｔｏｒａｇｅＳｅａｒｃｈｅｒを構築するために、Ｕｎｉｏｎメソッドを提供することができる。上記は一例にすぎず、限定するものと解釈されないものとする。追加のクエリ操作を表すために、ＳｔｏｒａｇｅＳｅａｒｃｈｅｒに対して追加のメソッドを提供することができる。言い換えれば、クエリ操作は、新しいＳｔｏｒａｇｅＳｅａｒｃｈｅｒｓを戻すＳｔｏｒａｇｅＳｅａｒｃｈｅｒクラスに対するメソッドとして提示することができる。

【００５９】

９０８で、クエリ結果にアクセスするために使用することができる列挙子を戻すために、ＧｅｔＥｎｕｍｅｒａｔｏｒメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。９１０で、クエリの最初の結果を戻すメソッドが提供される。これらのメソッドの非同期バージョンを提供することもできる。最初の結果を戻すために、ＧｅｔＦｉｒｓｔメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。結果数を戻すために、ＧｅｔＣｏｕｎｔメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。９１２で、ＳｔｏｒａｇｅＳｅａｒｃｈｅｒクエリからＳｔｏｒａｇｅＶｉｅｗを作成するために、ＣｒｅａｔｅＶｉｅｗメソッドを提供することができる。ＣｒｅａｔｅＶｉｅｗメソッドは、ビューに固有の情報を指定する追加のオプション有り、または追加のオプション無しで、ＳｔｏｒａｇｅＶｉｅｗＤｅｆｉｎｉｔｉｏｎを取得することができる。

【００６０】

ＳｔｏｒａｇｅＲｅｃｏｒｄクラスは、クエリが特定のアプリケーション定義のどんなタイプにも対応しないデータを戻したときのサーチャーの結果タイプとして使用される。例えば、Ｐｒｏｊｅｃｔ操作またはＧｒｏｕｐ操作の結果がＳｔｏｒａｇｅＲｅｃｏｒｄオブジェクトの集まりとなる。

【００６１】

【表１２】

```
// StorageRecord represents a value in a structurally typed query
     result.
public class StorageRecord :
     System.ComponentModel.ICustomTypeDescriptor
{
   // Gets the value of a field
   public object this[string name] { get; }
}
```

【００６２】

図１０は、１組の結果にわたる表示の方法を示している。１０００で、結果を表示するクラスが提供される。ＳｔｏｒａｇｅＶｉｅｗクラスは、１組の結果にわたるリッチなアプリケーションビューを提供する。ＳｔｏｒａｇｅＶｉｅｗは、フィルタリング、ソート、スクロール、グループ分け、セクション分け、セクションの展開／折りたたみなどの操作をサポートする。

【００６３】

【表１３】

```
sealed public class StorageView<T> : IVirtualList,
     IServiceContainer, IEnumerable, IListSource, IDisposable
   where T: StorageViewRecord
{
   public StorageViewDefinition CopyDefinition() {}
   public void ApplyDefinition(StorageViewDefinition definition) {
      }

   public int Count { get;}
   public T Current { get;}
   public T this[ViewRecordBookmark bookmark] {get;}

   public T FindRecord(ViewRecordBookmark bookmark, bool forward,
     string expression, params object[] parameters) {}
   public T FindRecord(StorageViewSeekOrigin seekOrigin, bool
     forward, string expression, params object[] parameters) {}

   public void MoveCurrentPosition(StorageViewSeekOrigin
     seekOrigin, int offset);
   public void MoveCurrentPosition(ViewRecordBookmark bookmark, int
     offset);

   public void Refresh() { }

   public ViewRecordBookmark GetBookmarkFromBinary(byte[] bookmark)
     { }
   public byte[] GetBinaryFromBookmark(ViewRecordBookmark bookmark)
     { }

   public void CollapseSection(params object[] sectionValues){}
   public void CollapseSection(ViewRecordBookmark bookmark){}
   public void ExpandSection(params object[] sectionValues){}
   public void ExpandSection(ViewRecordBookmark bookmark){}
   public void CollapseAllSections() { }
   public void ExpandAllSections() { }
   public void ExpandSectionLevel(int sectionLevel){}
   public void LoadSectionExpandState(System.Xml.XmlReader reader);
   public void SaveSectionExpandState(System.Xml.XmlWriter writer);

   public void SetExtendedFields(StorageViewRecord[] records,
     string fields);

   public IList IListSource.GetList();

   public event ViewChangedEventHandler ViewChanged;
}
```

【００６４】

１００２で、ＳｔｏｒａｇｅＶｉｅｗＤｅｆｉｎｉｔｉｏｎの新しいインスタンスを作成するために、ＣｏｐｙＤｅｆｉｎｉｔｉｏｎメソッドを提供することができる。指定されたＳｔｏｒａｇｅＶｉｅｗＤｅｆｉｎｉｔｉｏｎを現在のＳｔｏｒａｇｅＶｉｅｗに適用するために、ＡｐｐｌｙＤｅｆｉｎｉｔｉｏｎメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。１００４で、レコード（１件分のデータ）を検索する、レコードカウントを戻す、および現在のレコードを戻すためのメソッドが提供される。指定された位置またはブックマークを基準にして、指定されたフィルタに従って現在のＳｔｏｒａｇｅＶｉｅｗ内でＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄを見つけるために、ＦｉｎｄＲｅｃｏｒｄメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。現在のＳｔｏｒａｇｅＶｉｅｗ内のレコードカウントを戻すために、Ｃｏｕｎｔメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。ＳｔｏｒａｇｅＶｉｅｗ内の現在のＳｔｏｒａ

ｇｅＶｉｅｗＲｅｃｏｒｄを戻すために、Ｃｕｒｒｅｎｔメソッドを提供することができる。

【００６５】

１００６で、所与のブックマークについてＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄを戻すために、インデックス付きアクセサ（ａｃｃｅｓｓｏｒ：アクセス機構ともいう）（ｔｈｉｓ［］など）を提供することができる。このメソッドの非同期バージョンを提供することもできる。１００８で、位置を移動し、ビューをリフレッシュするメソッドが提供される。指定された位置またはブックマークおよびオフセットに従って、ＳｔｏｒａｇｅＶｉｅｗ内で現在の位置を移動するために、ＭｏｖｅＣｕｒｒｅｎｔＰｏｓｉｔｉｏｎメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。静的ＳｔｏｒａｇｅＶｉｅｗ内のデータをストアからの現在の値でリフレッシュするために、Ｒｅｆｒｅｓｈメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。１０１０で、ブックマークおよびその２進表現を取得するメソッドが提供される。持続性の２進表現からブックマークを取得するために、ＧｅｔＢｏｏｋｍａｒｋＦｒｏｍＢｉｎａｒｙメソッドを提供することができる。ブックマークから持続性の２進表現を取得するために、ＧｅｔＢｉｎａｒｙＦｒｏｍＢｏｏｋｍａｒｋメソッドを提供することができる。

【００６６】

１０１２で、セクション、レベル、およびフィールドを展開し、折りたたむメソッドが提供される。ＳｔｏｒａｇｅＶｉｅｗ内で定義されたすべてのセクションを折りたたむために、ＣｏｌｌａｐｓｅＡｌｌＳｅｃｔｉｏｎｓメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。ＳｔｏｒａｇｅＶｉｅｗ内で定義されたすべてのセクションを展開するために、ＥｘｐａｎｄＡｌｌＳｅｃｔｉｏｎｓメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。指定されたレベルを含むすべてのセクションを展開するために、ＥｘｐａｎｄＳｅｃｔｉｏｎＬｅｖｅｌメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。

【００６７】

１０１４で、レコードのフィールドを展開するメソッドが提供される。１組のＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄｓに関連付けられている展開されたフィールドを定義するために、ＳｅｔＥｘｔｅｎｄｅｄＦｉｅｌｄｓメソッドを提供することができる。１０１６で、展開されたセクションの状態を保存し、ロードするメソッドが提供される。展開された１組のセクションを指定する状態をロードするために、ＬｏａｄＳｅｃｔｉｏｎＥｘｐａｎｄＳｔａｔｅメソッドを提供することができる。このメソッドの非同期バージョンを提供することもできる。展開された１組のセクションを指定する状態を保存するために、ＳａｖｅＳｅｃｔｉｏｎＥｘｐａｎｄＳｔａｔｅメソッドを提供することができる。ＳｔｏｒａｇｅＶｉｅｗは、ＳｔｏｒａｇｅＶｉｅｗが変わったときにリスナ（待ち受けプログラム）に通知するＶｉｅｗＣｈａｎｇｅｄイベントを提示することができる。

【００６８】

図１１は、結果にわたるデータの初期ビューを提示する方法を示している。１１００で、データの初期ビューを定義するクラスが提供される。ＳｔｏｒａｇｅＶｉｅｗＤｅｆｉｎｉｔｉｏｎクラスは、ＳｔｏｒａｇｅＳｅａｒｃｈｅｒによって定義された結果にわたるデータの初期ビューを定義する。

【００６９】

【表１４】

```
public class StorageViewDefinition
{
  public string Sort { get; set;}
  public IList<StorageViewSection> Sections { get;}
  public int SectionExpandLevel { get; set; }

  public string Filter { get; set;}
  public void SetFilter(string expression, params object[]
    parameters);

  public string Fields { get; set;}
  public void SetFields(string expression, params object[]
    parameters);

  public IDictionary<string,object> Parameters { get;}

  public bool AutoRefresh { get; set; }

  public int PageSize { get; set;}

}
```

【００７０】

１１０２で、ＳｔｏｒａｇｅＶｉｅｗのソート基準を取得または設定するために、Ｓｏｒｔプロパティを提供することができる。１１０４で、セクションの変更、および展開を行うプロパティが提供される。ＳｔｏｒａｇｅＶｉｅｗ内で定義されたＳｅｃｔｉｏｎｓのリストを変更するために、Ｓｅｃｔｉｏｎｓプロパティを提供することができる。指定されたレベルを含むセクションを展開するために、ＳｅｃｔｉｏｎＥｘｐａｎｄＬｅｖｅｌプロパティを提供することができる。１１０６で、操作をフィルタリングするプロパティおよびメソッドが提供される。ＳｔｏｒａｇｅＶｉｅｗをフィルタにかけて、指定されたフィルタ状態に一致するＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄｓのみを提示するために、Ｆｉｌｔｅｒプロパティを提示することができる。ＳｔｏｒａｇｅＶｉｅｗをフィルタにかけて、指定されたパラメータを使用して指定されたフィルタ状態に一致するＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄｓのみを提示するために、ＳｅｔＦｉｌｔｅｒプロパティを提示することができる。１１０８で、提示されたフィールドを制限するプロパティおよびメソッドが提供される。ＳｔｏｒａｇｅＶｉｅｗによって提示されたフィールドを指定されたＦｉｅｌｄｓに制限するために、Ｆｉｅｌｄｓプロパティを提示することができる。ＳｔｏｒａｇｅＶｉｅｗによって提示されたフィールドを、指定されたパラメータを使用して指定されたＦｉｅｌｄｓに制限するために、Ｆｉｅｌｄｓメソッドを提示することができる。

【００７１】

１１１０で、フィルタ、ソート、およびセクションによって使用されるパラメータを列挙するコレクションが提供される。フィルタ、ソート、およびセクションの指定によって使用されるパラメータを列挙するＰａｒａｍｅｔｅｒｓコレクションを提供することができる。１１１２で、ＳｔｏｒａｇｅＶｉｅｗがストアへの変更と同期して自動的に保持されるかどうかを指定するために、ブール（Ｂｏｏｌｅａｎ）のＡｕｔｏＲｅｆｒｅｓｈプロパティを提示することができる。１１１４で、ストアから一度に取り出すべきＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄｓの数を指定するために、ＰａｇｅＳｉｚｅプロパティを提示することができる。

【００７２】

図１２は、ストレージレコードクラス（ｓｔｏｒａｇｅ　ｒｅｃｏｒｄ　ｃｌａｓｓ）を拡張する方法を示している。１２００で、ビュープロパティを追加するクラスが提供される。ＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄは、ＳｔｏｒａｇｅＲｅｃｏｒｄを拡張し、セクション分け情報、ブックマーク、およびフィールドセッターなど、ＳｔｏｒａｇｅＶ

ｉｅｗ固有のプロパティを追加する。ＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄｓは、ＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄ内の値が修正されると、リスナに通知するようＩＰｒｏｐｅｒｔｙＣｈａｎｇｅをサポートする。

【００７３】

【表１５】

```
public class StorageViewRecord : StorageRecord, IPropertyChange
{
  public virtual bool IsSectionRecord { get; }

  public virtual int SectionLevel { get; }
  public virtual string SectionName { get; }

  public virtual bool IsSectionExpanded { get; }

  public virtual ViewRecordBookmark Bookmark { get; }

  protected virtual void SetValueinRecord(int i, object value);
  protected virtual void SetValueinRecord(string name, object
    value);

}
```

【００７４】

１２０２で、ＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄがＳｔｏｒａｇｅＶｉｅｗ内のセクションヘッダーレコード（ｓｅｃｔｉｏｎ ｈｅａｄｅｒ ｒｅｃｏｒｄ）を表すかどうかを戻すために、ＩｓＳｅｃｔｉｏｎＲｅｃｏｒｄプロパティを提示することができる。１２０４で、セクション情報のプロパティが提供される。ＳｔｏｒａｇｅＶｉｅｗ内のＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄのレベルを戻すために、ＳｅｃｔｉｏｎＬｅｖｅｌプロパティを提示することができる。ＳｔｏｒａｇｅＶｉｅｗ内のセクションの名前を戻すために、ＳｅｃｔｉｏｎＮａｍｅプロパティを提示することができる。セクションが展開されるかどうかを戻すために、ＩｓＳｅｃｔｉｏｎＥｘｐａｎｄｅｄプロパティを提示することができる。１２０６で、現在のＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄのブックマークを戻すために、Ｂｏｏｋｍａｒｋプロパティを提示することができる。１２０８で、ＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄ内で指定されたフィールドの値を設定するために、ＳｅｔＶａｌｕｅｉｎＲｅｃｏｒｄメソッドを提示することができる。フィールドは、名前または序数によって指定することができる。

【００７５】

ＳｔｏｒａｇｅＶｉｅｗＳｅｃｔｉｏｎは、ＳｔｏｒａｇｅＶｉｅｗ内のセクション（グループ）を定義するために使用される。

【００７６】

【表１６】

```
public class StorageViewSection
{
  public StorageViewSection(string field) {}

  public string Field { get; }

  public string AggregateFields { get; set; }

  public string Sort { get; set; }

  public string Having { get; set; }
  public void SetHaving(string expression, params object[]
    parameters);
}
```

【００７７】
セクションが定義されているＳｔｏｒａｇｅＶｉｅｗ内にフィールドを指定するＳｔｏｒａｇｅＶｉｅｗＳｅｃｔｉｏｎを構築することができる。セクションが定義されているＳｔｏｒａｇｅＶｉｅｗ内のフィールドを戻すために、Ｆｉｅｌｄプロパティを提示することができる。セクションを算出する集合（ａｇｇｒｅｇａｔｅ）を取得または設定するために、ＡｇｇｒｅｇａｔｅＦｉｅｌｄｓプロパティを提示することができる。セクション内のＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄｓの順序を指定するために、Ｓｏｒｔプロパティを提示することができる。指定されたＡｇｇｒｅｇａｔｅフィールドに従ってＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄｓを制限するために、Ｈａｖｉｎｇプロパティを提示することができる。１組のパラメータとともに、指定されたＡｇｇｒｅｇａｔｅフィールドに従ってＳｔｏｒａｇｅＶｉｅｗＲｅｃｏｒｄｓを制限するために、ＳｅｔＨａｖｉｎｇメソッドを提示することができる。

【００７８】
ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎ＜Ｔ＞クラスは、その母集団（ｐｏｐｕｌａｔｉｏｎ）を遅らせることができる、強く型付けされたオブジェクトの集まりを表すために使用される。例えば、ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎは、親オブジェクトのコレクションプロパティで使用することができる。ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎは、その内容がアクセスされると、明示的または暗黙的に投入される。

【００７９】
【表１７】

```
public class StorageCollection<T> : ICollection<T>, IBindingList,
     ITypedList {
  public StorageCollection();
  public StorageCollection(object parent, StorageContext ctx,
     string role);
  public StorageContext Context { get; internal set;}
  public StorageDomain Domain { get; internal set;}

  public void Fill();
  public void Fill(StorageSearcher<T> searcher);
  public void Fill(IEnumerable<T> values);

  public bool IsFilled { get;}
  public void Reset();

  public StorageSearcher<T> Searcher { get; }

  public IEnumerator<T> GetEnumerator();

  // Support for ICollection
  bool ICollection<T>.Add(T obj);
  void ICollection<T>.Remove(T obj);
  void ICollection<T>.Clear();
  bool ICollection<T>.Contains(T t);
  public virtual int Count { get; }
  void ICollection<T>.CopyTo(T[] array, int arrayIndex);
  bool ICollection<T>.IsReadOnly { get { }}
}
```

【００８０】
ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎは、例えばＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎが親オブジェクトのコレクションプロパティ内のオブジェクトを表す場合、ＳｔｏｒａｇｅＣｏｎｔｅｘｔまたはＳｔｏｒａｇｅＤｏｍａｉｎ、親オブジェクト、およびＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎに関連付けられている役割(ｒｏｌｅ)を指定する情報で構築することができる。

【００８１】

ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎに関連付けられているＳｔｏｒａｇｅＣｏｎｔｅｘｔを戻すために、Ｃｏｎｔｅｘｔプロパティを提供することができる。ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎに関連付けられているＳｔｏｒａｇｅＤｏｍａｉｎを戻すために、Ｄｏｍａｉｎプロパティを提供することができる。コレクションにオブジェクトを追加するために、Ｆｉｌｌメソッドを提供することができる。Ｆｉｌｌメソッドは、ＩＥｎｕｍｅｒａｂｌｅ＜Ｔ＞またはＳｔｏｒａｇｅＳｅａｒｃｈｅｒを取得し、またはＳｔｏｒａｇｅＤｏｍａｉｎまたはＳｔｏｒａｇｅＣｏｎｔｅｘｔともに親プロパティおよびロールプロパティを使用して、ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎを投入する要求を生成することができる。ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎが投入されたかどうかを戻すために、ＩｓＦｉｌｌｅｄプロパティを提示することができる。ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎをリセットするために、Ｒｅｓｅｔメソッドを提示することができる。

【００８２】

コレクションの定義に対応するストアに対するＳｔｏｒａｇｅＳｅａｒｃｈｅｒを戻すために、Ｓｅａｒｃｈｅｒプロパティを提示することができる。ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎの内容にわたる列挙子を戻すために、ＧｅｔＥｎｕｍｅｒａｔｏｒメソッドを提示することができる。オブジェクトをＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎに追加するために、Ａｄｄメソッドを提示することができる。ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎからオブジェクトを削除するために、Ｒｅｍｏｖｅメソッドを提示することができる。ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎをクリアするために、Ｃｌｅａｒメソッドを提示することができる。ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎが指定されたオブジェクトインスタンスを含むかどうかを戻すために、Ｃｏｎｔａｉｎｓメソッドを提示することができる。ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎ内のオブジェクトの総数を指定するために、Ｃｏｕｎｔメソッドを提示することができる。指定されたオブジェクトをＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎにコピーするために、ＣｏｐｙＴｏメソッドを提示することができる。ＳｔｏｒａｇｅＣｏｌｌｅｃｔｉｏｎを追加したり削除したりすることができるかどうかを戻すために、ＩｓＲｅａｄＯｎｌｙプロパティを提示することができる。

【００８３】

図１３は、ストレージＡＰＩ１００をＣＤＰ１３０２に使用するシステム１３００を示している。ＣＤＰ１３０２は、データアプリケーションおよびアプリケーションフレームワーク１３０４とデータストア１３０６上のデータとの間のデータ管理を提供するために使用される。ＣＤＰ１３２０は、アプリケーションフレームワークおよびそれに関連付けられているエンドユーザアプリケーションにわたって共通のデータサービスを提供する。ＣＤＰ１３０２は、アプリケーションおよびアプリケーションフレームワーク１３０４、ランタイムコンポーネント１３０８、および制約／セキュリティエンジンコンポーネント１３１０とのインターフェースを円滑にするＡＰＩ１００をさらに含む。ＡＰＩ１００は、パブリック（公開）クラス、インターフェース、および静的ヘルパー機能の形のＣＤＰ１３０２を使用してアプリケーションのプログラミングインターフェースを提供する。例には、ＳｔｏｒａｇｅＣｏｎｔｅｘｔ、ＳｔｏｒａｇｅＳｅａｒｃｈｅｒ、Ｅｎｔｉｔｙ、Ｅｎｔｉｔｙ、ＴａｂｌｅＳｅｔ、Ｔａｂｌｅ、ＥｎｔｉｔｙＲｅｆｅｒｅｎｃｅ、ＴａｂｌｅＲｅｆｅｒｅｎｃｅなどがある。

【００８４】

ＣＤＰランタイムコンポーネント１３０８は、パブリックＡＰ層１００で提示される様々な機能を実施する層である。これは、オブジェクトリレーショナルマッピングおよびクエリマッピングを提供する、データモデルの制約を実施するなどによって、共通データモデルを実装する。より詳細には、ＣＤＰランタイム１３０８は、共通データモデルコンポーネントの実装、クエリプロセッサコンポーネント、セッションおよびトランザクションコンポーネント、セッションキャッシュおよび明示的なキャッシュを含むことができるオブジェクトキャッシュ、変更の追跡、競合の検出、およびイベンティング（ｅｖｅｎｔｉｎｇ：イベントのやり取り）を含むサービスコンポーネント、カーソルおよびルール（罫線）コンポーネント、ビジネスロジックホスティング（ｂｕｓｉｎｅｓｓ　ｌｏｇｉｃ

ｈｏｓｔｉｎｇ）コンポーネント、およびコアの持続性およびクエリサービスを提供する持続およびクエリエンジンを含む。持続性およびクエリサービスの中には、クエリ／更新マッピングを含むオブジェクトリレーショナルマッピングがある。ＣＤＰ１３０２は、データストア１３０６に対する制約や、ロールベースのセキュリティなどのセキュリティポリシーを適用する制約／セキュリティエンジン１３１０も含む。

【００８５】

次に図１４を参照すると、開示のＡＰＩアーキテクチャを実行するように動作可能なコンピュータのブロック図を示している。本発明の様々な態様に関する状況をさらに提示するために、図１４および以下の説明は、本発明の様々な態様を実施できるように適したコンピューティング環境１４００の簡潔な概要説明を行なう。本発明は、１つまたは複数のコンピュータ上で稼働し得るコンピュータ実行可能命令の一般的な文脈で上述してきたが、本発明を他のプログラムモジュールとの組合せで実施する、かつ／またはハードウェアおよびソフトウェアの組合せとして実施することもできることを、当業者であれば理解されよう。

【００８６】

一般にプログラムモジュールは、特定のタスクを実行し、または特定の抽象データ型を実装するルーチン、プログラム、コンポーネント、データ構造などを含む。さらに、本方法は、単一のプロセッサまたはマルチプロセッサのコンピュータシステム、ミニコンピュータ、メインフレームコンピュータ、パーソナルコンピュータ、ハンドヘルドコンピューティング装置、マイクロプロセッサベースおよび／またはプログラム可能家庭用電化製品など、１つまたは複数の関連の装置に操作可能にそれぞれ結合することができる他のコンピュータシステム構成で実施できることを当業者であれば理解されよう。

【００８７】

また、本発明の態様例は、いくつかのタスクが通信ネットワークによってリンクされているリモート（遠隔）処理装置によって実行される分散コンピューティング環境でも実施できる。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を、ローカル（現地）およびリモート（遠隔）のメモリ記憶装置に置くことができる。

【００８８】

コンピュータは、一般に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、コンピュータからアクセスできる使用可能な任意の媒体とすることができ、揮発性および不揮発性媒体、取外式および固定式媒体を含む。コンピュータ可読媒体は、それだけには限定されないが一例として、コンピュータ記憶媒体および通信媒体を含み得る。コンピュータ記憶媒体には、コンピュータ可読命令、データ構造、プログラムモジュール、他のデータなど、情報を記憶するための任意の方法または技術で実施される揮発性および不揮発性の取外式および固定式媒体がある。コンピュータ記憶媒体には、それだけには限定されないが、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリまたは他のメモリ技術、ＣＤ－ＲＯＭ、デジタルビデオディスク（ＤＶＤ）または他の光ディスク記憶装置、磁気カセット、磁気テープ、磁気ディスク記憶装置または他の磁気記憶装置、または所望の情報の格納に使用でき、コンピュータからアクセスできる他の任意の媒体などがある。

【００８９】

通信媒体は、一般に、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搬送波または他の移送機構などの変調されたデータ信号が取り込まれるもので、これには任意の情報配信媒体を含む。「変調されたデータ信号」という用語は、信号内に情報を符号化するような方法で信号の１つまたは複数の特性が設定または変更された信号を意味する。通信媒体には、それだけには限定されないが一例として、有線ネットワーク、直接結線された接続などの有線媒体、および音響、ＲＦ(無線周波数)、赤外線、その他の無線媒体などのワイヤレス媒体がある。また、上記のどんな組合せでもコンピュータ可読媒体の範囲内に含まれるものとする。

【００９０】

再度図１４を参照すると、本発明の様々な態様を実装する環境１４００の例は、コンピ

ュータ１４０２を含んでおり、コンピュータ１４０２は、処理ユニット１４０４、システムメモリ１４０６、およびシステムバス１４０８を含んでいる。システムバス１４０８は、それだけには限定されないが、システムメモリ１４０６を含むシステムコンポーネントを処理ユニット１４０４に結合する。処理ユニット１４０４は、市販の様々なプロセッサのうちのどんなものでもよい。デュアルマイクロプロセッサおよび他のマルチプロセッサアーキテクチャを処理ユニット１４０４として使用することもできる。

【００９１】

システムバス１４０８は、市販の様々なバスアーキテクチャのうちの任意のものを使用するメモリバス（メモリコントローラ付きまたはメモリコントローラ無し）、周辺バス、およびローカルバスとさらに相互接続し得るいくつかのタイプのバス構造のうちどんなものでもよい。システムメモリ１４０６は、読み取り専用メモリ（ＲＯＭ）１４１０およびランダムアクセスメモリ（ＲＡＭ）１４１２を含む。基本入出力システム（ＢＩＯＳ）は、ＲＯＭ、ＥＰＲＯＭ、ＥＥＰＲＯＭなどの不揮発性メモリ１４１０に格納されており、例えば起動中など、コンピュータ１４０２内の要素間での情報の転送を助ける基本ルーチンを含む。ＲＡＭ１４１２は、例えばデータをキャッシュに入れるための静的ＲＡＭなど高速ＲＡＭを含むこともできる。

【００９２】

コンピュータ１４０２は、適応の筐体（図示せず）により外部使用向けに設定することもできる内部ハードディスクドライブ（ＨＤＤ）１４１４（ＥＩＤＥ、ＳＡＴＡなど）、磁気フロッピー（登録商標）ディスクドライブ（ＦＤＤ）１４１６（例えば取外式ディスケット１４１８から読み取り、またはそこに書き込む）、および光ディスクドライブ１４２０（例えばＣＤ－ＲＯＭディスク１４２２を読み込む、またはＤＶＤなど他の大容量光媒体から読み取り、そこに書き込む）をさらに含む。ハードディスクドライブ１４１４、磁気ディスクドライブ１４１６、および光ディスクドライブ１４２０は、それぞれハードディスクドライブインターフェース１４２４、磁気ディスクドライブインターフェース１４２６、および光ディスクドライブインターフェース１４２８によってシステムバス１４０８に接続することができる。外部ドライブ実装用のインターフェース１４２４は、ユニバーサルシリアルバス（ＵＳＢ）およびＩＥＥＥ１３９４インターフェース技術のうちの少なくとも一方または両方を含む。他の外部ドライブ接続技術も本発明の意図内に含まれる。

【００９３】

ドライブ（駆動装置）およびその関連のコンピュータ可読媒体は、データ、データ構造、コンピュータ実行可能命令などの不揮発性の記憶域を提供する。コンピュータ１４０２では、ドライブおよび媒体は、任意のデータの記憶を適応のデジタル形式で収納する。上記のコンピュータ可読媒体の説明は、ＨＤＤ、取外式磁気ディスケット、およびＣＤやＤＶＤなどの取外式光媒体を指すが、ｚｉｐドライブ、磁気カセット、フラッシュメモリカード、カートリッジなど、コンピュータによって読み取り可能な他のタイプの媒体を動作環境例で使用してもよく、さらに、このような任意の媒体は、本発明の方法を実行するコンピュータ実行可能命令を含むことができることを当業者であれば理解されたい。

【００９４】

オペレーティングシステム１４３０、１つまたは複数のアプリケーションプログラム１４３２、他のプログラムモジュール１４３４およびプログラムデータ１４３６を含めて、いくつかのプログラムモジュールをドライブおよびＲＡＭ１４１２に格納することができる。オペレーティングシステム、アプリケーション、モジュール、および／またはデータのすべてまたは一部をＲＡＭ１４１２にキャッシュすることもできる。本発明は、市販の様々なオペレーティングシステムまたはオペレーティングシステムの組合せで実施できることを理解されたい。

【００９５】

ユーザは、コマンドおよび情報を、キーボード１４３８およびマウス１４４０などのポインティング装置など、１つまたは複数の有線／無線の入力装置を介してコンピュータ１

４０２に入力することができる。他の入力装置（図示せず）には、マイクロフォン、ＩＲリモートコントロール、ジョイスティック、ゲームパッド、スタイラスペン、タッチ画面などがある。これらおよび他の入力装置は、システムバス１４０８に結合されている入力装置インターフェース１４４２を介して処理ユニット１４０４に接続されることが多いが、パラレルポート、ＩＥＥＥ１３９４シリアルポート、ゲームポート、ＵＳＢポート、ＩＲインターフェースなど他のインターフェースで接続することができる。

【００９６】

モニタ１４４４または他のタイプの表示装置も、ビデオアダプタ１４４６などのインターフェースを介してシステムバス１４０８に接続される。コンピュータは一般に、モニタ１４４４に加えて、スピーカやプリンタなど他の周辺出力装置（図示せず）を含んでいる。

【００９７】

コンピュータ１４０２は、リモートコンピュータ１４４８など１つまたは複数のリモートコンピュータへの有線および／または無線の通信を介した論理接続を使用してネットワーク式環境で動作することができる。リモートコンピュータ１４４８は、ワークステーション、サーバコンピュータ、ルータ、パーソナルコンピュータ、携帯用コンピュータ、マイクロプロセッサベースの娯楽装置、ピア装置、または他の一般のネットワークノードなどとすることができ、一般にコンピュータ１４０２に関連して記載した多くまたはすべての要素を含むが、簡潔にするために、メモリ／記憶装置１４５０のみを示している。示した論理接続は、ローカルエリアネットワーク（ＬＡＮ）１４５２、および／または広域ネットワーク（ＷＡＮ）１４５４などのより大きいネットワークへの有線／無線の接続を含む。このようなＬＡＮおよびＷＡＮネットワーキング環境は、オフィスや会社ではごく一般的であり、イントラネットなど全社規模のコンピュータネットワークを容易にし、これらはすべて、インターネットなどのグローバルな通信ネットワークと接続することができる。

【００９８】

ＬＡＮネットワーキング環境で使用する場合、コンピュータ１４０２は、有線および／または無線の通信ネットワークインタフェースまたはアダプタ１４５６を介してローカルネットワーク１４５２に接続される。アダプタ１４５６は、無線アダプタ１４５６と通信するために配置されている無線アクセスポイントも含み得るＬＡＮ１４５２との有線または無線の通信を円滑にすることができる。

【００９９】

ＷＡＮネットワーキング環境で使用する場合、コンピュータ１４０２は、モデム１４５８を含むことができ、ＷＡＮ１４５４を介して通信サーバに接続され、またはインターネットによるなどＷＡＮ１４５４を介して通信を確立する他の手段を有する。モデム１４５８は、内蔵または外付け、および有線または無線の装置とすることができ、シリアルポートインタフェース１４４２を介してシステムバス１４０８に接続される。ネットワーク式環境では、コンピュータ１４０２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メモリ／記憶装置１４５０に格納することができる。図示したネットワーク接続は例であり、コンピュータ間の通信リンクを確立する他の手段を使用することができることは理解されよう。

【０１００】

コンピュータ１４０２は、プリンタ、スキャナ、デスクトップ、および／または携帯用コンピュータ、ＰＤＡ、通信衛星、無線で検出できるタグに関連付けられている任意の設備または場所（キオスク、新聞雑誌売り場、休憩室など）、および電話など、無線通信で動作可能に配置されている任意の無線装置またはエンティティと通信するよう動作可能である。これらは、少なくともＷｉ－ＦｉおよびＢｌｕｅｔｏｏｔｈ（商標）無線技術を含む。したがって、通信は、従来のネットワークと同様に予め定義されている構造、または単に少なくとも２つの装置間の随時の通信とすることができる。

【０１０１】

Ｗｉ－Ｆｉ(ワイ・ファイ)またはＷｉｒｅｌｅｓｓ　Ｆｉｄｅｌｉｔｙによって、家のソファー、ホテルの部屋のベッド、または職場の会議室から、配線無しにインターネットへの接続が可能になる。Ｗｉ－Ｆｉは、コンピュータなどのこのような装置がデータを屋内および屋外の基地局の範囲内のどこででも送受信できるようにするセル方式電話で使用されているものと似た無線技術である。Ｗｉ－Ｆｉネットワークは、ＩＥＥＥ８０２．１１（ａ、ｂ、ｇなど）と呼ばれる無線技術を使用して、安全で信頼性が高い高速無線接続を提供する。Ｗｉ－Ｆｉネットワークは、コンピュータを互いに、インターネットに、および有線のネットワーク（ＩＥＥＥ８０２．３またはＥｔｈｅｒｎｅｔ（登録商標）を使用する）に接続するために使用することができる。Ｗｉ－Ｆｉネットワークは、ライセンス不要の２．４ＧＨｚおよび５ＧＨｚの無線帯域で、例えば１１Ｍｂｐｓ（８０２．１１ａ）または５４Ｍｂｐｓ（８０２．１１ｂ）データ転送速度で、または両方の帯域（デュアルバンド）を含む製品で動作するため、ネットワークは、多くのオフィスで使用されている基本的な１０ＢａｓｅＴ配線のＥｔｈｅｒｎｅｔ（登録商標）ネットワークと似た実環境での性能を提供することができる。

【０１０２】

次に図１５を参照すると、本発明によるコンピューティング環境１５００の例の概略ブロック図を示している。システム１５００は、１つまたは複数のクライアント１５０２を含む。クライアント１５０２は、ハードウェアおよび／またはソフトウェア（スレッド、プロセス、コンピューティングデバイスなど）とすることができる。クライアント１５０２は、例えば本発明を使用することによって、クッキー（ｃｏｏｋｉｅ）および／または関連の文脈情報を収納することができる。

【０１０３】

システム１５００は、１つまたは複数のサーバ１５０４も含む。サーバ１５０４もハードウェアおよび／またはソフトウェア（スレッド、プロセス、コンピューティングデバイスなど）とすることができる。サーバ１５０４は、例えば本発明を使用することによって変換を行うためにスレッド（メッセージ群）を収納することができる。クライアント１５０２とサーバ１５０４との間の考え得る１つの通信は、２つ以上のコンピュータプロセス間で送信されるように構成されたデータパケットの形式とすることができる。データパケットは、例えばクッキーおよび／関連の文脈情報を含み得る。システム１５００は、クライアント１５０２とサーバ１５０４との間の通信を円滑にするために使用できる通信フレームワーク１５０６（インターネットなどのグローバル通信ネットワークなど）を含む。

【０１０４】

通信は、有線（光ファイバを含む）および／または無線の技術を介して促進することができる。クライアント１５０２は、クライアント１５０２にローカルな情報（クッキーおよび／または関連の文脈情報など）を格納するために使用できる１つまたは複数のクライアントデータストア１５０８に動作可能に接続される。同様に、サーバ１５０４は、サーバ１５１４にローカルな情報を格納するために使用できる１つまたは複数のサーバデータストア１５１０に動作可能に接続される。

【０１０５】

上記で説明してきたことは、開示された本発明の例示を含む。当然、コンポーネントおよび／または方法の考え得るすべての組合せを記述することはできないが、これ以上の多くの組合せおよび置き換えが可能であることを当業者は理解されよう。したがって、本発明は、特許請求の範囲の意図および範囲内のこのようなすべての代替形態、修正形態、および変形形態を含むものとする。さらに、「含む」という用語が詳細な説明または特許請求の範囲で使用されている限り、このような用語は、請求項で移行語として使用されるときに「含む」が解釈される「含む」という用語と同じように包含的であるものとする。

【図面の簡単な説明】

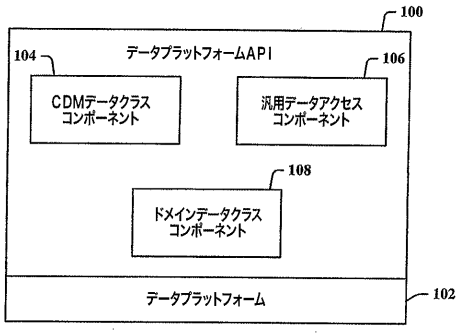
【０１０６】

【図１】本発明の一態様によるデータプラットフォームのストレージアプリケーションプログラムインターフェース（ＡＰＩ）を示す図である。

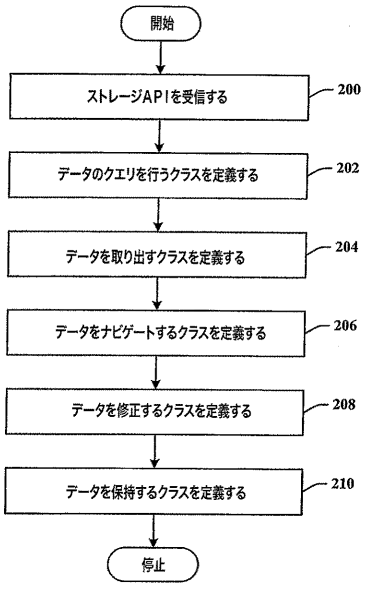
【図２】開示された一態様によるストレージＡＰＩを提供する方法を示す図である。
【図３】ストレージＡＰＩのジェネリックデータアクセスコンポーネントを示すより詳細な図である。
【図４】データモデルのストレージＡＰＩを提供する方法を示す図である。
【図５】テーブルセットタイプを提示する方法を示す図である。
【図６】ＡＰＩにおいてＷｉｎＦＳ機能を提供する方法を示す図である。
【図７】ストア内のクラスを表す方法を示す図である。
【図８】クライアントと１つまたは複数のストアとの間の接続をカプセル化する方法を示す図である。
【図９】ストアに対するクエリを構築する方法を示す図である。
【図１０】１組の結果にわたる表示の方法を示す図である。
【図１１】結果にわたるデータの初期ビューを提示する方法を示す図である。
【図１２】ストレージレコードクラスを拡張する方法を示す図である。
【図１３】ストレージＡＰＩを共通データプラットフォームに使用するシステムを示す図である。
【図１４】開示されたアーキテクチャを実行するように動作可能なコンピュータを示すブロック図である。
【図１５】コンピューティング環境の例を示す概略ブロック図である。
【符号の説明】
【０１０７】
１００　データプラットフォームＡＰＩ
１０２　データプラットフォーム
１０４　ＣＤＭデータクラスコンポーネント
１０６　ジェネリックデータアクセスコンポーネント
１０８　ドメインデータクラスコンポーネント
１３０２　共通データプラットフォーム（ＣＤＰ）
１３０４　アプリケーションおよびアプリケーションフレームワーク
１３０６　データストア
１３０８　ランタイム
１３１０　制約／セキュリティ
１４０４　処理ユニット
１４０６　システムメモリ
１４０８　バス
１４１４　内蔵ＨＤＤ
１４１４　外部ＨＤＤ
１４１８　ディスク
１４２０　光ドライブ
１４２２　ディスク
１４２４　インターフェース
１４２６　インターフェース
１４２８　インターフェース
１４３０　オペレーティングシステム
１４３２　アプリケーション
１４３４　モジュール
１４３６　データ
１４３８　キーボード
１４４０　マウス
１４４２　入力装置インターフェース
１４４４　モニタ
１４４６　ビデオアダプタ

１４４８　リモートコンピュータ
１４５０　メモリ／ストレージ
１４５６　ネットワークアダプタ
１４５８　モデム
１５０２　クライアント
１５０４　サーバ
１５０６　通信フレームワーク
１５０８　クライアントデータストア
１５１０　サーバデータストア

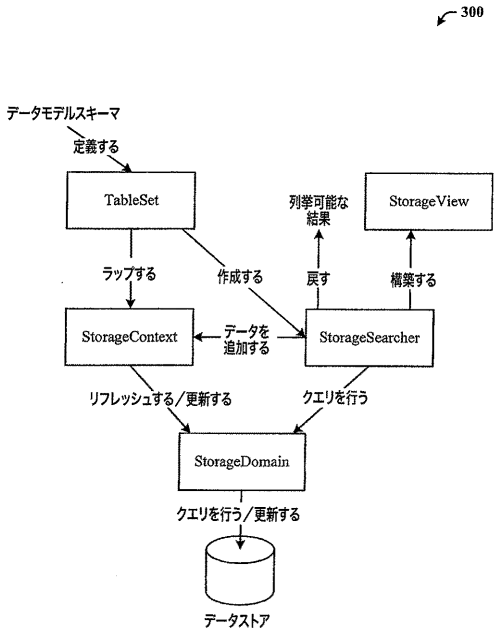
【図１】



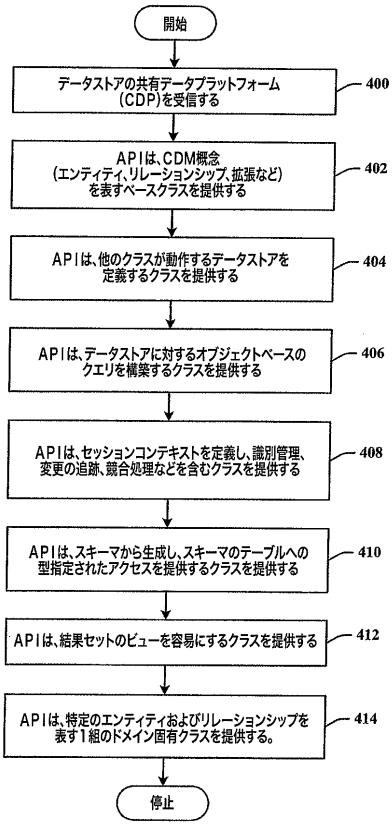
【図２】



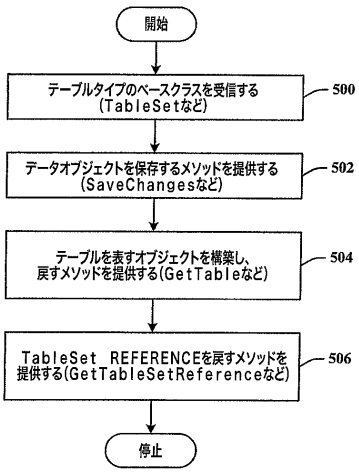
【図３】


300
データモデルスキーマ
定義する
TableSet
列挙可能な結果
StorageView
ラップする
作成する
戻す
構築する
StorageContext
データを追加する
StorageSearcher
リフレッシュする／更新する
クエリを行う
StorageDomain
クエリを行う／更新する
データストア

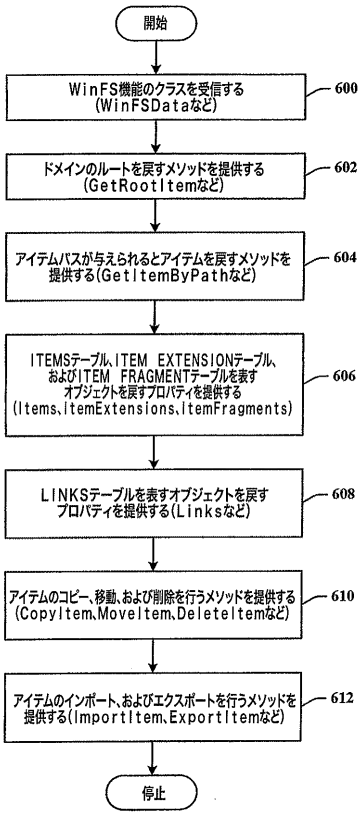
【図４】


開始
データストアの共有データプラットフォーム(CDP)を受信する
400
APIは、CDM概念(エンティティ、リレーションシップ、拡張など)を表すベースクラスを提供する
402
APIは、他のクラスが動作するデータストアを定義するクラスを提供する
404
APIは、データストアに対するオブジェクトベースのクエリを構築するクラスを提供する
406
APIは、セッションコンテキストを定義し、識別管理、変更の追跡、競合処理などを含むクラスを提供する
408
APIは、スキーマから生成し、スキーマのテーブルへの型指定されたアクセスを提供するクラスを提供する
410
APIは、結果セットのビューを容易にするクラスを提供する
412
APIは、特定のエンティティおよびリレーションシップを表す1組のドメイン固有クラスを提供する。
414
停止

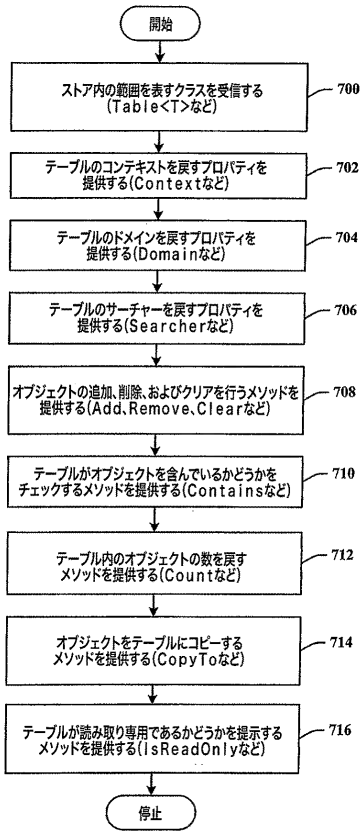
【図５】


開始
テーブルタイプのベースクラスを受信する(TableSetなど)
500
データオブジェクトを保存するメソッドを提供する(SaveChangesなど)
502
テーブルを表すオブジェクトを構築し、戻すメソッドを提供する(GetTableなど)
504
TableSet REFERENCEを戻すメソッドを提供する(GetTableSetReferenceなど)
506
停止

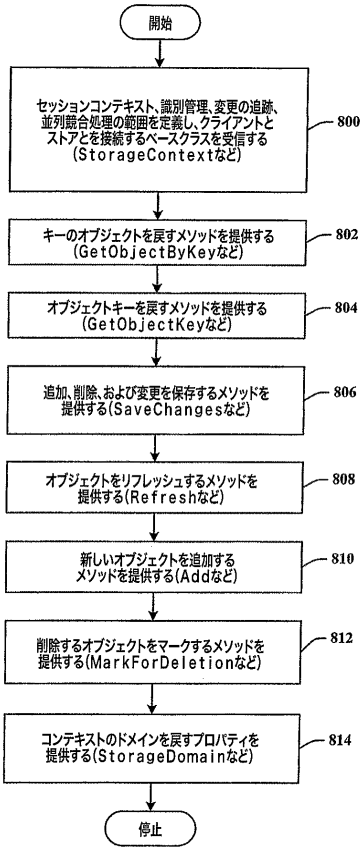
【図６】


開始
WinFS機能のクラスを受信する(WinFSDataなど)
600
ドメインのルートを戻すメソッドを提供する(GetRootItemなど)
602
アイテムパスが与えられるとアイテムを戻すメソッドを提供する(GetItemByPathなど)
604
ITEMSテーブル、ITEM EXTENSIONテーブル、およびITEM FRAGMENTテーブルを表すオブジェクトを戻すプロパティを提供する(Items、ItemExtensions、ItemFragments)
606
LINKSテーブルを表すオブジェクトを戻すプロパティを提供する(Linksなど)
608
アイテムのコピー、移動、および削除を行うメソッドを提供する(CopyItem、MoveItem、DeleteItemなど)
610
アイテムのインポート、およびエクスポートを行うメソッドを提供する(ImportItem、ExportItemなど)
612
停止

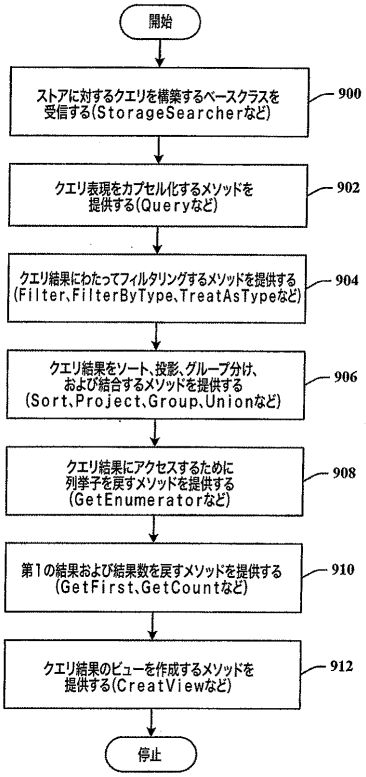
【図７】


開始
ストア内の範囲を表すクラスを受信する (Table<T>など)
700
テーブルのコンテキストを戻すプロパティを提供する(Contextなど)
702
テーブルのドメインを戻すプロパティを提供する(Domainなど)
704
テーブルのサーチャーを戻すプロパティを提供する(Searcherなど)
706
オブジェクトの追加、削除、およびクリアを行うメソッドを提供する(Add、Remove、Clearなど)
708
テーブルがオブジェクトを含んでいるかどうかをチェックするメソッドを提供する(Containsなど)
710
テーブル内のオブジェクトの数を戻すメソッドを提供する(Countなど)
712
オブジェクトをテーブルにコピーするメソッドを提供する(CopyToなど)
714
テーブルが読み取り専用であるかどうかを提示するメソッドを提供する(IsReadOnlyなど)
716
停止

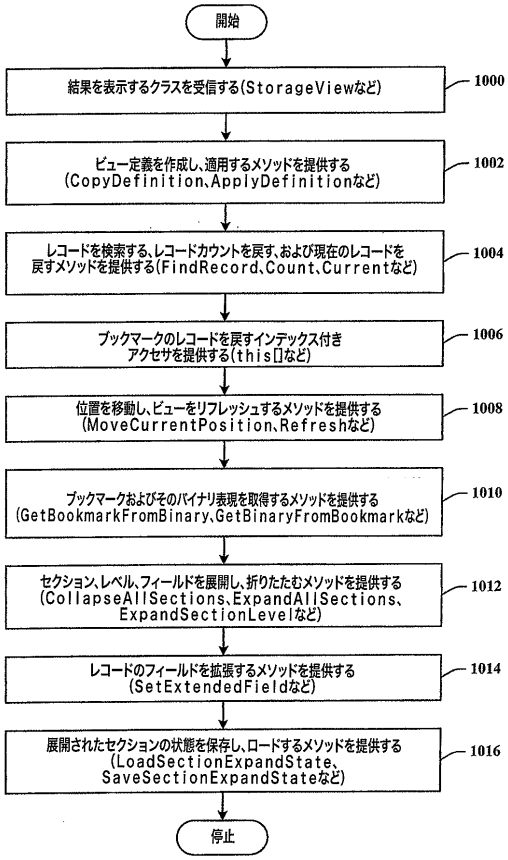
【図８】


開始
セッションコンテキスト、識別管理、変更の追跡、並列競合処理の範囲を定義し、クライアントとストアとを接続するベースクラスを受信する (StorageContextなど)
800
キーのオブジェクトを戻すメソッドを提供する (GetObjectByKeyなど)
802
オブジェクトキーを戻すメソッドを提供する (GetObjectKeyなど)
804
追加、削除、および変更を保存するメソッドを提供する(SaveChangesなど)
806
オブジェクトをリフレッシュするメソッドを提供する(Refreshなど)
808
新しいオブジェクトを追加するメソッドを提供する(Addなど)
810
削除するオブジェクトをマークするメソッドを提供する(MarkForDeletionなど)
812
コンテキストのドメインを戻すプロパティを提供する(StorageDomainなど)
814
停止

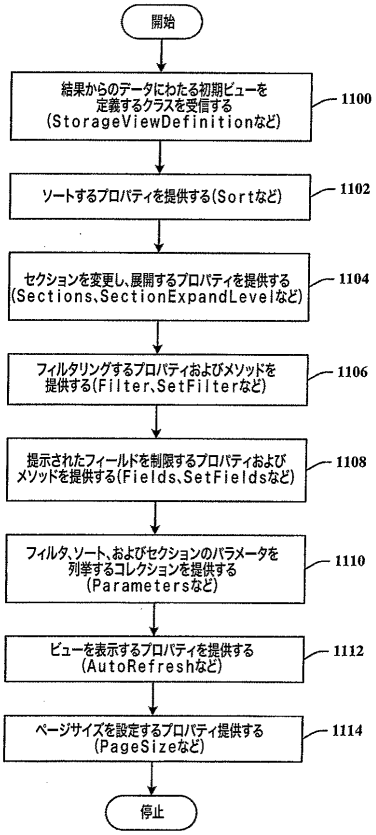
【図９】


開始
ストアに対するクエリを構築するベースクラスを受信する(StorageSearcherなど)
900
クエリ表現をカプセル化するメソッドを提供する(Queryなど)
902
クエリ結果にわたってフィルタリングするメソッドを提供する (Filter、FilterByType、TreatAsTypeなど)
904
クエリ結果をソート、投影、グループ分け、および結合するメソッドを提供する (Sort、Project、Group、Unionなど)
906
クエリ結果にアクセスするために列挙子を戻すメソッドを提供する (GetEnumeratorなど)
908
第1の結果および結果数を戻すメソッドを提供する (GetFirst、GetCountなど)
910
クエリ結果のビューを作成するメソッドを提供する(CreatViewなど)
912
停止

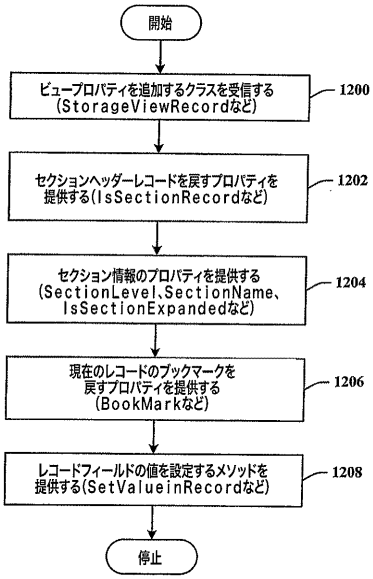
【図１０】


開始
結果を表示するクラスを受信する(StorageViewなど)
1000
ビュー定義を作成し、適用するメソッドを提供する (CopyDefinition、ApplyDefinitionなど)
1002
レコードを検索する、レコードカウントを戻す、および現在のレコードを戻すメソッドを提供する(FindRecord、Count、Currentなど)
1004
ブックマークのレコードを戻すインデックス付きアクセサを提供する(this[]など)
1006
位置を移動し、ビューをリフレッシュするメソッドを提供する (MoveCurrentPosition、Refreshなど)
1008
ブックマークおよびそのバイナリ表現を取得するメソッドを提供する (GetBookmarkFromBinary、GetBinaryFromBookmarkなど)
1010
セクション、レベル、フィールドを展開し、折りたたむメソッドを提供する (CollapseAllSections、ExpandAllSections、ExpandSectionLevelなど)
1012
レコードのフィールドを拡張するメソッドを提供する (SetExtendedFieldなど)
1014
展開されたセクションの状態を保存し、ロードするメソッドを提供する (LoadSectionExpandState、SaveSectionExpandStateなど)
1016
停止

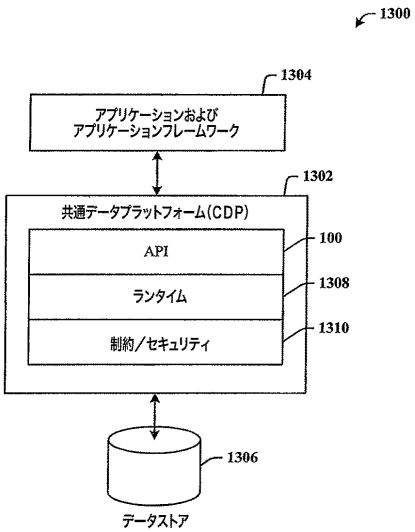
【図１１】



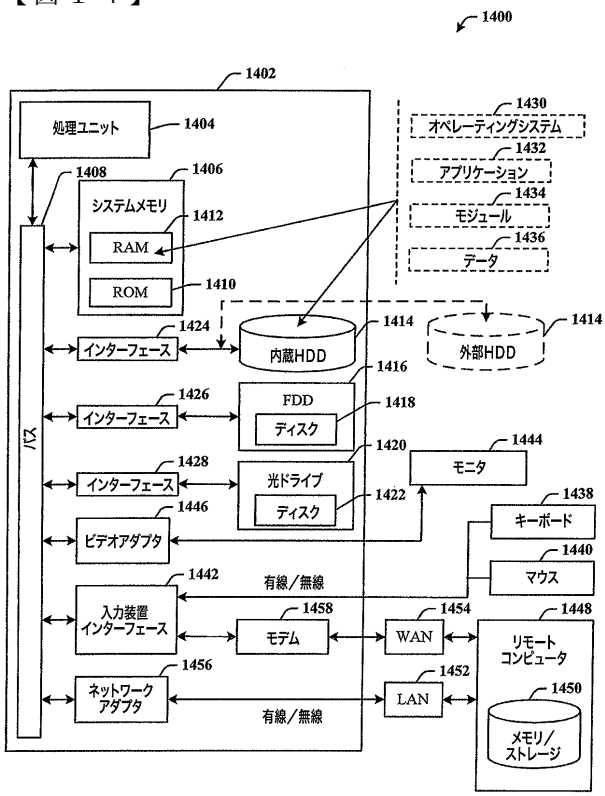
【図１２】



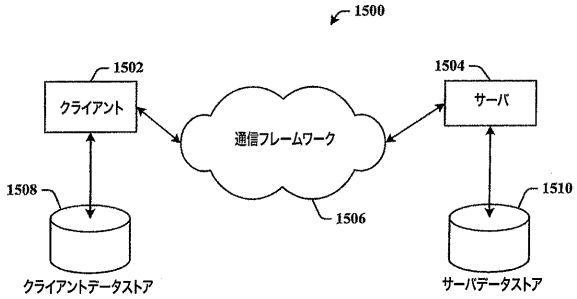
【図１３】



【図１４】



【図１５】


1500
1502
クライアント
1504
サーバ
通信フレームワーク
1506
1508
クライアントデータストア
1510
サーバデータストア

フロントページの続き

(72)発明者　ベンジャミン　アルバーリ
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内
(72)発明者　エドワード　ジー．シェパード
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内
(72)発明者　マイケル　イー．ディーム
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内
(72)発明者　マイケル　ジェイ．ピゾ
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内
(72)発明者　ラメシュ　ナガラジャン
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　ワン　マイクロソフト　ウェイ　マイクロソフト　コーポレーション内

Ｆターム(参考) 5B065 BA01　CA15　CA16
5B082 GA08