

(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2006-244493

(P2006-244493A)

(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	510A		5B065
G06F 3/06 (2006.01)		G06F 3/06	301G		5B082

審査請求 未請求 請求項の数 20 O L (全 34 頁)

(21) 出願番号 特願2006-52065 (P2006-52065)
 (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, 533
 (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 11/155, 428
 (32) 優先日 平成17年6月17日 (2005.6.17)
 (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 9805
 2-6399 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 ジャック エス. リッチンス
 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

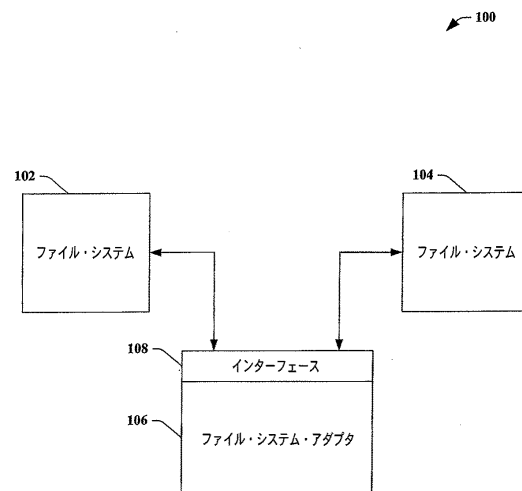
(54) 【発明の名称】 データベース内で表現されるファイル・システム

(57) 【要約】

【課題】 少なくとも2つの異なるファイル・システム同士を容易に適合させるシステムおよび／または方法を提供すること。

【解決手段】 ファイル・システム・アダプタは、第1のファイル・システムと第2のファイル・システムの間で統一性および／またはシームレスな統合を与えることが可能である。ファイル・システム・アダプタは、少なくとも1つのプロパティをアイテムに与えるファイルと、アイテムに関連付けられたファイル・ストリームとによって質的に強化されたファイル・バック・アイテムを用いることが可能である。ファイル・バック・アイテムは、第1のファイル・システム用のAPIと第2のファイル・システム用のAPIとのうちの少なくとも1つによる操作を可能にすることができる。

【選択図】 図1



【特許請求の範囲】

【請求項 1】

異なるファイル・システム間の統一性を実現することを容易にするシステムであって、第 1 のファイル・システムおよび第 2 のファイル・システムからデータを受け取るインターフェースと、

少なくとも 1 つのプロパティをアイテムに与えるファイル、および前記アイテムに関連付けられたファイル・ストリームによって質的に強化されたファイル・バック・アイテムを用いて、前記第 1 のファイル・システム用の A P I および前記第 2 のファイル・システム用の A P I のうちの少なくとも一方による操作を可能にすることによって、前記第 1 のファイル・システムと前記第 2 のファイル・システムとの間で統一性を実現するファイル・システム・アダプタと
を備えることを特徴とするシステム。 10

【請求項 2】

前記第 1 のファイル・システムは、ファイルおよびディレクトリを関連のファイル・ストリームとともに利用して、アプリケーション用の空間に、より大きな構造を与える、16 ビット・ファイル・システムと 32 ビット・ファイル・システムとのうちの少なくとも 1 つであることを特徴とする請求項 1 に記載のシステム。

【請求項 3】

前記第 2 のファイル・システムは、情報を複合型として表すために、アイテム、サブアイテム、プロパティ、およびリレーションシップのうちの少なくとも 1 つを定義するファイル・ストレージ・システムであることを特徴とする請求項 1 に記載のシステム。 20

【請求項 4】

前記第 1 のファイル・システムを解析して、ディレクトリ、ファイル、ファイル・ストリーム、属性、ノード、リーフ・ノード、およびセキュリティ特性のうちの少なくとも 1 つを決定するアナライザ・コンポーネントをさらに備えることを特徴とする請求項 1 に記載のシステム。

【請求項 5】

前記第 1 のファイル・システムのリーフ・ノードの前記ファイル・バック・アイテムと、前記第 1 のファイル・システムのディレクトリ内の非リーフ・ノードの前記アイテムのうちの少なくとも 1 つを生成するジェネレータ・コンポーネントをさらに備えることを特徴とする請求項 1 に記載のシステム。 30

【請求項 6】

前記第 1 のファイル・システム内の前記ファイルの属性を、前記第 2 のファイル・システムによって利用される前記ファイル・バック・アイテムに関連付ける属性コンポーネントをさらに備えることを特徴とする請求項 1 に記載のシステム。

【請求項 7】

前記属性は、読み出し専用、圧縮済みファイル、アーカイブ・ファイルのうちの少なくとも 1 つを表すことが可能なビットであることを特徴とする請求項 6 に記載のシステム。

【請求項 8】

前記ファイル・バック・アイテムと、前記第 1 のファイル・システム内で行われるファイル・バックアップとの間の同期化を可能にするメタデータ・ハンドラ・コンポーネントをさらに備えることを特徴とする請求項 1 に記載のシステム。 40

【請求項 9】

前記メタデータ・ハンドラは、
バックアップ・ファイル・ストリームにおいてプロパティが修正された際に、新しい値を前記第 2 のファイル・システムにおいて前記ファイル・バック・アイテム内の適切なフィールドにプロモートすること、および、

前記第 2 のファイル・システム用の A P I を介してプロパティが変更されたときに新しい値を前記バックアップ・ファイル・ストリームにデモートすること

のうちの少なくとも 1 つを可能にすることを特徴とする請求項 8 に記載のシステム。 50

【請求項 10】

別のファイル・バック・アイテムの複製であるがそのファイル・ストリームを含まないゴースト化アイテム、

前記ゴースト化アイテムを生成するゴースト化操作、および、

前記ゴースト化アイテムを通常アイテムに変換するゴースト化解除操作

のうちの少なくとも1つを提供するゴースト・コンポーネントをさらに備えることを特徴とする請求項1に記載のシステム。

【請求項 11】

前記少なくとも1つのファイル・バック・アイテムの同期化を、少なくとも1つのファイル・バック・アイテムに関連付けられたファイル・ストリームのスワップを可能にすることによって提供する属性トンネリング・コンポーネントをさらに備えることを特徴とする請求項1に記載のシステム。

10

【請求項 12】

前記ファイル・ストリームはファイル・ストリーム・テーブル内にあり、そこで、前記アイテムIDはスワップされることが可能であることを特徴とする請求項11に記載のシステム。

【請求項 13】

前記ファイル・バック・アイテムは、セマンティック・レイヤ、ファイル・ストリーム、ファイル属性、エンティティ状態、変更単位、およびアイテム・データのうちの少なくとも1つを収容することを特徴とする請求項1に記載のシステム。

20

【請求項 14】

周期を有しない、接続されたアイテムのグラフを与える有向非周期グラフ(DAG)コンポーネントをさらに備えることを特徴とする請求項1に記載のシステム。

【請求項 15】

請求項1に記載のシステムの各コンポーネント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 16】

異なるファイル・システム間の統一性を実現することを容易にする、コンピュータ実装方法であって、

ファイルおよびディレクトリを関連のファイル・ストリームとともに利用する第1のファイル・システムを解析して、アプリケーション用の空間に、より大きな構造を与えること、

30

少なくとも1つのプロパティをアイテムに与えるファイルと、前記アイテムに関連付けられたファイル・ストリームと、によって質的に強化されたファイル・バック・アイテムを生成すること、および、

前記ファイル・バック・アイテムを利用して、前記第1のファイル・システムと第2のファイル・システムとの間で統一性を実現すること

を備えることを特徴とする方法。

【請求項 17】

前記第1のファイル・システム内の非リーフ・ノードに対してアイテムを生成すること

40

、前記第1のファイル・システム内のリーフ・ノードに対してファイル・バック・アイテムを生成すること、

前記ファイルからの属性を前記ファイル・バック・アイテムに組み込むこと、および、

前記第1のファイル・システム用のAPIと前記第2のファイル・システム用のAPIとのうちの少なくとも1つによる操作を可能にすること

のうちの少なくとも1つをさらに備えることを特徴とする請求項16に記載の方法。

【請求項 18】

ゴースト化およびゴースト化解除のうちの少なくとも1つを行うこと、

属性トンネリングを利用すること、および、

50

DAGを利用すること

のうちの少なくとも1つをさらに備えることを特徴とする請求項16に記載の方法。

【請求項19】

ファイル・システム・アダプタとインターフェースとの間で伝達されるデータ・パケットであって、請求項16に記載の方法を容易にすることを特徴とするデータ・パケット。

【請求項20】

異なるファイル・システム間の統一性を実現することを容易にする、コンピュータ実装システムであって、

第1のファイル・システムおよび第2のファイル・システムからデータを受け取る手段と、

少なくとも1つのプロパティをアイテムに与えるファイル、および前記アイテムに関連付けられたファイル・ストリームによって質的に強化されたファイル・バック・アイテムを用いて、前記第1のファイル・システム用のAPIおよび前記第2のファイル・システム用のAPIのうちの少なくとも一方による操作を可能にすることによって、前記第1のファイル・システムと前記第2のファイル・システムとの間で統一性を実現する手段とを備えることを特徴とするシステム。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、異なるファイル・システムを容易に適合させるシステムおよび／または方法に関する。

【背景技術】

【0002】

コンピュータ・テクノロジー（たとえば、マイクロ・プロセッサ速度、メモリ容量、データ転送帯域幅、ソフトウェア機能など）の進歩は、全般的に、様々な産業におけるコンピュータ応用分野の増加に寄与している。より一層強力なサーバ・システムは、複数のサーバからなるアレイとして構成されることが多く、たとえば、ワールド・ワイド・ウェブなどの外部ソースから発せられるサービス・リクエストするために、一般的に提供されている。

【0003】

使用可能な電子データの量が増えるにつれて、そのようなデータを、ユーザ・フレンドリかつ迅速なデータの検索および取り出しを容易にする、管理可能な形で記憶することの重要性が増している。今日の一般的な手法は、電子データを1つまたは複数のデータベースに格納することである。一般に、典型的なデータベースは、たとえばコンピュータ・プログラムが所望のデータ区画を素早く検索および選択できるように構成されたデータを有する、整理された、情報の集合体であると言えることができる。一般に、データベース内のデータは、1つまたは複数のテーブルによって編成される。そのようなテーブルは、行（row）および列（column）の配列として整理されている。

【0004】

また、これらのテーブルはレコードのセットを含むことが可能であり、レコードはフィールドのセットを含む。一般にレコードは、テーブル内の行（row）としてインデックスが付けられ、レコード・フィールドは通常、列（column）としてインデックスが付けられる。このインデックスの行／列のペアによって、テーブル内にある特定のデータを参照することが可能になる。たとえば、行は、販売取引、人物、プロジェクトなどに関連する完全なデータレコードを格納することが可能である。同様に、テーブルの列（column）は、同じ一般データ形式を有する行の個別部分を定義することが可能であり、列は、レコードのフィールドを定義することが可能である。

【0005】

データの個々の部分それぞれは、独立しており、概してあまりインフォーマティブ（informative：特別に伝えることに意味があること）ではない。データベース・アプリケーション

10

20

30

40

50

ョンは、データをより有用なものにし、それにより、ユーザによるデータの整理および処理を支援する。データベース・アプリケーションは、ユーザが、データを比較し、並べ替え、順序づけ、結合し、分離し、連係などを行うことを可能にするので、その結果、データから有用な情報を生成することが可能となる。

【発明の開示】

【発明が解決しようとする課題】

【0006】

しかしながら、データベースの容量および汎用性は、驚くべき量にまで膨れ上り、事実上無限の記憶容量をデータベースに使用することを可能にしている。さらに、一般的なデータベース・システムのクエリ機能は、時刻、ファイル・エクステンション、ロケーション、およびサイズによるものに制限されている。たとえば、データベースに関連付けられた膨大な量のデータを検索する場合の一般的な検索が、ファイル名、ファイル・サイズ、データ作成日時に制限されている。このような技法は不完全であり、不適切である。

10

【0007】

エンド・ユーザからのデータが引き続き存在し、その作成が増え続けるのに伴って、そのようなデータを検索、関連付け、および記憶することに関する問題や困難さはピークに達しつつある。エンド・ユーザは、文書を作成したり、写真を記憶したり、コンパクト・ディスクから音楽をリッピングしたり、電子メールを受信したり、送信済み電子メールのコピーを残したりする。たとえば、音楽のコンパクト・ディスクを作成する単純なプロセスでも、エンド・ユーザは、メガ・バイト規模のデータを作成することが可能である。その音楽をコンパクト・ディスクからリッピングすること、そのファイルを適切なフォーマットに変換すること、ケース・カバーを作成すること、コンパクト・ディスクのラベルをデザインすることなど、すべてデータの作成が必要である。

20

【0008】

ユーザ側に関わる複雑さだけでなく、開発者の側にもデータについての同様の問題がある。開発者は、個人用アプリケーションから開発が高度な企業向けアプリケーションに至る無数のアプリケーションを創出し記述する。アプリケーションの創出および／または開発中に、開発者は、常時でないにしても頻繁に、データを（一個所に）収集する。そのようなデータを取得する間、それらのデータを記憶しなければならない。言い換えると、データの検索、関連付け、および記憶に関わる問題や困難さは、開発者とエンド・ユーザの両方を危うくする。

30

【0009】

さらに、データベースの技術的進歩は、多数のフォーマットおよび／または構成を生み出した。データベースのフォーマットおよび／またはコンフィギュレーションは、当然のごとく増大した。すなわち、そのような変化により、統合の困難性を引き起こす可能性がある。言い換えると、開発上の違いに基づいて、異なるファイル・データベースは互いによく適合しないのが普通である。そこで、従来の複数のシステムおよび複数データベースに関連する上述の例示的問題を克服する必要性が存在する。

【課題を解決するための手段】

【0010】

以下では、本発明のいくつかの態様の基本的理解が得られるように、本発明の要約を示す。この要約は、本発明の包括的概要ではない。この要約は、本発明の重要な、またはクリティカルな要素を特定することや、本発明の範囲を線引きすることを意図するものではない。この要約は、後述の、本発明のより詳細な説明への導入として、本発明のいくつかの概念を簡単な形態で示すことのみを目的とする。

40

【0011】

本発明は、複数の異なるファイル・システム (disparate file systems) を容易に適合させるシステムおよび／または方法に関する。ファイル・システム・アダプタが、第1のファイル・システムと第2のファイル・システムの間の統一性 (uniformity) の提供を可能にする。この統一性は、第1のファイル・システム用のアプリケーションおよび／また

50

はA P Iが第2のファイル・システムに実装されることを可能にすることができる。このファイル・システム・アダプタは、ファイル・バック・アイテム (file-backed item) を実装することが可能であり、このファイル・バック・アイテムは、第2のファイル・システムに関連するアイテムであり、それに関連付けられた少なくとも1つのプロパティを取り込むファイルおよび／またはファイル・ストリームに依拠する第2のファイル・システムに関連するアイテム、とすることが可能である。ファイル・バック・アイテムは、そのファイルおよび／または関連付けられたファイル・ストリームを利用して、そのアイテムに、第2のファイル・システムにあるプロパティおよび／またはメタデータを取り込むことが可能である。このファイル・システム・アダプタは、ファイル・バック・アイテムを利用して、第1のファイル・システムの現在および以前のバージョンに関連する既存のA P Iによる操作だけでなく、第2のファイル・システムに関連付けられたA P Iによる操作も可能にする。このファイル・システム・アダプタはまた、第1のファイル・システムと第2のファイル・システムとの間に同期化を提供することが可能である。このファイル・バック・アイテムが、第1のファイル・システムの特性、たとえば、限定されないけれども、属性、セキュリティ特性、ファイル・ストリーム、ファイルなど、を第2のファイル・システムに組み込むことが可能であることを認識し、理解されたい。

【0012】

本発明の一態様によれば、このファイル・システム・アダプタは、属性コンポーネントを含むことが可能である。この属性コンポーネントは、第1のファイル・システムに関連付けられたファイルおよび／またはファイル・ストリームに関連する属性を第2のファイル・システムに与えることが可能である。ファイルは、限定されないけれども、読み出し専用ファイル、圧縮済みファイル、アーカイブ・ファイルなど少なくとも1つの属性を有することが可能であり、そのような属性を、1つのビットおよび／または複数のビットで表すことが可能である。この属性コンポーネントは、第1のファイル・システム内のファイルおよび／またはファイル・ストリームを表すファイル・バック・アイテムがそのような属性を第2のファイル・システムに組み込むことを可能にすることができる。

【0013】

本発明の別の態様によれば、ファイル・システム・アダプタは、メタデータ・ハンドラ・コンポーネントを含むことが可能であり、これは、少なくとも1つのプロパティがアイテム、ファイル・バック・アイテム、およびバック・ファイル (たとえば、ファイル・バック・アイテムによって表される、ファイル・システム内のファイル) の間で同期した状態に保たれることを可能にする。プロパティは、バック・ファイル・ストリーム内で修正されることが可能である。この場合は、このメタデータ・ハンドラ・コンポーネントが第2のファイル・システム内でその新しい値をアイテム内の適切なフィールドにプロモートすることが可能である。さらにプロパティは、第2のファイル・システムに関連するA P Iを介して変更されることが可能であり、また、このメタデータ・ハンドラ・コンポーネントは、第1のファイル・システム内でその新しい値をバック・ファイル・ストリームにデモートすることが可能である。

【0014】

本発明のさらに別の態様によれば、このファイル・システム・アダプタは、ファイル・バック・アイテムのプロパティに対して同期化を提供するゴースト・コンポーネントを含むことが可能である。さらに、属性トンネル・コンポーネントは、第2のファイル・システム内での名前変更および／またはリソース保全を容易にするために属性トンネリングを行うことに利用することが可能である。さらにまた、このファイル・システム・アダプタは、本発明との併用で有向非周期グラフを利用するD A Gコンポーネントを含むことが可能である。本発明の他の態様は、複数の異なるファイル・システムを容易に適合させる方法を提供する。

【0015】

以下の説明および添付図面では、本発明の特定の例示的態様について詳細に説明する。これらの態様は、本発明の原理を用いることが可能である様々な形態のうちの一部を示し

ているに過ぎず、本発明は、そのような態様とそれらの等価物をすべて包含するものとする。本発明の他の利点や新規な特徴は、図面とともに考察したとき、以下の本発明の詳細説明から明らかになるであろう。

【発明を実施するための最良の形態】

【0016】

本出願に使用される際、「コンポーネント」、「システム」、「インターフェース」などの用語は、コンピュータ関連エンティティ、すなわち、ハードウェア、（たとえば、実行中の）ソフトウェア、および／またはファームウェアを意味するものとする。たとえば、コンポーネントは、プロセッサで実行中のプロセス、プロセッサ、オブジェクト、実行可能ファイル、プログラム、および／またはコンピュータとすることが可能である。たとえば、サーバで実行中のアプリケーションも、そのサーバも、コンポーネントとすることができ、1つまたは複数のコンポーネントが1つのプロセス内に常駐することが可能であり、1つのコンポーネントが1つのコンピュータに局所化したり、かつ／または2つ以上のコンピュータに分散したりすることが可能である。

10

【0017】

本発明は、図面を参照しながら説明され、同様の参照符号が、複数の図面を通して同様の要素を表すために用いられる。以下の説明では、本発明が深く理解されるように、例示を目的として、多数の具体的な細部を説明する。しかしながら、それらの具体的な細部がなくても本発明が実施可能であることは自明であろう。他の例では、本発明を容易に説明するために、よく知られた構造や装置をブロック図の形で示している。

20

【0018】

次に図に戻ると、図1は、あるファイル・システムを異なるファイル・システムと容易に適合させるシステム100を示す。ファイル・システム102は、ファイル・ストレージ・システムであることが可能であり、その構造は、ファイル・ストリームおよび／またはディレクトリに基づいている。ファイル・システム102は、ファイル・ストリームを利用することが可能であり、その結果、アプリケーションおよび／またはアプリケーション・プログラム・インターフェース（API）がそのファイル・ストリームを質的に強化（enrich）して、追加の構造を与えることが可能となる。ファイル・システム104は、アイテム、サブアイテム、プロパティ、およびリレーションシップが定義されて、ファイル・システム104内の情報が複合型のインスタンスとして表現されることが可能である、より複雑なモデルであることが可能である。アイテムは、ファイル・システム104における一貫性の最小単位として定義されることが可能であり、セキュリティ保護、シリアル化、同期化、コピー、バックアップ／復元などが独立に行われることが可能である。アイテムはタイプのインスタンスであり、ファイル・システム104内のすべてのアイテムは、アイテムの単一グローバル・エクステンションに格納することが可能である。ファイル・システム104は、少なくとも1つのアイテムおよび／またはコンテナ構造に基づくことが可能である。ファイル・システム104は、アイテムとしてファイルに埋め込まれる多彩なメタデータをエクスポーズするストレージ・プラットフォームであることが可能である。ファイル・システム104は、説明される機能をサポートするためにデータ・モデルによって表されるファイル・ストレージ・システムであることが可能であることを理解されたい。

30

40

【0019】

ファイル・システム・アダプタ106は、ファイル・システム102とファイル・システム104との間に互換性および／または統一性を与えることが可能である。互換性および／または統一性により、ファイル・システム102用のアプリケーションおよび／またはAPIがファイル・システム104に実装されることを可能にすることができる。ファイル・システム・アダプタ106は、ファイル・バック・アイテムを実装することが可能であり、このファイル・バック・アイテムは、ファイル・システム104に関連するアイテムであって、それに関連付けられた少なくとも1つのプロパティを取り込むファイルおよび／またはファイル・ストリームに依拠するファイル・システム104に関連するアイ

50

テム、とすることが可能である。このファイル・バック・アイテムは、ファイル・システム102と、十分に体系化されたファイル・システム104および／またはそれらに関連付けられたデータ・モデルとの間のブリッジとして動作することが可能である。たとえば、ファイル・システム104内の画像アイテムは、ファイル・バック・アイテムであることが可能であり、この場合、この画像アイテムは、標準的な画像フォーマット（たとえば、J P E G、T I F、B M Pなど）に準拠する、ファイル・システム102のファイルによってバッキングされる。言い換えると、このファイル・バック・アイテムは、ファイル・システム104におけるプロパティおよび／またはメタデータをアイテムに取り込むファイルおよび／または関連付けられた（たとえば、ファイル・システム102によって利用される）ファイル・ストリームを利用することが可能である。ファイル・システム・アダプタ106は、ファイル・バック・アイテムを利用して、ファイル・システム102の現在および以前のバージョンに関連する既存のA P Iによる操作だけでなく、ファイル・システム104に関連付けられたA P Iによる操作も可能にする。ファイル・システム・アダプタ106はまた、ファイル・システム102とファイル・システム104との間に同期化を提供することが可能である。特に、ファイル・バック・アイテムは、バッキング・ファイル内の対応するデータと同期したデータを与えることが可能である。このファイル・バック・アイテムが、これらに限定されないが、属性、セキュリティ特性、ファイル・ストリーム、ファイルなどファイル・システム102の特性をファイル・システム104に与えることが可能であることを理解されたい。

10

20

30

40

50

【0020】

システム100はさらにインターフェース・コンポーネント108を含み、インターフェース・コンポーネント108は、ファイル・システム・アダプタ106をほとんどすべてのオペレーティング・システムに統合するための様々なアダプタ、コネクタ、チャネル、通信パスなどを与える。さらに、インターフェース・コンポーネント108は、データおよびファイル・システム・アダプタ106との対話を可能にする様々なアダプタ、コネクタ、チャネル、通信パスなどを与えることが可能である。インターフェース・コンポーネント108はファイル・システム・アダプタ106に組み込まれているが、そのような実装がそのようには限定されないことを理解されたい。たとえば、インターフェース・コンポーネント108は、システム100に対してデータの受信または送信を行うスタンドアロン・コンポーネントであることが可能である。

【0021】

図2は、少なくとも2つの異なるファイル・システムを容易に適合させて統一性を与えるシステム200を示す。ファイル・システム・アダプタ206は、ファイル・システム202とファイル・システム204との間に統一性および／または互換性を与えることが可能である。しかしながら、ファイル・システム・アダプタ206は、複数の異なるファイル・システムを適合させることが可能であることを理解されたい。ファイル・システム202は、少なくとも1つのA P Iおよび／またはアプリケーションを利用する、ファイル・ストリームおよびディレクトリをベースとするファイル・システムであることが可能である。たとえば、ファイル・システム202に関連するA P Iおよび／またはアプリケーションは、限定されないが、16ビット、32ビット、その他のバージョンであることが可能である。ファイル・システム204は、アイテムを一貫性の単位として利用する複合構造であることが可能であり、その場合はメタデータ、プロパティ、リレーションシップが複合型のインスタンスとして実装される。ファイル・システム204は、データの形態（shape）を記述し、データに関する、ある種のセマンティクスの一貫性を意味する制約を宣言し、データ間のセマンティクスの関連付けを定義することが可能なデータ・モデルであることが可能である。ファイル・システム202に関連するA P Iおよび／またはアプリケーションは、ファイル・システム・アダプタ206を利用することによって、ファイル・システム204内で統一的に利用されることが可能である。ファイル・システム202、ファイル・システム204、およびファイル・システム・アダプタ206は、それぞれ、図1のファイル・システム102、ファイル・システム104、およびファイル

・システム・アダプタ 106 とほぼ同様であることが可能であることを理解されたい。

【0022】

ファイル・システム・アダプタ 206 は、ファイル・システム 202 を解析するアナライザ・コンポーネント 210 を含むことが可能である。アナライザ・コンポーネント 210 は、それに関連付けられたディレクトリおよび／またはファイルを決定し、ファイル・システム 204 に適合可能とし、かつ／またはそれに互換であるシステムを提供することを容易にすることが可能である。アナライザ・コンポーネント 210 は、少なくとも 1 つのリーフ・ノードを決定することが可能である。ファイル・システム 202 のすべてのディレクトリおよび／またはファイルがファイル・システム 204 内のアイテムとして表されることが可能であるのに対し、リーフ・ノードはファイル・バック・アイテムとして表されることが可能であることを理解されたい。したがって、リーフ・ノードではないディレクトリおよび／またはファイルは、ファイル・システム 204 内に存在するが、これに関連付けられたストリームは存在しない。

【0023】

ファイル・システム・アダプタ 206 は、ファイル・バック・アイテムを作成および／または生成するジェネレータ・コンポーネント 212 をさらに含むことが可能である。ジェネレータ・コンポーネント 212 は、ファイル・システム 202 の解析結果に少なくとも部分的に基づくファイル・バック・アイテムを生成することが可能である。たとえば、ジェネレータ・コンポーネント 212 は、ファイル・システム 202 に関連付けられた任意のリーフ・ノードについてファイル・バック・アイテムを生成することが可能であり、このファイル・バック・アイテムは、ファイル・システム 202 からのファイルおよび／またはファイル・ストリームによって質的に強化 (enrich) されているため、ファイル・システム 204 と互換である。ジェネレータ・コンポーネント 212 は、「ファイル・バック・アイテムが、関連付けられた (たとえば、バッキング) ファイル・ストリームを有すること」、「ファイル・バック・アイテムが他のアイテム (たとえば、それらが名前空間内のリーフ・ノードである) を含まないこと」、「ファイル・バック・アイテムがゴースト化されることが可能であること (後述)」、および「ファイル属性 (後述) がファイル・バック・アイテムに関して保持されること」のうちの少なくとも 1 つが当てはまるファイル・バック・アイテムを生成することが可能である。

【0024】

ファイル・バック・アイテムは、ファイル・システム 202 とファイル・システム 204 との間のブリッジとして動作することが可能である。ファイル・システム 204 のすべてのアイテムは、1) そのコンテンツの少なくとも一部がファイル・システム 202 のファイルおよび／またはファイル・ストリームからのものである場合、および／または 2) 基になるファイルのコンテンツが、ファイル・システム 202 に関連する少なくとも 1 つの API および／またはアプリケーションによって操作される場合には、ファイル・バック・アイテムとして指定することが可能である。ファイル・システム 204 内のファイル・バック・アイテムは、アイテムに関連するファイル・ストリームだけでなく、アイテムに関連付けられたメタデータ・プロパティも記憶することが可能である。ファイル・システム 202 のフォルダ／ディレクトリ／ファイル階層メカニズムは、ファイル・システム 204 においては、ファイル・バック・アイテムを含む他のアイテムを収容するコンテナ・アイテムを利用して表されることが可能であることを理解されたい。したがって、ファイル・システム 202 API を利用するアプリケーションは、ファイル・システム 204 のストレージ・メカニズムとシームレスに動作することが可能である。

【0025】

図 3 は、ファイル・ストリーム関連ファイル・システムとメタデータ関連ファイル・システムとを容易に適合させるシステム 300 を示す。ファイル・システム・アダプタ 306 は、ファイル・システム 302 と、それとは異なるファイル・システム 304 との間で、統一性および／または互換性のある対話を可能にすることができる。ファイル・システム・アダプタ 306 は、少なくとも 1 つのプロパティおよび／またはメタデータを、ファ

イル・システム 302 に関連付けられたファイルおよび／またはファイル・ストリームを使用して取り込むことが可能な、ファイル・システム 304 内のアイテムとして、ファイル・バック・アイテムを利用することが可能である。ファイル・バック・アイテムは、ファイル・システム 302（たとえば、ファイル・フォーマット）と、ファイル・システム 304（たとえば、十分に体系化された複合型のインスタンス）との間のブリッジとして動作することが可能である。

【0026】

ファイル・システム・アダプタ 306 は、ファイル・システム 302 に関連付けられたファイルおよび／またはファイル・ストリームに関連する属性をファイル・システム 304 に与える属性コンポーネント 310 を含むことが可能である。たとえば、ファイル・システム 302 は、ファイル・ストリーム・ベースのストレージ・システムを利用する。この場合、ファイルは、これらに限定されないけれども、読み出し専用ファイル、圧縮済みファイル、アーカイブ・ファイルなども、少なくとも 1 つの属性を有することが可能であり、そのような属性は、1 つのビットおよび／または複数のビットで表すことが可能である。属性コンポーネント 310 は、ファイル・システム 302 内のファイルおよび／またはファイル・ストリームを表すファイル・バック・アイテムがそのような属性をファイル・システム 304 に組み込むことを可能にすることができる。属性コンポーネント 310 は、そのような属性を、プロパティとして表現するか、かつ／または、プロパティとしてそれぞれのファイル・バック・アイテムに対応づけることが可能であることを理解されたい。一例では、属性コンポーネント 310 は、ファイル・システム 302 からの呼び出しによってファイル・ストリームを閉じた直後にファイル属性に対する更新を提供することが可能である。このプロパティは、ファイル・システム 304 に関連する A P I を用いて更新することが可能である（後述）。

【0027】

属性コンポーネント 310 はさらに、すべてのアイテムに共通である属性を収容可能なネスト型のアイテム属性を与えることが可能である。これらの属性は、ファイル・システム 304 を通して動作している任意のアプリケーションおよび／またはファイル・システム 304 に関連付けられた A P I によって設定可能とすることができる。アプリケーションによって利用される、かつ／またはファイル・システム 302 と互換とするための例として、以下の属性を与えることが可能である。

【0028】

10

20

30

【表 1】

名前	型 (type)	説明
IsArchive	FileStorageSystem. Boolean	アイテムがArchivalとしてマーキングされていればTRUE。 アプリケーションはこのプロパティを設定および使用することが可能である。 FileStorageSystemは、このプロパティのセマンティクスを解釈／実施することができない。 これはデフォルト=TRUEを有する。
IsHidden	FileStorageSystem. Boolean	アイテムが隠されるようにマーキングされていればTRUE。 アプリケーションはこの属性を用いてアイテムを表示したり隠したりすることが可能である。 FileStorageSystemは、このプロパティのセマンティクスを解釈／実施することができない。 これはデフォルト=FALSEを有する。
IsReadOnly	FileStorageSystem. Boolean	アイテムは読み出し専用としてマーキングされている。 アプリケーションはこれらのアイテムが更新されないようにすることが可能である。 FileStorageSystemは、このプロパティのセマンティクスを解釈／実施することができない。 これはデフォルト=FALSEを有する。
IsSystem	FileStorageSystem. Boolean	アイテムがオペレーティング・システムの一部かどうかを示す。アプリケーションは、このプロパティを更新することが可能であり、表示または他の目的のためにこれを解釈して用いることが可能である。 FileStorageSystemは、このプロパティのセマンティクスを解釈／実施することができない。 これはデフォルト=FALSEを有する。
IsTemporary	FileStorageSystem. Boolean	このアイテムが一時記憶用として用いられていることを示す。 FileStorageSystemは、このプロパティのセマンティクスを解釈／実施することができない。 これはデフォルト=FALSEを有する。

10

20

30

【0029】

ファイル・システム・アダプタ306は、少なくとも1つのプロパティがアイテム、ファイル・バック・アイテム、およびバック・ファイル（たとえば、ファイル・バック・アイテムによって表される、ファイル・システム302内のファイル）の間で同期した状態に保たれることを可能にするメタデータ・ハンドラ・コンポーネント312を含むことが可能である。たとえば、プロパティは、バック・ファイル・ストリーム内で修正されることが可能でありこの場合、メタデータ・ハンドラ・コンポーネント312は、ファイル・システム304内でその新しい値をアイテム内の適切なフィールドにプロモートすることが可能である。別の例では、プロパティは、ファイル・システム304に関連するAPIを介して変更されることが可能であり、メタデータ・ハンドラ・コンポーネント312は、ファイル・システム302内でその新しい値をバック・ファイル・ストリームにデモートすることが可能である。異なる型のファイル・バック・アイテムは、それ

40

50

に関連付けられた、異なるメタデータ・ハンドラ・コンポーネント 3 1 2 を有することが可能であることを理解されたい。

【0030】

たとえば、アプリケーションによってファイルが閉じられた後、メタデータ・ハンドラ・コンポーネント 3 1 2 によってプロモートが非同期で行われることが可能である。一方、デモートは非同期で行われることが可能である。あるアイテムとバックアップ・ファイルとの間でプロモート済みプロパティが同期しているかどうかを、特殊なアイテムプロパティで示すことが可能である。プロモーション・ステータスは、アイテム・プロモーション・エラーステータスを示すことが可能である。このステータスは、ファイル・バック・アップアイテムのデフォルト・ストリーム上でのファイル・クローズと非同期に設定されることが可能である。このステータスは、非ファイル・バック・アップアイテムについては NULL であることが可能である。次の表は、プロモーション・ステータスの可能な値を示している。

10

【0031】

【表 2】

型 (type)	値	状態	説明
Int32	0	READY	アイテムが変更された。プロモーションは正常に実行された。
	1	STALE	アイテムが変更された。プロモーションは実行されていない。
	2	ERROR_STALE	アイテムが変更された。プロモーションは実行されたが、何らかの致命的原因によって失敗した。
	3	ERROR_DELAYED	アイテムが変更された。プロモーションは実行されたが、何らかの過渡的原因によって失敗した。

20

【0032】

ファイル・システム 3 0 6 は、少なくとも 1 つのアイテムおよび／またはファイル・バック・アップアイテムが格納されているアイテム・テーブルを利用することが可能であることを理解されたい。このアイテム・テーブルは、このアイテム・テーブルに格納することが可能な各ファイル・バック・アップアイテムに、ファイル・ストリームおよび／またはファイル属性に関連付けるファイル・ストリーム・テーブルおよび／またはファイル属性テーブルを利用することが可能である。ファイル・ストリーム・テーブルおよび／またはファイル属性テーブルは、（たとえば、質的に強化 (enrich) するか、かつ／または少なくとも 1 つのプロパティを取り込むファイル・ストリームおよび／またはファイルを格納する）ファイル・バック・アップアイテムになるアイテムに関連するそれぞれのファイル・ストリームおよびファイル属性を格納することが可能である。ファイル属性はアイテム・テーブルに格納可能であることを理解されたい。ファイル属性セマンティクスは、たとえば、更新 API によって実装可能である。

30

【0033】

さらに、グローバル・アイテム・テーブルは、ファイル・バック・アップアイテムに対する以下の追加物を含むことが可能である。

40

【0034】

【表 3】

名前	型 (type)	説明
(TableItemの現在の列)		
LastAccessTime	FileStorageSystem.DateTime	ファイルまたはディレクトリが最後にアクセスされた時刻。ベースはこの属性を更新しない：必要に応じて更新可能。更新頻度が非常に高いので、同期していない。
StreamSize	BigInt	ストリームのサイズ。
AllocationSize	Bigint	ファイル・ストリームのアロケーション・サイズ。
PromotionStatus	Int32	プロモーション・エラー・フラグ

10

【0035】

たとえば、ファイル・ストリーム・テーブルは、以下のプロパティを有することが可能である。

20

【0036】

【表 4】

名前	型 (type)	説明
ItemId	Uniqueidentifier not null	FileStorageSystemアイテムの識別情報。
RowId	Uniqueidentifier not null ROWGULDCOL unique default (newid())	各ファイル・ストリームの一意識別子。
Stream	varbinary(max) FILESTREAM not null	ファイル・ストリーム

30

【0037】

以下に、ファイル・ストリーム・ストレージを利用するファイル・システムの例を示す。ここでは32ビットシステムを実装しているが、32ビット・システムは本発明の限定として見なされるべきではない。以下は、システム300およびファイル・ストレージ・システム(FSS)(たとえば、ファイル・システム304)によってサポートされることが可能な32ビット・ファイル・システムの属性の例である。

【0038】

40

【表 5】

32ビットファイル属性	FSS内の名前	FSS型 (type)	デフォルト値	信頼レベル	32ビット・セマンティクス	FSSセマンティクス
CreationTime	Created	DateTime	Time when item is created	Update API, FSA-Store APIs, WRITE_CONTROL	ファイルが生成された時刻を示す。	アイテムが生成された時刻を示す。
ChangeTime	Modified	DateTime	Time when item is created	Update API, FSA-Store APIs, WRITE_CONTROL	ChangeTime: 属性、ファイル・ストリーム、またはディレクトリ・コンテンツが変更された場合に更新可能。	アイテムが最後に変更された時刻を示す。
LastAccessTime	LastAccessTime	DateTime	Time when item is created	Update API, FSA-Store APIs, WRITE_CONTROL	ファイルまたはディレクトリがアクセスされた新しい時刻を示す。その新しいアクセス時刻が、LastAccessTime属性の値から1時間を超えて後の場合は更新される。	アイテムが最後にアクセスされた時刻を示す。そのアクセス時刻が、LastAccessTime属性の値から1時間を超えて後の場合は更新される。
FILE_ATTRIBUTE_HIDDEN	IsHidden	Bool	FALSE	WRITE_CONTROL	そのファイルまたはディレクトリは隠される。	ターゲット・アイテムが通常のフォルダ・リスティングに含まれていない。
EndOfFile	StreamSize	Int32	0	FSS-Store API	ファイル終了の絶対位置を、ファイル開始位置からのバイト・オフセットとして示す。ファイル終了へのバイト・オフセットを示す。	ファイル終了の絶対位置を、バッキング・ファイル・ストリームの開始位置からのバイト・オフセットとして示す。ファイル終了へのバイト・オフセットを示す。

10

20

30

【0039】

【表 6】

AllocationSize	AllocationSize	Int32	0	FSS_Store API	ファイル・アロケーション・サイズをバイト単位で示す。 通常、この値は、土台となる物理デバイスのセクタサイズまたはクラスタサイズの複数倍の値である。	バッキング・ファイル・ストリームのアロケーション・サイズをバイト単位で示す。通常、この値は、土台となる物理デバイスのセクタサイズまたはクラスタサイズの複数倍の値である。
FILE_ATTRIBUTES_ARCHIVE	IsArchive	Bool	TRUE	WRITE_CONTROL	ファイルまたはディレクトリは、アーカイブ・ファイルまたはアーカイブ・ディレクトリである。 アプリケーションはこの属性を用いてファイルにバックアップまたは除去のマーキングを施す。	アイテムは、アーカイブ・ファイルまたはアーカイブ・ディレクトリである。アプリケーションはこの属性を用いてファイルにバックアップまたは除去のマーキングを施す。
FILE_ATTRIBUTES_TEMPORARY	IsTemporary	Bool	FALSE	WRITE_CONTROL	Files: ファイルは一時記憶用として用いられる。	アイテムは一時記憶用として用いられる。
FILE_ATTRIBUTE_READONLY	IsReadOnly	Bool	FALSE	WRITE_CONTROL	ファイルは読み出し専用であることが可能である。	アイテムは読み出し専用であることが可能である。
FILE_ATTRIBUTE_SYSTEM	IsSystem	Bool	FALSE	WRITE_CONTROL	ファイルまたはディレクトリは、オペレーティング・システムの一部であるか、オペレーティング・システムによって独占的に用いられる。	アイテムは、オペレーティング・システムの一部であるか、オペレーティング・システムによって独占的に用いられる。
NumberOfLinks	Calculated Value				ファイルへのハード・リンクの数。	常に1。ただし、ルートディレクトリは0を有する。

10

20

30

【 0 0 4 0 】

【表 7】

DeletePending	Calculated Value				削除保留ステータス。TRUEは、ファイル削除が要求されていることを示す。	32ビットのFBIにおいて削除が保留されている場合はTRUE。それ以外の場合はFALSE。
FILE_ATTRIBUTES_DEVICE	Calculated Value				予約済み：使用しない。	常にFALSE。
FILE_ATTRIBUTES_DIRECTORY	Calculated Value				ハンドルはディレクトリを識別する。	ファイル・バック・アイテムの場合はFALSE。それ以外の場合はTRUE。
FILE_ATTRIBUTES_OFFLINE	IsGhosted	Bool	FALSE	このフラグは、32ビットAPIを介して設定可能とすべきではない。	この属性は、ファイルデータがオフラインストレージに物理的に移されたことを示す。この属性は、Remote Storage(階層型ストレージ管理ソフトウェア)によって用いられる。	アイテムがゴースト化されているかどうかを示す。

10

20

【0041】

上の例ではまた、以下の属性を部分的かつ／または完全にサポートすることが可能である。

【0042】

【表 8】

32ビットファイル属性	FSS内の名前	FSS型 (type)	デフォルト値	信頼レベル	セマンティクス	FSSセマンティクス
LastWriteTime	Modified	DateTime	CreationTime	UpdateAPIs, FSA-StoreAPI, WRITE_CONTROL	ファイル・ストリームが最後に修正された時刻か、ディレクトリの列挙結果が最後に変更された時刻を示す。	アイテムが最後に変更された時刻を示す。

30

【0043】

図4は、あるファイル・システムを異なるファイル・システムに適合させるか、かつ／または組み込むことを容易にするシステム400を示す。ファイル・システム・アダプタ406は、ファイル・システム402とファイル・システム404とに統一性を与えることが可能である。ファイル・システム402は、ファイル・ストリームに基づくことが可能であり、ディレクトリ、フォルダ、およびファイルを含むことが可能である。たとえば、ファイル・システム402は、16ビット・ファイル・システム、32ビット・ファイル・システム（たとえば、FATファイル・システム）であることが可能であり、API（たとえば、16ビットAPI、32ビットAPI）は、それぞれのオペレーティング・システムに、そのようなファイル・システムを利用する。ファイル・システム404は、複合型に基づくシステムであることが可能であり、型、アイテム、プロパティ、リレーションシップ、型 (type) のインスタンス、コンテナなどを含む。ファイル・システム40

40

50

4 は、情報の単位を、関連するメタデータおよび／またはプロパティを有するオブジェクトとして表すことを理解されたい。さらに、ファイル・システム 4 0 2 および 4 0 4 は、図 3 のファイル・システム 3 0 2 および 3 0 4、図 2 のファイル・システム 2 0 2 および 2 0 4、図 1 のファイル・システム 1 0 2 および 1 0 4 と実質的に類似することが可能である。

【0 0 4 4】

ファイル・システム・アダプタ 4 0 6 は、ファイル・システム 4 0 2 のファイル構造に少なくとも基づいてファイル・バック・アイテムを生成して、ファイル・システム 4 0 4 とのシームレスな統合を可能にすることができる。このファイル・バック・アイテムは、ファイルおよび／またはファイル・ストリームを取り込んだり、かつ／またはそれらによって質的に強化 (enrich) して、ファイル・システム 4 0 4 によって利用されるプロパティを作成および／またはアタッチしたりすることが可能である。ファイル・バック・アイテムは、ファイル・システム 4 0 2 用 A P I を、このファイル・バック・アイテムが、それに関連付けられたファイルおよび／またはファイル・ストリームを有する型 (type) であるため、その A P I フォーマットがファイル・ストリームを必要としているにもかかわらず、ファイル・システム 4 0 4 で利用可能にすることが可能である。言い換えると、ファイル・バック・アイテムは、ファイル・システム 4 0 2 とファイル・システム 4 0 4 との間のブリッジである。

【0 0 4 5】

ファイル・システム・アダプタ 4 0 6 は、ファイル・バック・アイテムのプロパティの同期化を提供するゴースト・コンポーネント 4 1 0 を含むことが可能である。ゴースト・コンポーネント 4 1 0 は、別のファイル・バック・アイテムの複製であってファイル・ストリームを含まないゴースト化アイテムを生成することが可能である。さらに、ゴースト・コンポーネント 4 1 0 は、ゴースト化アイテムを生成する操作および／または既存のアイテムをゴースト化アイテムに変換する操作、であるゴースト化を行うことが可能である。ゴースト・コンポーネント 4 1 0 はまた、ゴースト化アイテムを通常アイテムに変換する操作であるゴースト化解除 (unghosting) を行うことも可能である。ゴースト・コンポーネント 4 1 0 はファイル・システム・アダプタ 4 0 6 に組み込まれているが、ゴースト・コンポーネント 4 1 0 は独立コンポーネントおよび／またはスタンドアロン・ユニットであることも可能であることを理解されたい。

【0 0 4 6】

以下の表は、ゴースト・コンポーネント 4 1 0 の機能に関連する、さらなる説明を与えることが可能である。

【0 0 4 7】

【表 9】

名前	方向	型 (type)	説明
ItemId	IN	Guid	ゴースト化／ゴースト化解除するアイテムの Id。
isGhost	IN	BIT	アイテムがゴースト化されるべきか、ゴースト化解除されるべきかを示す。
concurrencyToken	IN	BIGINT	concurrencyToken は、アイテムの、コンカレンシ・トークン (concurrency token) の期待値である。入力値が NULL であれば、チェックが行われない。デフォルト値は NULL である。

【0 0 4 8】

一例では、ゴースト・コンポーネント 4 1 0 は、ファイル・システム 4 0 4 用エンティティをゴースト化することが可能である。そのアイテムがファイル・バック・アイテムでない場合は、エラーを返すことが可能である。そのアイテムが既にゴースト化されていれば、この操作は空操作であって成功を返す。ゴースト・コンポーネント 4 1 0 は、その情報を、そのゴースト化アイテム・ツリーにあるすべてのサブエンティティ (たとえば、す

すべての埋め込みアイテム、リレーションシップ、および／またはエクステンション)に伝播することが可能であり、そのエンティティ状態がゴースト化であることを通知するためにビットをTRUEに設定する。そのゴースト化アイテムに関するファイル・ストリーム・コンテンツは、ゴースト化が行われた後(たとえば、ゴースト化によってファイル・ストリームがNULLに設定された後)、削除することが可能である。

【0049】

別の例では、エンティティがゴースト・コンポーネント410によってゴースト化解除されることが可能である。そのアイテムがファイル・バック・アイテムでない場合は、エラーが返される。アイテムがゴースト化されていない場合、この操作は空操作であって成功を返す。エンティティがゴースト化されていないというエンティティ状態を伝えるためにビットをFALSEに設定することが可能である。長さゼロのファイル・ストリームを生成し、ファイル・ストリーム・テーブルに追加することが可能である。ゴースト化解除ではファイル・ストリーム・コンテンツが復元されないため、呼び出し側は、明示的に、ファイル・ストリームに適切なコンテンツを取り込まなければならない(たとえば、ほぼ同様のトランザクションで行うことが好ましい)。一例では、ゴースト化解除の用法パターンとして、「トランザクションを開く」、「アイテムをゴースト化解除する」、「ファイル・ストリーム・コンテンツを取り込み／復元する」、「トランザクションをコミットする」などがある。

【0050】

ファイル・システム・アダプタ406は、属性トンネル・コンポーネント(ATC)412を含むことが可能である。ATC412は、ファイル・システム404内での名前変更および／またはリソース保全を容易にするために属性トンネリングを行う。ファイル・システム402は、一時ファイルを利用することが可能である。そのようなファイルは、ファイルを開いたり、ファイルにアクセスしたり、ファイルを編集したりすると生成される。ファイルがファイル・システム402に保存されると、一時ファイル名と保存ファイル名とがスワップされる。たとえば、ファイル・システム402内でファイルfood.docが作成され、アクセスされることが可能であるとする。ファイル・システム402内でアクセスが行われると、ファイルf1.tmpが生成されることが可能である。food.docは、f2.tmpに移されることが可能である。この場合、f1.tmpはfood.docに移される。その後、ファイル・システム402においてファイルf2.tmpを削除することが可能である。しかしながら、ファイル・システム404の構造と、情報の単位を表すためのアイテムの利用に基づく、名前を更新するAPI呼び出しは、負荷のかかるリソースである。一般に、ATC412は、関連付けられたファイル・ストリームが2つのファイル・バック・アイテムの間でスワップされることを可能にする。たとえば、ファイル・ストリーム・テーブルに格納されている(複数の)ファイル・ストリームはATC412によって利用されることが可能であり、2つのファイル・ストリームのアイテムIDをスワップすることが可能である。ATC412を、ファイル・システム・アダプタ406に組み込まれているものとして示したが、ATC412は、スタンドアロン・コンポーネントであることも、任意の好適なコンポーネントに組み込まれることも、それらの組み合わせの形で用いられることも可能であることを理解されたい。

【0051】

図5は、複数の異なるファイル・システムを容易に適合させるために生成されるファイル・バック・アイテムを示す。特に、ファイル・バック・アイテム502は、ファイル・システム504とファイル・システム506との間で統一性を実現することが可能である。ファイル・システム504は、図4のファイル・システム402、図3のファイル・システム302、図2のファイル・システム202、および図1のファイル・システム102と実質的に類似することが可能であることを理解されたい。さらに、ファイル・システム506は、図4のファイル・システム404、図3のファイル・システム304、図2のファイル・システム204、および図1のファイル・システム104と実質的に類似することが可能であることを理解されたい。

10

20

30

40

50

【0052】

ファイル・バック・アイテム502は、（たとえば、更新API、および／または前述のメタデータ・ハンドラ・コンポーネントによって実装される）セマンティクスを与えるセマンティック・レイヤ508を提供することが可能である。ファイル・バック・アイテム502は、また、ファイル・ストリーム510を含む。ファイル・ストリーム510は、ファイル・システム504テクノロジーに含まれることが可能である。ファイル・システム504に関連付けられたアプリケーション（たとえば、32ビットのアプリケーションおよび／またはAPI）は、ファイル・バック・アイテム502を利用して、ファイル・システム506内に格納されているファイルのファイル・ストリームに、ダイレクトに操作することが可能である。多くのファイル・システム504テクノロジー、たとえば、限定

10

【0053】

【表10】

名前	型 (type)	説明
ItemId	Uniaueidentifier not null	File Storage System Itemの識別情報。
RowId	Uniqueidentifier not null ROWUmCOL unique default (newidp)	各ファイル・ストリームの一意識別子。
Stream	varbinary(max) FILESTREAM not null	ファイル・ストリーム。

20

【0054】

さらに、ファイル・バック・アイテム502は、ファイル属性512、エンティティ状態514、変更単位 (change units) 516、およびアイテム・データ518を含むことが可能である。ファイル属性512はアイテム・テーブルに格納されることが可能であり、アイテム・テーブルでは、ファイル属性セマンティクスが、たとえば、更新APIによって実装されることが可能である。エンティティ状態514は、エンティティ状態アイテム・ユーザ定義型 (UDT) フィールドに格納された属性を有することが可能である。エンティティ状態属性のセマンティクスは、更新APIおよび／またはメタデータ・ハンドラ・コンポーネント（図示せず）によって実装されることが可能である。さらに、ファイル・バック・アイテム502は、変更単位516を利用して、アイテムに関連するデータを表す、少なくとも1つの単位および／またはアイテム・データ518を変更することが可能である。

30

【0055】

一例として、以下の表は、エンティティ状態514の少なくとも1つの特性を表すことが可能である。

40

【0056】

【表 1 1】

名前	型 (type)	説明
IsCompoundItemType	Boolean	エンティティがItem型であって、スキーマ宣言において「複合アイテム (compound item)」として宣言されている場合は、TRUE。エンティティがItem型であって、スキーマ宣言において複合アイテムとして宣言されて「いない」場合 (たとえば、汎用コンテナとして宣言されている場合) は、FALSE。エンティティがItem型ではない場合 (たとえば、エンティティがリンク、アイテム・フラグメント、またはエクステンションである場合) は、このビットの値がその所有者／ソース・アイテムと同じになる。これはデフォルト=TRUEを有する。
RootCompoundItemId	Guid	このエンティティが格納されている、一番上の複合アイテムのItemIdを保存する。これが複合アイテム階層の一部でない場合 (たとえば、「真の汎用コンテナ」である場合)、値はNULLである。このエンティティが一番上のRoot複合アイテムである場合、値はItemIdと同じであることが可能である。このエンティティが複合アイテム階層内に含まれる場合、値は、階層の一番上のRootCompoundアイテムのItemIdである。これはデフォルト=FALSEを有する。
IsRootFileBackedItem	Boolean	アイテムがRoot File-Backedアイテムの場合は、TRUE。アイテムがFile-Backed Itemでない場合、またはアイテムがルートFile-backed Item (たとえば、File-Backedアイテム・ツリー内に含まれるアイテム) でない場合は、FALSE。 他のすべてのエンティティ (リンク、エクステンション、およびアイテム・フラグメント) について、このフラグは、それぞれの所有者／ソース・アイテムと同じ値を有する。これはデフォルト=FALSEを有する。
RootFileBackedItemId	Guid	このエンティティが格納されている、ルートFile-BackedアイテムのItemIdを保存する。 これがFile-Backed Itemの一部でない場合、値はNULL。 このエンティティがRoot File-Backedアイテムである場合、値はItemIdと同じであることが可能である。このエンティティがFileBacked Itemツリー内に含まれる場合、値は、Root File-BackedアイテムのItemIdである。
IsGhost	Boolean	エンティティが別のアイテムのゴースト化レプリカ (の一部) である場合は、TRUE。これはデフォルト=FALSEを有する。

10

20

30

【0057】

40

図6は、ファイル・ストリーム関連ファイル・システムとメタデータ関連ファイル・システムとを容易に適合させるシステム600を示す。ファイル・システム・アダプタ606は、ファイル・システム602 (たとえば、32ビットAPIを利用する、ファイル・システム・ストリーム・ベースのシステム) とファイル・システム604 (たとえば、情報の単位を表すためにアイテム、型、プロパティ、およびリレーションシップを利用するオブジェクト・ベースのシステム) との間に統一性および／またはシームレスな統合を与えることが可能である。ファイル・システム602は、図5のファイル・システム504、図4のファイル・システム402、図3のファイル・システム302、図2のファイル・システム202、および図1のファイル・システム102とかなりよく似ていることが可能であることを理解されたい。さらに、ファイル・システム604は、図5のファイル

50

・システム 506、図 4 のファイル・システム 404、図 3 のファイル・システム 304、図 2 のファイル・システム 204、および図 1 のファイル・システム 104 とかなりよく似ていることが可能であることを理解されたい。

【0058】

有向非周期グラフ (DAG) コンポーネント 608 は、本発明との併用で利用されることが可能である。DAG コンポーネント 608 は、周期がまったくない有向グラフを利用することが可能である。その場合、頂点に関しては、そのような頂点で始まって終わる有向パスが存在しない。ソースは、入ってくるエッジ (辺) を有しない頂点であり、シンクは、出ていくエッジ (辺) を有しない頂点である。有限の DAG は、少なくとも 1 つのソースと少なくとも 1 つのシンクとを有する。DAG の長さは、その長さ (たとえば、最長有向パスのエッジ (辺) の数) である。ファイル・システム 602 は DAG を組み込まないが、ファイル・システム 604 はファイル・システム・アダプタ 606 を介して DAG コンポーネント 608 を利用することが可能であることを理解されたい。DAG コンポーネント 608 をスタンドアロン・ユニットとして示したが、DAG コンポーネント 608 は、ファイル・システム・アダプタ 606 および／または他の任意の好適なコンポーネントに組み込まれることが可能である。DAG コンポーネント 608 を利用すると、ファイルは 2 つのディレクトリを同時に記述することが可能である。たとえば、ファイルがあるディレクトリから削除され、物理的に除去されたとき、そのファイルにとってディレクトリが親のままであることが可能である。

10

20

【0059】

図 7 は、あるファイル・システムを異なるファイル・システムに容易に適合させるインテリジェンスを用いるシステム 700 を示す。システム 700 は、ファイル・システム 702、ファイル・システム 704、ファイル・システム・アダプタ 706、およびインターフェース 108 を含むことが可能であり、これらはすべて、これまでの図で示したそれぞれのコンポーネントと実質的に類似することが可能である。システム 700 はさらに、インテリジェント・コンポーネント 708 を含む。インテリジェント・コンポーネント 708 は、少なくとも 2 つの異なるファイル・システムを容易に適合させるためにファイル・システム・アダプタ 706 によって利用されることが可能である。たとえば、インテリジェント・コンポーネント 708 を利用して、ファイル・システム 702 のファイル構造を容易に決定することが可能である。

30

【0060】

インテリジェント・コンポーネント 708 は、イベントおよび／またはデータを介して取り込まれた一連の観察結果から、システム、環境、および／またはユーザの状態を判断したり、またはそれを推論したりすることが可能であることを理解されたい。推論によって、特定のコンテキストまたはアクションを識別することや、たとえば、複数の状態にまたがる確率分布を生成することが可能である。推論は、確率論的であること、すなわち、データおよびイベントの考察に基づいて、関心の対象である複数の状態にまたがる確率分布を計算することであることが可能である。推論はまた、一連のイベントおよび／またはデータから上位のイベントを構成するための手法を意味する場合もある。そのような推論によって、一連の観察されたイベントおよび／または記憶されたイベント・データから新しいイベントまたはアクションが構築される。これらは、イベントが時間的に接近した近接性で関連しているかどうか、およびイベントおよびデータが 1 つまたは複数のイベント・ソースおよびデータ・ソースに由来しているかどうかにかかわらず行われる。本発明に関連する自動および／または推論によるアクションを実行することに関して、様々な (明示的および／または暗黙的に訓練された) 分類スキームおよび／またはシステム (たとえば、サポート・ベクトル・マシン、ニューラル・ネットワーク、エキスパート・システム、バイズ信頼ネットワーク、ファジー・ロジック、データ融合エンジン・・・) を用いることが可能である。

40

【0061】

分類器 (classifier: クラシファイア) は、入力属性ベクトル $x = (x_1, x_2, x_3$

50

、 x_4 、 x_n)を、その入力クラスに属する信頼度(confidence)、すなわち、 $f(x) = \text{confidence}(\text{class})$ にマッピングする関数である。そのような分類では、確率論的解析および／または統計ベースの解析(たとえば、解析の有用性とコストの因子分析)を用いて、ユーザが自動的に実行してほしいと考えるアクションを予測または推論することが可能である。サポートベクトルマシン(SVM)が、用いることが可能な分類子の一例である。SVMは、可能な入力の空間にある超曲面(hypersurface)を見つけることによって動作し、超曲面は、非トリガ・イベントからトリガ基準を分離しようとする。これにより、訓練データと同一ではないがそれに近いデータをテストする場合に、分類が直観的に正しいものになる。他の有向および無向モデル分類手法として、たとえば、ナイーブ・ベイズ、ベイズ・ネットワーク、決定木、ニューラル・ネットワーク、ファジー・ロジック・モデル、様々なパターンの独立性を使用した確率論的分類モデルなども利用できる。ここで用いている分類も、優先度のモデルを開発するために利用される統計回帰を含んでいる。

【0062】

プレゼンテーション・コンポーネント710は、ユーザと、ファイル・システム・アダプタ706に接続された任意のコンポーネントとの間の対話を容易にするための様々なタイプのユーザ・インターフェースを提供することが可能である。図に示すように、プレゼンテーション・コンポーネント710は、ファイル・システム・アダプタ706とともに利用されることが可能な、独立したエンティティである。しかしながら、プレゼンテーション・コンポーネント710および／または同様のビュー・コンポーネントは、ファイル・システム・アダプタ706に組み込まれるか、かつ／またはスタンドアロン・ユニットであることが可能であることを理解されたい。プレゼンテーション・コンポーネント710は、1つまたは複数のグラフィカル・ユーザ・インターフェース(GUI)、コマンド・ライン・インターフェース、その他を備えることが可能である。たとえば、GUIは、データのロード、インポート、読み出し、その他を行う領域または手段をユーザに提供するようにレンダリングされることが可能であり、それらの結果を表示する領域を含むことが可能である。それらの領域は、編集コントロール、コンボ・ボックス、ラジオ・ボタン、チェック・ボックス、プッシュ・ボタン、グラフィック・ボックスなどの、ダイアログ・ボックス、静的コントロール、ドロップダウン・メニュー、リスト・ボックス、ポップ・アップ・メニュー、を含む、知られているテキスト領域および／またはグラフィック領域を含むことが可能である。さらに、領域を表示可能にするかどうかを決定するナビゲーション用垂直および／または水平スクロール・バーやツールバー・ボタンなど、プレゼンテーションを容易にするユーティリティも用いることが可能である。たとえば、ユーザは、ファイル・システム・アダプタ706に接続されているコンポーネントのうちの1つまたは複数のコンポーネントと対話することが可能である。

【0063】

ユーザはまた、それらの領域と対話して情報を選択および入力するために、たとえば、マウス、ローラーボール、キーパッド、キーボード、ペン、および／または音声起動など、様々な機器を用いることが可能である。一般に、プッシュ・ボタンやキーボードのEnterキーなどの機構は、情報を入力した後、検索を開始するために用いられることが可能である。しかしながら、本発明はそのようには限定されないことを理解されたい。たとえば、チェック・ボックスをハイライトするだけで情報伝達を開始することが可能である。別の例では、コマンド・ライン・インターフェースを用いることも可能である。たとえば、コマンド・ライン・インターフェースは、(たとえば、テキスト・メッセージをディスプレイに表示し、オーディオ・トーンを発して)テキスト・メッセージによる情報入力をユーザに求めることが可能である。それに対しユーザは、インターフェース・プロンプトに示された選択肢に対応する英数字や、プロンプトによって投げかけられた質問への回答に対応する英数字など、適切な情報を入力することが可能である。コマンド・ライン・インターフェースは、GUIおよび／またはAPIと関連して用いられることが可能であることを理解されたい。さらに、コマンド・ライン・インターフェースは、グラフィック

・サポートが限定されていて、かつ／または通信チャネルの帯域幅が低いハードウェア（たとえば、ビデオカード）および／またはディスプレイ（たとえば、白黒やEGA）と関連して用いられることが可能である。

【0064】

図8～10は、本発明による手法を示す。説明を簡単にするために、一連の動作（acts）として手法を示し、説明する。本発明は、例示した動作（act）および／または動作（acts）の順序によって限定されないことを理解されたい。たとえば、動作（acts）は、様々な順序で、かつ／または同時に行われることが可能であり、本明細書で提示および説明していない他の動作（acts）とともに行われることも可能である。さらに、本発明による手法を実施するために、例示したすべての動作が必要であるとは限らない場合がある。さらに、当業者であれば、本手法が、状態図またはイベントによって相互に関係づけられた状態の系列として表されることも可能であることを理解されよう。

10

【0065】

図8は、少なくとも2つの異なるファイル・システムを容易に適合させるか、かつ／またはそれらのファイル・システムに統一性を与える手法800を示す。第1のファイル・システムは、ファイル・ストレージ・システムであることが可能であり、その構造は、ファイル・ストリームおよび／またはディレクトリに基づいている。第1のファイル・システムは、ファイル・ストリームを利用することが可能であり、その結果、アプリケーションおよび／またはAPIがそのファイル・ストリームを質的に強化（enrich）して、追加の大きな構造を与えることが可能となる。第2のファイル・システムは、アイテム、サブアイテム、プロパティ、およびリレーションシップが定義されて、第2のファイル・システム内の情報が複合型のインスタンスとして表現されることが可能である、より複雑なモデルであることが可能である。アイテムは、第2のファイル・システムにおける統一性の最小単位として定義されることが可能であり、セキュリティ保護、シリアル化、同期化、コピー、バックアップ／復元などが独立に行われることが可能である。アイテムは型（type）のインスタンスであり、第2のファイル・システム内のすべてのアイテムは、アイテムの単一グローバル・エクステンツに格納することが可能である。第2のファイル・システムは、少なくとも1つのアイテムおよび／またはコンテナ構造に基づくことが可能である。第2のファイル・システムは、アイテムとしてファイルに埋め込まれる多彩なメタデータをエクスポートするストレージ・プラットフォームであることが可能である。

20

30

【0066】

参照符号802では、ファイル・ストリームに基づく第1のファイル・システムを解析することが可能である。そのような解析によって、ファイル構造と、統一性を与えるために利用されるコンテンツとを決定することが可能である。参照符号804では、解析に少なくとも基づいてファイル・バック・アイテムを生成することが可能である。たとえば、解析によって、32ビット・ファイル・ストレージ・システムと、そのような構造に関連付けられたリーフ・ノードの数とを決定することが可能である。したがって、第1のファイル・システムに関連付けられたリーフ・ノードごとにファイル・バック・アイテムを生成することが可能であり、アイテムを利用して、リーフ・ノードではないファイルおよび／またはディレクトリを表すことが可能である。ファイル・バック・アイテムは、第2のファイル・システム内のアイテムであって、その複数のプロパティの一部またはすべてを取り込むファイルおよび／またはファイル・ストリームに依拠するアイテムであることが可能である。ファイル・バック・アイテムは、第1のファイル・システムと第2のファイル・システムとの間のブリッジとして動作することが可能である。参照符号806では、ファイル・バック・アイテムを利用し、ファイル・ストリームおよびファイルを利用する第1のファイル・システムと、第1のファイル・システムに関連付けられることが可能なファイルおよび／またはファイル・ストリームを含むファイル・バック・アイテムを組み込んだ第2のファイル・システムと、に少なくとも部分的に基づいて、異なる2つのファイル・システムの間で統一性を実現することが可能である。

40

【0067】

50

図 9 は、あるファイル・システムを異なるファイル・システムに適合させる手法 900 を示す図である。参照符号 902 では、ファイルおよび／またはファイル・ストリームをその構造として利用する第 1 のファイル・システムを解析する。そのような解析から、参照符号 904 において、第 1 のファイル・システムのリーフ・ノードでない各ディレクトリおよび／またはファイルについてアイテムを生成する。参照符号 906 で、リーフ・ノードが第 1 のファイル・システムに存在することを決定し、それへのファイル・バック・アイテムを生成することが可能である。ファイル・バック・アイテムは、情報の単位を表す型の複合インスタンスを利用する第 2 のファイル・システムに関連付けられたアイテムである。参照符号 908 では、ファイル・バック・アイテムは、そのファイルを第 2 のファイル・システム内のアイテムとして表すために、質的に強化 (enrich) され、かつ／または、ファイル、ファイル・ストリームを使用して、属性、セキュリティ特性などを取り込む。参照符号 910 では、ファイル・バック・アイテムを利用して、第 1 のファイル・システムを第 2 のファイル・システムに適合させる。

10

20

30

40

50

【0068】

図 10 は、ファイル・ストリーム構造のファイル・システムとメタデータ関連ファイル・システムとを容易に適合させる手法 1000 を示す。参照符号 1002 で、ファイル・ストリーム構造のファイル・システム (たとえば、第 1 のファイル・システムと称する) を解析する。参照符号 1004 で、第 1 のファイル・システムの解析結果に少なくとも部分的に基づいてファイル・バック・アイテムを生成することが可能である。このファイル・バック・アイテムは、そのプロパティのうちの少なくとも 1 つをファイルおよび／またはファイル・ストリームによって質的に強化 (enrich) することが可能な、第 2 のファイル・システムに関連付けられたアイテムである。1006 では、ファイル・バック・アイテムを利用して、第 1 のファイル・システムを第 2 のファイル・システムに適合させることが可能である。

【0069】

参照符号 1008 では、第 1 のファイル・システムと第 2 のファイル・システムとの間で属性の整合性および同期がとれるようにするゴースト化が提供される。たとえば、ゴースト化アイテムが、ファイル・ストリームを含まないけれども別のファイル・バック・アイテムの複製となる、ように、ゴースト化アイテムを生成することが可能である。さらに、ゴースト化アイテムを生成する、操作および／または既存のアイテムをゴースト化アイテムに変換する操作であるゴースト化を行うことが可能である。さらに、ゴースト化解除を利用することが可能であり、これは、ゴースト化アイテムを通常のアイテムに変換する操作である。参照符号 1010 では、第 1 のファイル・システムからの属性を、第 2 のファイル・システムに関連付けられたファイル・バック・アイテムに組み込んだり、かつ／またはこのファイル・バック・アイテムで利用したりすることが可能である。第 1 のファイル・システムは、限定されないが、読み出し専用ファイル、圧縮済みファイル、アーカイブ・ファイルなどすることが可能な属性を含むことが可能であり、このような属性は、1 つのビットおよび／または複数のビットで表すことが可能である。この属性は、ファイル・バック・アイテムを介して第 2 のファイル・システムに組み込まれることが可能であることを理解されたい。

【0070】

参照符号 1012 では、属性トンネリングおよび／または DAG を利用することが可能である。属性トンネリングにより、少なくとも 2 つのファイル・バック・アイテムの間でファイル・ストリームをスワップし、第 2 のファイル・システムにおいてリソースの保全 (conservation) および／または一時ファイル間での名前変更をすることを可能となる。DAG は、周期を有しない、接続されたアイテムのグラフを提供することが可能である。参照符号 1014 では、そのファイル・バック・アイテムを用いることによって、第 1 のファイル・システムからの API を、第 2 のファイル・システムで利用することが可能であり、これによってシームレスな統合が行われる。言い換えると、このファイル・バック・アイテムは、アイテムに関連付けられたファイル・ストリームだけでなく、アイテムに

関連付けられたメタデータ・プロパティも含む。このファイル・バック・アイテムは、第1のファイル・システムのAPI部分だけでなく、第2のファイル・システムに関連付けられたAPIによっても、操作可能である。

【0071】

本発明の各種態様を実装するための補足的なコンテキストを提供するために、以下では、図11～12を参照しながら、本発明の各種態様を実装することが可能である好適なコンピューティング環境について簡単かつ一般的に説明する。ここまで本発明について、ローカル・コンピュータおよび／またはリモート・コンピュータで実行されるコンピュータ・プログラムの、コンピュータで実行可能な命令の一般的なコンテキストで説明してきたが、当業者であれば、本発明が他のプログラム・モジュールとの組み合わせでも実装可能であることを理解されよう。一般に、プログラム・モジュールには、ルーチン、プログラム、コンポーネント、データ構造体などがあり、これらは、特定のタスクを実行したり、特定の抽象データタイプを実装したりする。

10

【0072】

さらに、当業者であれば、本発明の方法が他のコンピュータ・システム構成でも実施可能であることを理解されよう。そうした構成として、単一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、ならびに、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／またはプログラマブルである家庭用電子機器などがあり、これらのそれぞれは1つまたは複数の関連機器と動作可能に通信することが可能である。例示した、本発明の態様は、通信ネットワーク経由でリンクされたりリモート処理装置によって特定のタスクが実行される分散コンピューティング環境でも実施可能である。しかしながら、本発明の、全部でなくとも一部の態様は、スタンドアロン・コンピュータにおいても実施可能である。分散コンピューティング環境では、プログラム・モジュールは、ローカルおよび／またはリモートのメモリ・ストレージ装置に配置されることが可能である。

20

【0073】

図11は、本発明と相互に作用することが可能なコンピューティング環境例1100の概略ブロック図である。システム1100は、1つまたは複数のクライアント1110を含む。クライアント1110は、ハードウェアおよび／またはソフトウェア（たとえば、スレッド、プロセス、コンピューティング装置など）であることが可能である。システム1100はさらに、1つまたは複数のサーバ1120を含む。サーバ1120は、ハードウェアおよび／またはソフトウェア（たとえば、スレッド、プロセス、コンピューティング装置など）であることが可能である。サーバ1120は、たとえば本発明を用いて変換を実行するスレッドを収容することが可能である。

30

【0074】

クライアント1110とサーバ1120との間の可能な通信の1つは、2つ以上のコンピュータ・プロセスの間で伝送されるように適合されたデータ・パケットの形をとることが可能である。システム1100は、クライアント1110とサーバ1120との間の通信を促進するために用いることが可能な通信フレームワーク1140を含む。クライアント1110は、情報を記憶するために用いられることが可能な、クライアント1110の近くにある1つまたは複数のクライアント・データ・ストア1150に動作可能に接続される。同様に、サーバ1120は、情報を記憶するために用いられることが可能な、サーバ1120の近くにある1つまたは複数のサーバ・データ・ストア1130に動作可能に接続される。

40

【0075】

図12に示す、本発明の各種態様を実装する例示的環境1200は、コンピュータ1212を含む。コンピュータ1212は、処理ユニット1214、システム・メモリ1216、およびシステム・バス1218を含む。システム・バス1218は、システム・メモリ1216を含むがこれに限定されないシステム・コンポーネントを処理ユニット121

50

4に接続する。処理ユニット1214は、各種の市販プロセッサのどれであってもよい。デュアル・マイクロ・プロセッサや他のマルチ・プロセッサ・アーキテクチャも、処理ユニット1214として用いることが可能である。

【0076】

システム・バス1218は、メモリ・バスまたはメモリ・コントローラ、ペリフェラル・バスまたは外部バス、および／またはローカル・バスを含む様々なタイプのバス構造のどれであってもよく、それに用いられる任意の様々な市販バス・アーキテクチャとして、Industrial Standard Architecture (ISA)、Micro-Channel Architecture (MSA)、Extended ISA (EISA)、Intelligent Drive Electronics (IDE)、VESA Local Bus (VLB)、Peripheral Component Interconnect (PCI)、Card Bus、Universal Serial Bus (USB)、Advanced Graphics Port (AGE)、Personal Computer Memory Card International Associationバス (PCMCIA)、Firewire (IEEE 1394)、Small Computer Systems Interface (SCSI) などがあり、これらに限定されない。 10

【0077】

システム・メモリ1216は、揮発性メモリ1220および不揮発性メモリ1222を含む。基本入出力システム (BIOS) は、コンピュータ1212内で起動中などに要素間の情報転送を行う基本ルーチンを含み、不揮発性メモリ1222に格納される。限定ではなく例示として、不揮発性メモリ1222は、読み出し専用メモリ (ROM)、プログラマブルROM (PROM)、電氣的プログラマブルROM (EPROM)、電氣的消去可能プログラマブルROM (EEPROM)、またはフラッシュ・メモリを含むことが可能である。揮発性メモリ1220は、外部キャッシュ・メモリとして動作するランダムアクセスメモリ (RAM) を含む。限定ではなく例示として、RAMは、静的RAM (SRAM)、動的RAM (DRAM)、同期DRAM (SDRAM)、ダブル・データ・レートSDRAM (DDR SDRAM)、エンハンスドSDRAM (ESDRAM)、シンクリンクDRAM (SLDRAM)、Rambus Direct RAM (RDRAM)、Direct Rambus Dynamic RAM (DRDRAM)、Rambus Dynamic RAM (RDRAM) など、様々な形で利用可能である。 20 30

【0078】

コンピュータ1212はさらに、リムーバブル／非リムーバブル、揮発性／不揮発性コンピュータ・ストレージ媒体を含む。たとえば、図12は、ディスク・ストレージ1224を示している。ディスク・ストレージ1224は、磁気ディスク・ドライブ、フロッピー (登録商標) ディスク・ドライブ、テープ・ドライブ、Jazドライブ、Zipドライブ、LS-100ドライブ、フラッシュ・メモリ・カード、メモリ・スティックなどの装置を含み、これらに限定されない。さらに、ディスク・ストレージ1224は、ストレージ媒体を別個に、または他のストレージ媒体と組み合わせて含むことが可能であり、そのようなストレージ媒体として、コンパクト・ディスクROM装置 (CD-ROM)、書き込み可能CD装置 (CD-Rドライブ)、書き換え可能CD装置 (CD-RWドライブ)、デジタル多用途ディスクROMドライブ (DVD-ROM) などの光ディスク・ドライブがあり、これらに限定されない。ディスク・ストレージ装置1224とシステム・バス1218との接続を容易にするために、インターフェース1226のようなリムーバブルまたは非リムーバブル・インターフェースを一般的に用いる。 40

【0079】

図12では、ユーザと、好適な動作環境1200に示されている基本コンピュータ・リソースとの間の媒介として動作するソフトウェアが示されていることを理解されたい。そのようなソフトウェアは、オペレーティング・システム1228を含む。オペレーティング・システム1228は、ディスク・ストレージ1224に格納されることが可能であっ 50

て、コンピュータ・システム 1 2 1 2 のリソースの管理および配分を行うように動作する。システム・アプリケーション 1 2 3 0 は、システム・メモリ 1 2 1 6 またはディスク・ストレージ 1 2 2 4 に格納されたプログラム・モジュール 1 2 3 2 およびプログラム・データ 1 2 3 4 を介して、オペレーティング・システム 1 2 2 8 によるリソース管理を利用する。本発明は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・システムの組み合わせとともに実装可能であることを理解されたい。

【0080】

ユーザは、入力装置 1 2 3 6 を介してコンピュータ 1 2 1 2 にコマンドや情報を入力する。入力装置 1 2 3 6 として、マウスのようなポインティング・デバイス、トラック・ボール、スタイラス、タッチ・パッド、キーボード、マイク、ジョイスティック、ゲーム・パッド、パラボラアンテナ、スキャナ、TV チューナカード、デジタルカメラ、デジタルビデオカメラ、Web カメラなどがあり、これらに限定されない。これらの入力装置や他の入力装置は、インターフェース・ポート 1 2 3 8 からシステム・バス 1 2 1 8 を経て処理ユニット 1 2 1 4 に接続される。インターフェース・ポート 1 2 3 8 としては、たとえば、シリアル・ポート、パラレル・ポート、ゲーム・ポート、ユニバーサル・シリアル・バス (USB) などがある。出力装置 1 2 4 0 は、入力装置 1 2 3 6 と同じタイプのポートのうちのいくつかを用いる。したがって、たとえば、USB ポートを用いて、入力をコンピュータ 1 2 1 2 に供給したり、コンピュータ 1 2 1 2 からの情報を出力装置 1 2 4 0 に出力したりすることが可能である。出力アダプタ 1 2 4 2 は、出力装置 1 2 4 0 の中でも特にモニタ、スピーカ、プリンタのような、特殊なアダプタを要する出力装置 1 2 4 0 を例示するために提供されている。限定ではなく例示として、出力アダプタ 1 2 4 2 には、出力装置 1 2 4 0 とシステム・バス 1 2 1 8 との間の接続手段を提供するビデオカードおよびサウンドカードが含まれる。リモート・コンピュータ 1 2 4 4 のように入力機能と出力機能の両方を備えた他の装置および／または装置のシステムもあることに注意されたい。

【0081】

コンピュータ 1 2 1 2 は、リモート・コンピュータ 1 2 4 4 のような 1 つまたは複数のリモート・コンピュータとの論理接続を用いることにより、ネットワーク環境で動作可能である。リモート・コンピュータ 1 2 4 4 としては、パーソナル・コンピュータ、サーバ、ルータ、ネットワーク PC、ワークステーション、マイクロ・プロセッサ・ベースの機器、ピア・デバイスまたは他の一般的なネットワーク・ノードなどを用いることが可能であり、リモート・コンピュータ 1 2 4 4 は、一般に、コンピュータ 1 2 1 2 に関して前述した要素のほとんどまたはすべてを含む。簡単のために、メモリ・ストレージ装置 1 2 4 6 だけをリモート・コンピュータ 1 2 4 4 とともに図示した。リモート・コンピュータ 1 2 4 4 は、ネットワーク・インターフェース 1 2 4 8 を介してコンピュータ 1 2 1 2 に論理的に接続され、物理的には通信接続 1 2 5 0 を介して接続されている。ネットワーク・インターフェース 1 2 4 8 は、ローカル・エリア・ネットワーク (LAN) やワイド・エリア・ネットワーク (WAN) などの有線および／または無線通信ネットワークを包含する。LAN テクノロジとしては、ファイバ分散データ・インターフェース (FDDI)、銅分散データ・インターフェース (Copper Distributed Data Interface) (CDDI)、イーサネット (登録商標)、トークン・リングなどがある。WAN テクノロジとしては、ポイント・ツー・ポイント・リンク、統合サービス・デジタル網 (ISDN) やその変形物のような回路交換網、パケット交換網、デジタル加入者線 (DSL) などがあり、これらに限定されない。

【0082】

通信接続 1 2 5 0 は、ネットワーク・インターフェース 1 2 4 8 をバス 1 2 1 8 に接続するために用いられるハードウェア／ソフトウェアを指す。通信接続 1 2 5 0 は、わかりやすく示すためにコンピュータ 1 2 1 2 の内部にあるように図示されているが、コンピュータ 1 2 1 2 の外部にあってもよい。ネットワーク・インターフェース 1 2 4 8 への接続に必要なハードウェア／ソフトウェアとして、あくまで例であるが、一般電話グレードの

10

20

30

40

50

モデム、ケーブルモデム、DSLモデムなどを含むモデム、ISDNアダプタ、イーサネット（登録商標）カードなどの、内蔵技術および外付け技術とがある。

【0083】

ここまで述べてきた中には、本発明の例が含まれている。本発明を説明するために、コンポーネントまたは方法の、考えられるすべての組み合わせを説明することはもちろん不可能であるが、当業者であれば、本発明の他の多くの組み合わせや並べ替えが可能であることを理解されよう。したがって、本発明は、添付の特許請求項の趣旨および範囲に当てはまる代替物、修正物、および変形物のすべてを包含するものとする。

【0084】

特に、前述のコンポーネント、装置、回路、システム、その他によって実行される各種機能については、そのようなコンポーネントを説明するために用いている用語（「手段（means）」への参照を含む）は、特に断らない限り、説明されているコンポーネントの明示された機能を実行する任意のコンポーネント（たとえば、機能等価物）に対応するものとする。これは、開示された構造と構造的に等価でなくても、本明細書で示された、本発明の例示的態様における機能を実行するものであれば当てはまる。この点において、本発明が、システムならびに、本発明の各種方法の動作および／またはイベントを実行する、コンピュータで実行可能な命令を有するコンピュータ可読媒体を含むことも理解されよう。

【0085】

さらに、様々な実施態様のうちの1つに関してのみ本発明の特定の特徴を開示したが、そのような特徴は、他の実施態様の1つまたは複数の他の特徴と組み合わせられることが可能であり、このことは、任意または特定の応用例にとって望ましいものであり、有利な点である可能性がある。さらに、用語「含む（includes、including）」およびその変形物が詳細説明または特許請求の範囲に用いられる限りにおいては、これらの用語は、用語「備える（comprising）」と同様に包含的であるものとする。

【図面の簡単な説明】

【0086】

【図1】ファイル・ストリーム関連ファイル・システムとメタデータ関連ファイル・システムとを容易に適合させる例示的システムのブロック図である。

【図2】少なくとも2つの異なるファイル・システムを容易に適合させて統一性を与える例示的システムのブロック図である。

【図3】ファイル・ストリーム関連ファイル・システムとメタデータ関連ファイル・システムとを容易に適合させる例示的システムのブロック図である。

【図4】少なくとも2つの異なるファイル・システムを容易に同期化させる例示的システムのブロック図である。

【図5】あるファイル・システムを異なるファイル・システムと容易に適合させるファイル・バック・アイテムのブロック図である。

【図6】ファイル・ストリーム関連ファイル・システムとメタデータ関連ファイル・システムとを容易に適合させる例示的システムのブロック図である。

【図7】ファイル・ストリーム関連ファイル・システムとメタデータ関連ファイル・システムとを容易に適合させる例示的システムのブロック図である。

【図8】少なくとも2つの異なるファイル・システムを適合させて統一性を与える例示的手法を示す図である。

【図9】あるファイル・システムを異なるファイル・システムに適合させる例示的手法を示す図である。

【図10】ファイル・ストリーム関連ファイル・システムとメタデータ関連ファイル・システムとを適合させる例示的手法を示す図である。

【図11】本発明の新規な態様を用いることが可能な例示的ネットワーキング環境を示す図である。

【図12】本発明に従って用いることが可能な例示的運用環境を示す図である。

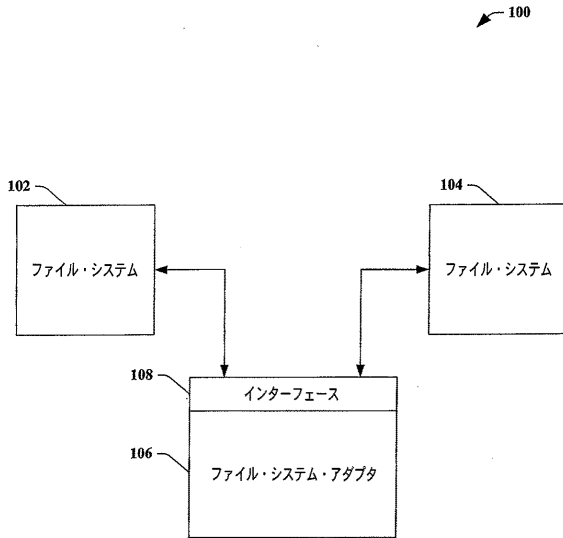
【符号の説明】

【0087】

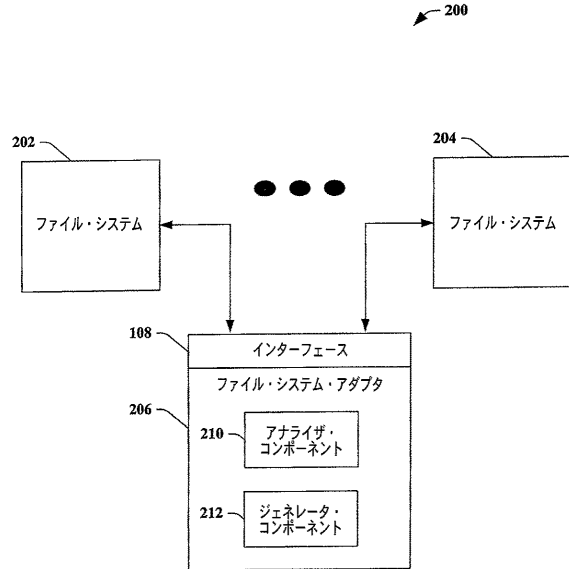
100	ファイル・システムを異なるファイル・システムと容易に適合させるシステム	
102	ファイル・システム	
104	ファイル・システム	
106	ファイル・システム・アダプタ	
108	インターフェース	
200	少なくとも2つの異なるファイル・システムを容易に適合させて統一性を与えるシステム	
202	ファイル・システム	10
204	ファイル・システム	
206	ファイル・システム・アダプタ	
210	アナライザ・コンポーネント	
212	ジェネレータ・コンポーネント	
300	ファイル・ストリーム関連ファイル・システムとメタデータ関連ファイル・システムとを容易に適合させるシステム	
302	ファイル・システム	
304	ファイル・システム	
306	ファイル・システム・アダプタ	
310	属性コンポーネント	20
312	メタデータ・ハンドラ・コンポーネント	
400	あるファイル・システムを異なるファイル・システムに適合させるか、かつ／または組み込むことを容易にするシステム	
402	ファイル・システム	
404	ファイル・システム	
406	ファイル・システム・アダプタ	
410	ゴースト・コンポーネント	
412	属性トンネル・コンポーネント	
502	ファイル・バック・アイテム	
504	ファイル・システム	30
506	ファイル・システム	
508	セマンティック・レイヤ	
510	ファイル・ストリーム	
512	ファイル属性	
514	エンティティ状態	
516	変更単位	
518	アイテム・データ	
600	ファイル・ストリーム関連ファイル・システムとメタデータ関連ファイル・システムとを容易に適合させるシステム	
602	ファイル・システム	40
604	ファイル・システム	
606	ファイル・システム・アダプタ	
608	DAGコンポーネント	
700	あるファイル・システムを異なるファイル・システムに容易に適合させるインテリジェンスを用いるシステム	
702	ファイル・システム	
704	ファイル・システム	
706	ファイル・システム・アダプタ	
708	インテリジェント・コンポーネント	
710	プレゼンテーション・コンポーネント	50

1 1 0 0	本発明と相互に作用することが可能なコンピューティング環境例	
1 1 1 0	クライアント	
1 1 2 0	サーバ	
1 1 3 0	サーバ・データ・ストア	
1 1 4 0	通信フレームワーク	
1 1 5 0	クライアント・データ・ストア	
1 2 0 0	本発明の各種態様を実装する例示的環境	
1 2 1 4	処理ユニット	
1 2 1 6	システム・メモリ	
1 2 1 8	バス	10
1 2 2 0	揮発性	
1 2 2 2	不揮発性	
1 2 2 4	ディスク・ストレージ	
1 2 2 6	インターフェース	
1 2 2 8	オペレーティング・システム	
1 2 3 0	アプリケーション	
1 2 3 2	モジュール	
1 2 3 4	データ	
1 2 3 6	入力装置	
1 2 3 8	インターフェース・ポート	20
1 2 4 0	出力装置	
1 2 4 2	出力アダプタ	
1 2 4 4	リモート・コンピュータ	
1 2 4 6	メモリ・ストレージ	
1 2 4 8	ネットワーク・インターフェース	
1 2 5 0	通信接続	

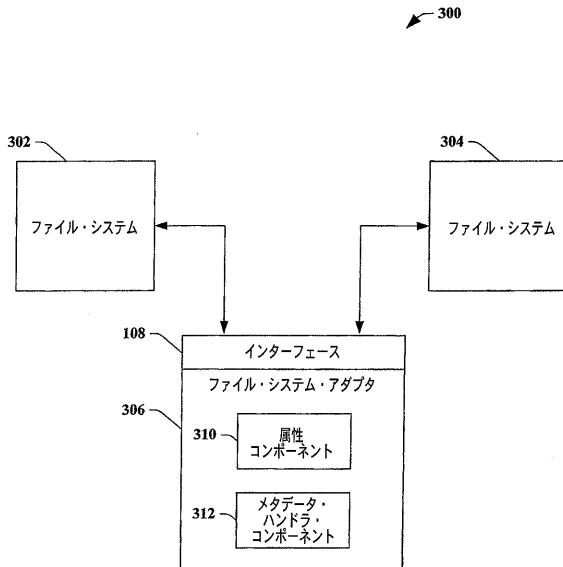
【図 1】



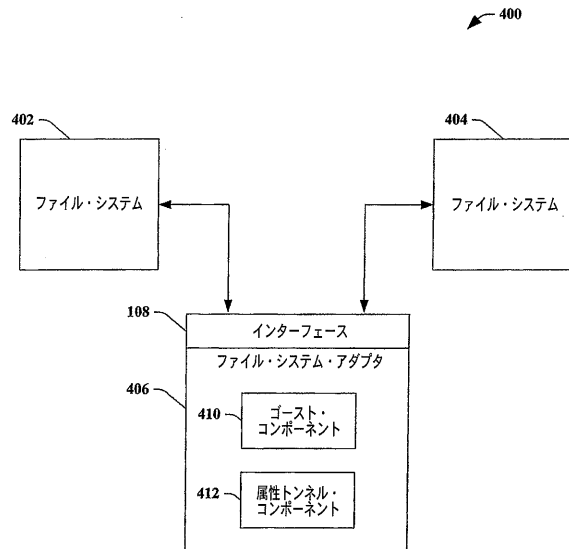
【図 2】



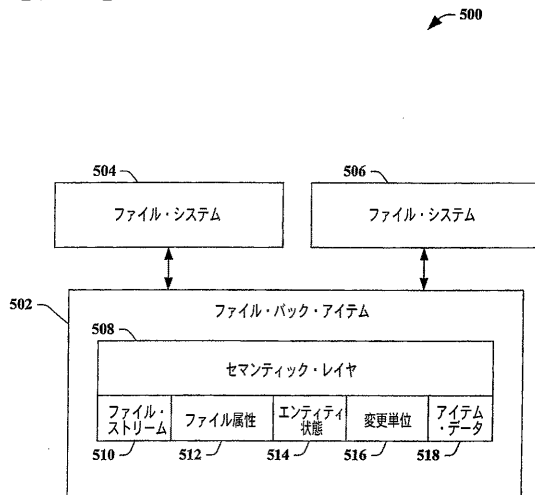
【図 3】



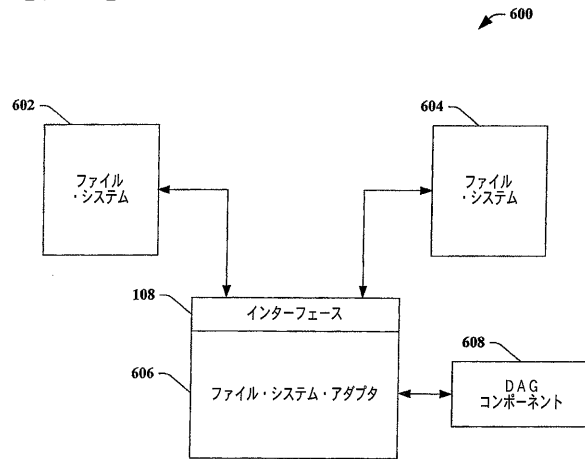
【図 4】



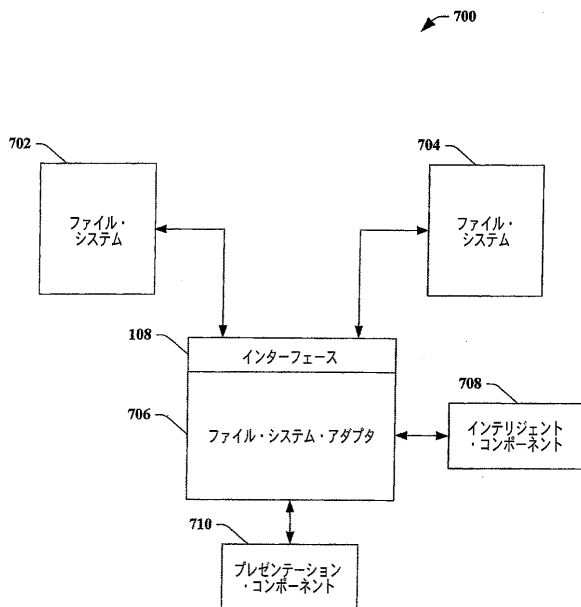
【図 5】



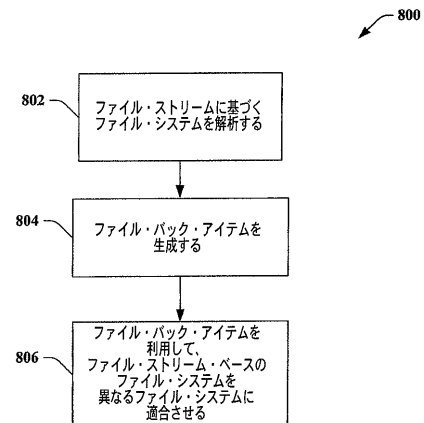
【図 6】



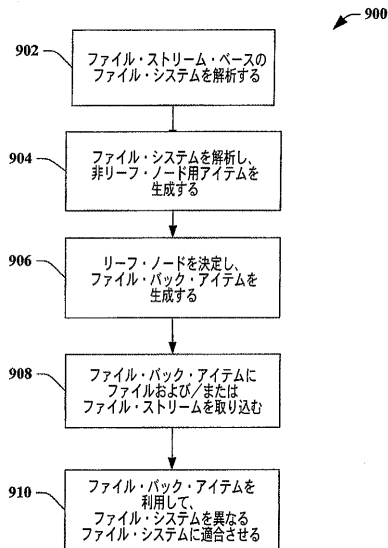
【図 7】



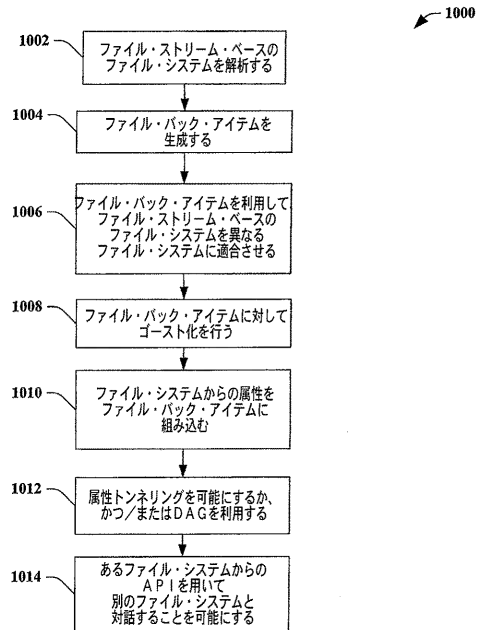
【図 8】



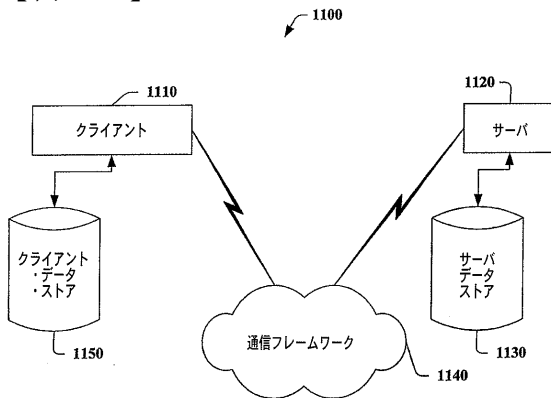
【図 9】



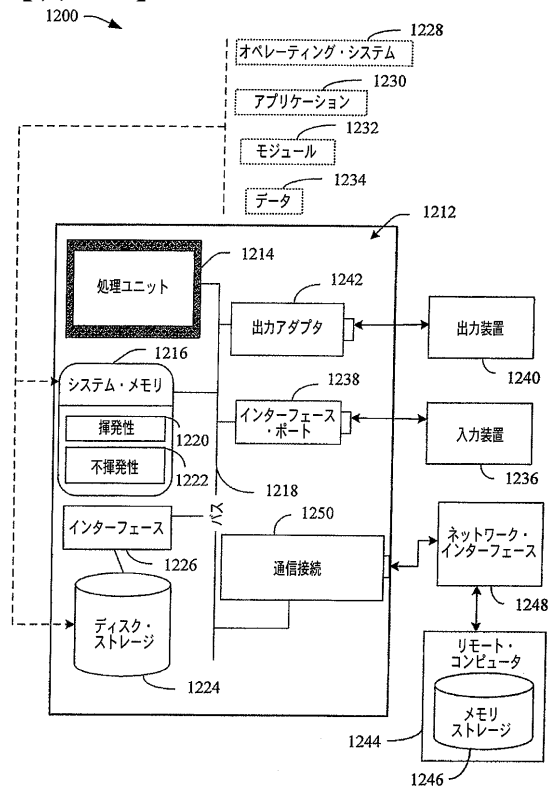
【図 10】



【図 11】



【図 12】



フロントページの続き

(72)発明者 ジェイソン ティー. ハンター

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 スリニヴァスムルジー ピー. アチャリヤ

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 CC08

5B082 GA02