(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5175284号
(P5175284)
(45)発行日　平成25年4月3日(2013.4.3)　(24)登録日　平成25年1月11日(2013.1.11)

(51)Int.Cl.　　F I
G06F 17/21 (2006.01)　　G06F 17/21 548A
G06Q 10/10 (2012.01)　　G06Q 10/10 130H
G06F 12/00 (2006.01)　　G06F 12/00 517

請求項の数 17　(全 27 頁)

(21)出願番号　特願2009-522320 (P2009-522320)
(86)(22)出願日　平成19年7月30日 (2007.7.30)
(65)公表番号　特表2009-545793 (P2009-545793A)
(43)公表日　平成21年12月24日 (2009.12.24)
(86)国際出願番号　PCT/GB2007/002864
(87)国際公開番号　WO2008/015395
(87)国際公開日　平成20年2月7日 (2008.2.7)
審査請求日　平成22年7月29日 (2010.7.29)
(31)優先権主張番号　11/461,087
(32)優先日　平成18年7月31日 (2006.7.31)
(33)優先権主張国　米国(US)

(73)特許権者　506277306
クラスター　セブン　リミテッド
イギリス　イー１１　２アールジェイ　ロンドン　ウォンステッド　ハイ　ストリート　３４－４０
(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　アンドリュー　リーブス
イギリス　シーティー２１　４エービー　ケント　ハイズ　ソルトウッド　ブロックヒル　ロード　６

最終頁に続く

(54)【発明の名称】スプレッドシートおよびその他の文書の記憶および処理

(57)【特許請求の範囲】
【請求項１】
１つまたは複数のスプレッドシート内のデータを監視および編集するシステムであって、
スプレッドシート内の１つまたは複数のエンティティを選択するエンティティセレクタと、
２つの時点間で前記選択されたエンティティの位置のシフトを追跡するトラッカと、
前記トラッカから受信された情報から、前記スプレッドシート内のエンティティの位置のシフトを表す１つまたは複数のオフセット値を導出するように構成されるオフセット判断器と、
前記スプレッドシートデータを異なる時点における前記スプレッドシートデータのバージョンと比較する前に、前記スプレッドシートデータに前記オフセット値を適用するオフセットアプリケータと
を備えることを特徴とするシステム。
【請求項２】
前記トラッカは、前記２つの時点における前記選択されたエンティティの位置を表すデータを記憶するように構成され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】
前記オフセット判断器は、前記トラッカから受信された情報を使用して前記２つの時点間における前記選択されたエンティティの位置を判断し、前記２つの時点間における前記

選択されたエンティティの位置の差異を計算することにより前記選択されたエンティティのためのオフセット値を導出するように構成されることを特徴とする、請求項２に記載のシステム。

【請求項４】

前記１つまたは複数のエンティティは、１つまたは複数の列内のエンティティを含むことを特徴とする、請求項１に記載のシステム。

【請求項５】

前記１つまたは複数のエンティティは、１つまたは複数の行内のエンティティを含むことを特徴とする、請求項１に記載のシステム。

【請求項６】

エンティティが選択された行または列のうちの１つで作成されたか否かを判断するエンティティ作成判断器をさらに備えることを特徴とする、請求項２または３に記載のシステム。

【請求項７】

エンティティが選択された行または列のうちの１つから削除されたか否かを判断するエンティティ削除判断器をさらに備えることを特徴とする、請求項２または３に記載のシステム。

【請求項８】

選択された行または列のうちの１つのエンティティが、選択された行または列の別のエンティティの複製であるか否かを判断するエンティティ複製判断器をさらに備えることを特徴とする、請求項２または３に記載のシステム。

【請求項９】

１つまたは複数のスプレッドシート内のデータを監視および編集するためのコンピュータにより実行される方法であって、

スプレッドシート内の１つまたは複数のエンティティを選択するステップと、

２つの時点間で前記選択されたエンティティの位置のシフトを追跡するステップと、

前記選択されたエンティティの位置のシフトから、前記スプレッドシート内のエンティティの位置のシフトを表す１つまたは複数のオフセット値を導出するステップと、

前記スプレッドシートデータを異なる時点における前記スプレッドシートデータのバージョンと比較する前に、前記スプレッドシートデータに前記オフセット値を適用するステップと

を含むことを特徴とするコンピュータにより実行される方法。

【請求項１０】

前記２つの時点における前記選択されたエンティティの位置を表すデータを記憶するステップをさらに含むことを特徴とする、請求項９に記載のコンピュータにより実行される方法。

【請求項１１】

前記２つの時点間における選択されたエンティティの位置を判断するステップと、

前記２つの時点間の前記選択されたエンティティの位置の差異を計算することにより、前記選択されたエンティティのためのオフセット値を導出するステップと

をさらに含むことを特徴とする、請求項１０に記載のコンピュータにより実行される方法。

【請求項１２】

前記１つまたは複数のエンティティは、１つまたは複数の列内のエンティティを含むことを特徴とする、請求項９に記載のコンピュータにより実行される方法。

【請求項１３】

前記１つまたは複数のエンティティは、１つまたは複数の行内のエンティティを含むことを特徴とする、請求項９に記載のコンピュータにより実行される方法。

【請求項１４】

エンティティが選択された行または列のうちの１つで作成されたか否かを判断するステ

ップをさらに含むことを特徴とする、請求項１２または１３に記載のコンピュータにより実行される方法。
【請求項１５】
エンティティが選択された行または列のうちの１つから削除されたか否かを判断するステップをさらに含むことを特徴とする、請求項１２または１３に記載のコンピュータにより実行される方法。
【請求項１６】
選択された行または列のうちの１つのエンティティが、選択された行または列の別のエンティティの複製であるか否かを判断するステップをさらに含むことを特徴とする、請求項１２または１３に記載のコンピュータにより実行される方法。
【請求項１７】
コンピュータを、１つまたは複数のスプレッドシート内のデータを監視および編集する手段として機能させるためのプログラムであって、前記プログラムはコンピュータを、
スプレッドシート内の１つまたは複数のエンティティを選択する手段と、
２つの時点間で前記選択されたエンティティの位置のシフトを追跡する手段と、
前記選択されたエンティティの位置のシフトから、前記スプレッドシート内のエンティティの位置のシフトを表す１つまたは複数のオフセット値を導出する手段と、
前記スプレッドシートデータを異なる時点における前記スプレッドシートデータのバージョンと比較する前に、前記スプレッドシートデータに前記オフセット値を適用する手段と
として機能させることを特徴とする、プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、スプレッドシートおよびその他の文書のデータやコードの変更を監査(auditing：検査)し追跡することと、スプレッドシートおよびその他の文書の圧縮とを含む、スプレッドシートおよびその他の文書の記憶および処理に関する。
【背景技術】
【０００２】
スプレッドシートアプリケーションは、ユーザが２次元配列および３次元配列のデータを操作できるようにするコンピュータプログラムである。スプレッドシートアプリケーションのユーザは、各々が数値データ、文章、ソフトウェアオブジェクト、または数式などのエンティティを記憶することが可能な２次元配列のセルが提供される。３次元スプレッドシートは、一定の順序を有するそのような配列を数個有する。セルが数式を含む場合には、ディスプレイは通常、数式が定義する計算結果を示す。数式は、セルの中にそのスプレッドシートの別のセル内または別のスプレッドシート内の値である入力変数を含むことができる。当然、他のセルの値は、他のセル内の数式の計算結果である場合がある。このようなセル間の関連付けを任意の複雑度まで拡張することができる。
【０００３】
スプレッドシートは、個人データにおいて迅速な計算を実行するのに使用されるツールと考えられていた。その一方、組織の運営に不可欠であるデータや機能性は、従来は、データで特定の管理された動作を行うプログラマにより開発されたコンピュータアプリケーションをサポートするデータベース内に記憶されてきた。コンピュータのスプレッドシートは、パーソナルコンピュータの成長の結果、多くの組織においてユビキタスなものとなってきた。したがって、実際には、多くの組織では、データを管理するこの従来の形式をとらなくなった。プログラムされたデータベースに欠けている計算を実行するのに慌しく一緒に入力されたスプレッドシートが、クリエータの仕事の中核をなし、また効率的にビジネスの運営の中核をなす不可欠なツールとなってきた。
【０００４】
現状において、スプレッドシートをユーザにとって魅力なものにしている特徴は、全体

として、編成の観点からすれば厄介なものになる。しかし、スプレッドシート内のセルの相互接続される性質は、このような変更の結果が予測されにくいということを意味する可能性がある。このことは、特に、スプレッドシートが別のスプレッドシートによって、場合によっては別の人によって参照される場合に言える。また、いかなる時点においてもスプレッドシートのコンテンツを判断するのは可能であるとは言えないので、スプレッドシートのコンテンツから導き出された結論の理由が追跡可能でない場合がある。さらに、現代のパーソナルコンピュータの力は、例えば、スプレッドアプリケーションに任意の複雑度の計算や動作を実行するスクリプト言語の実装を実用可能とするほどである。従来のプログラミング言語において、まだこのコードは、プログラマには精通されたバージョン制御や変更追跡を受けにくい。

【発明の概要】

【発明が解決しようとする課題】

【０００５】

明らかに、この状況は大規模の組織には受け入れられない。それは、この状況がデータや機能性の品質制御を不可能にし、ビジネスにおいて財務不正の抑止力として必要である監査可能なビジネス活動（例えば、決算書の作成）の構築の障害となる。

【０００６】

商業アプリケーションでは、多くのスプレッドシートが必要であり、各スプレッドシートのサイズは大きくなる場合が多い。このことで、全てのスプレッドシートを記憶するのにかなりの量のメモリが必要となる。典型的には、スプレッドシートのセルの７０％が数式を含む。数式は、通常、数式から導出された結果値の文字数を超えることが多いサイズの文字列として記憶される。これらの理由のために、スプレッドシート内の数式を記憶するのに使用されるスペースは、使用される全体のスペースのかなりの割合を示す可能性がある。例えば、スプレッドシート内で使用されるスペースの最大で９０％までが数式を記憶するのに使用される場合がある。さらに、典型的には、スプレッドシート内の非常に高い割合の数式が複製される。例えば、数式を列の最上部で作成し、次にそれを下方にドラッグして下部のセルに数式を貼り付けることは、数式の使用には共通のシナリオである。場合によっては、スプレッドシート内の数式の最大で９９．５％までが複製される。

【０００７】

本発明の目的は、任意の特定の時点でスプレッドシートの状態が回復され、状態間の変化が問い合わせできるように、スプレッドシート上で実行される動作が記録される機構を提供することである。

【０００８】

本発明のさらなる目的は、行や列の挿入や削除などの動作やスプレッドシート内のかなりの数のエンティティをシフトさせるソート動作を考慮するように、スプレッドシート内の変更を判断することである。

【０００９】

本発明のさらなる目的は、スプレッドシート内に生じる数式などのエンティティをより効率的な方法で記憶できる機構を提供することである。

【００１０】

さらに、スプレッドシートを非常に魅力的なものにする固有の柔軟性が失われないことが本発明の目的である。

【課題を解決するための手段】

【００１１】

本発明は、独立請求項内で定義される。好ましい特徴は、付属請求項内で示される。

【００１２】

第１の局面から、本発明は１つまたは複数のスプレッドシート内のデータを監視(monitoring)および監査するシステムを提供する。該システムは、内部にスプレッドシートデータを含むファイルを記憶することができるファイルストアと、データベースと、監視手段とを備え、監視手段は、ファイルストア内のスプレッドシートデータにおける変更を検出

し、データベース内に変更を記録するよう機能する。

【００１３】

データベース内に適切なデータを記憶することにより、個々のスプレッドシートに対して行われた全ての変更の完全な履歴を構築することができる。

【００１４】

ファイルストアは、本発明のさまざまな実施形態において、さまざまな形式で構成され、多くのさまざまなコンポーネントを含むことができる。例えば、ファイルストアは、１つまたは複数のファイルサーバおよびワークステーションコンピュータを含むことができる。ファイルストアのコンポーネントは、多様な位置に存在してもよく、ローカルエリアまたは広域エリアのネットワークリンクにより相互接続されてもよい。同様に、監視手段は、ローカルエリアまたは広域エリアのネットワークリンクを使用して、ファイルストアにアクセスしてもよい。監視手段は、最も好適には、監視されるスプレッドシートのファイルストアを検索するファイル位置モジュールを含む。

【００１５】

典型的な実施形態では、監視手段は、ファイルストア内のファイル（現ファイル）をそのファイルの前バージョンと比較する比較手段を含む。これにより、システムは同じファイルの成功のバージョンと保存したバージョンとの差異を判断することができる。より詳細には、比較手段は、現ファイル内の各セルを前バージョンの対応するセルと比較することができ、セルのコンテンツが変更されたと判断される場合には、その変更の記録を作成することができる。したがって、変更はセル単位に基づいて記録することができる。このような記録は、スプレッドシートファイルに対して行われた変更の性質の本質を理解するために、検査することができる。変更の性質は、追加されたデータ、変更されたデータ、削除されたデータ、追加された関数、変更された関数、削除された関数、関数に対するデータ、データに対する関数、セルエラー、関数再計算、スクリプト変更、外部参照変更、名前が付けられた範囲変更、リンク変更、パスワード変更、追加されたシート、削除されたシート、名前が変更されたシート、名前が変更されたスプレッドシート、削除されたスプレッドシート、追加されたシートのうちの１つまたは複数として分類してもよい。全ての実施形態がこれらの分類の全てを含むことができるとは限らないこと、いくつかの実施形態はさらにここに挙げられていない追加の分類を含むことができることは理解されるであろう。記録は、変更がなされた時を示すタイムスタンプや変更を行ったユーザのアイデンティティなどの追加のデータを含んでもよい。

【００１６】

バイナリフォーマットで記憶されたファイルが本質的に互いに比較しにくいことは十分に認識されている。したがって、本発明の特に効率的な実施形態では、監視されるファイルが解析されて、例えば、テキストベースフォーマット（ＸＭＬなど）のより簡単に処理可能な形式に変換される。さらに、ＸＭＬには、ファイルを処理するのに利用可能な多くのツールがある、またファイルをテキストベースの直列化形式および階層形式の両方で表示することができるという利点がある。

【００１７】

スプレッドシートアプリケーションは、計算ツールとしての進歩と平行して、さらに高性能なスクリプト機能を組み込んだ。このことにより、スプレッドシートアプリケーションが今では高性能のプログラミング環境と見なされるまでに進化した。例えば、Ｍｉｃｒｏｓｏｆｔ　Ｅｘｃｅｌの場合、ＶＢＡ（Ｖｉｓｕａｌ　Ｂａｓｉｃ　ｆｏｒ　Ａｐｐｌｉｃａｔｉｏｎｓ）のプログラミング言語は、高い複雑度のプログラムを実装することが可能であるスクリプト言語として提供されている。スプレッドシートデータと同様に、このことによりユーザがプログラムを迅速にかつ柔軟に作成することができる。しかし、この柔軟性は、ソフトウェアの展開全体にわたって制御がほとんどなされないということを意味する。特に、バージョン制御システムまたはソースコード制御システムがないので、イベント後になぜプログラムコードに対して変更が行われたのかを見ることができない。

【００１８】

本発明のさらなる目的は、ユーザに柔軟性をもたせることに限定しないで、スプレッドシートなどの文書内に含まれるコードを管理するための手段を提供することである。

【００１９】

別の局面から、本発明は１つまたは複数の文書内のプログラムコードを監視および監査するためのシステムを提供する。該システムは、内部に文書を含むファイルを記憶できるファイルストアと、データベースと、監視手段とを備え、監視手段は、ファイルストア内の文書に含まれるコードの変更を検出し、データベース内にその変更を記録するよう機能する。

【００２０】

スプレッドシートファイルに関する実施形態と同様に、データベースは、ある期間にわたってコードの進行や変化を監視および監査できるように、コードに対して行われた変更履歴を含む。

【００２１】

本発明のこの局面の実施形態は、コードを含むさまざまなタイプの文書に関して機能することができる。例えば、これらの文書は、スプレッドシート、ワードプロセッサ文書、データベースファイルなどを含んでもよい。

【００２２】

多くの場合、本発明のこれら２つの局面は両方とも、共通のシステムにより提供されるだろう。

【００２３】

別の局面から、本発明は、本発明の第１の局面または第２の局面の実施形態の監視手段を構成するために、コンピュータハードウェア上で実行可能なコンピュータプログラム製品を提供する。

【００２４】

さらに別の局面から、本発明は、１つまたは複数のスプレッドシート内のデータを監視および監査するためにネットワークへの接続のためのサーバを提供する。該ネットワークは、内部にスプレッドシートデータを含むファイルを記憶できるファイルストアと、データベースと、ファイルストア内のスプレッドシートデータにおける変更を検出し、データベース内にその変更を記録するように機能するサーバとを含む。

【００２５】

該サーバは、本発明の第１の局面における監視手段を構成してもよい。

【００２６】

さらなる局面では、本発明は、スプレッドシートを処理し、記憶するためのシステム、方法、コンピュータプログラム製品を提供する。該システムは、スプレッドシート内に生じる各固有の数式を判断するように構成された判断ユニットと、各固有の数式を記憶するための第１のストアと、各固有の数式に対する固有の識別表示を記憶するための第２のストアと、スプレッドシート内に生じる各数式をその対応する数式識別表示と置換するように構成された置換ユニットとを備える。

【００２７】

スプレッドシート内に生じる各数式を記憶するのではなく、固有の数式のみが記憶される。数式は、スプレッドシート内に生じる各々の固有の数式のためのエントリを備える数式テーブル内に記憶される。数式テーブルはさらに、各数式に対する固有の数式ＩＤ（識別表示）を記憶する。このように、スプレッドシート内に生じる数式は、それらの対応するＩＤと置換することができる。スプレッドシート内の数式の各々のインスタンスではなく、各固有の数式のみが記憶されるので、このことが冗長性を取り除き、スプレッドシート内のスペースの大幅な節約につながる。さらに、数式ＩＤが文字列として数式を記憶するのに使用されるスペースの一部分を占めることで、さらにスペースの節約につながる。

【００２８】

さらなる局面では、本発明は、１つまたは複数のスプレッドシート内のデータを監視および監査するためのシステム、方法、およびコンピュータプログラム製品を提供する。該

システムは、スプレッドシート内の１つまたは複数のエンティティを選択するためのエンティティセレクタと、２つの時点間の選択されたエンティティの位置のシフトを追跡するためのトラッカと、トラッカから受信された情報から、スプレッドシート内のエンティティの位置のシフトを表す１つまたは複数のオフセット値を導出するように構成されるオフセット判断器と、スプレッドシートデータを異なる時点におけるスプレッドシートデータのバージョンと比較する前に、オフセット値をスプレッドシートデータに適用するオフセットアプリケータとを備える。

【００２９】

行もしくは列が挿入または削除される時、または行もしくは列がソートされる時、スプレッドシートが比較される前に、エンティティのシフトにより生じるノイズが除去される。１つまたは複数の選択された行および／または１つまたは複数の選択された列内のエンティティは、スプレッドシート内のエンティティのサンプルを形成する。スプレッドシート内のエンティティのサンプルの移動は、スプレッドシート内のエンティティの全体の移動を判断するために追跡される。次に、スプレッドシートの２つのバージョンの比較に先立って、エンティティの判断された移動は、スプレッドシートの２つのバージョン間のセル単位の比較が実行される前に、スプレッドシートの前のバージョンに適用される。このように、セル単位のスプレッドシートの比較のプロセスでは、たまたま同一セルを占める値間の比較ではなく、同じようにシフトした値間の比較がなされる。

【００３０】

上述の全ての場合において、本発明の実施形態の動作は、クライアントの動作を妨げるものでないのが好ましく、またできる限り実施形態の存在がクライアントのユーザに対して完全にトランスペアレントであることが好ましい。このため、本発明の実施形態がネットワーク上のクライアントとは非同期的に動作することが非常に好ましい。特に、実施形態がファイルストア内で保存されているファイルを処理する間は、クライアントの動作は遮断されるべきではない。

【００３１】

ここで、本発明の一実施形態を一例として添付図面を参照して詳細に述べる。

【図面の簡単な説明】

【００３２】

【図１】本発明の一実施形態のハイレベル概略図である。

【図２】図１の実施形態におけるデータを維持するプロセスを示す図である。

【図３】図１の実施形態におけるファイルをキャプチャするプロセスを示す図である。

【図４】図１の実施形態のコンポーネントを示す図である。

【図５】本発明が具現化される企業内のコンピュータのネットワークの概略図である。

【図６】本発明の一実施形態の主要な機能コンポーネントを示すブロック図である。

【図７】他のタスククラスを派生するベースクラスを示す図である。

【図８】種々のタスククラスを示す図である。

【図９】ファイル検索クラスの動作を示す図である。

【図１０】変換クラスの動作を示す図である。

【図１１】ファイル検索クラスの動作を示す図である。

【図１２】マクロデコンポーザクラスの動作を示す図である。

【図１３】比較クラスの動作を示す図である。

【図１４】バルクロードクラスの動作を示す図である。

【図１５】ＸＭＬストレージクラスの動作を示す図である。

【図１６】実施形態の動作時に生成される種々の表示を示すスクリーンショットである。

【図１７】実施形態の動作時に生成される種々の表示を示すスクリーンショットである。

【図１８】実施形態の動作時に生成される種々の表示を示すスクリーンショットである。

【図１９】実施形態の動作時に生成される種々の表示を示すスクリーンショットである。

【図２０】圧縮された数式を記憶するのに使用される方法のフロー図である。

【図２１】スプレッドシート内の変更の追跡時に、行および列の挿入処理、削除処理なら

びにデータソート処理を考慮するのに使用される方法のフロー図である。
【発明を実施するための形態】
【００３３】
実施形態の概説
図１に示すように、実施形態は、既存の企業のデスクトップおよびネットワーク環境とともに機能するシステムである。動作時、本システムは、ネットワーク内のスプレッドシートファイルの存在を識別し、次に、管理業務により選択されたスプレッドシートファイルをキャプチャする。次に、本システムは、これらのファイルのコンテンツのデータベースを構築する。第１のデータベースが構築されると、本システムは管理されているファイルに生じるイベントに関してネットワークを監視する。スプレッドシートのファイル保存を含む（これに限らないが）、選択されたトリガイベント時に、本システムはファイルのコピーを検索し、それを要素部分に分解して、これをデータベース内にある原本と比較する。
【００３４】
次に、本システムは、データベース内で行われた変更を保存する。ファイルの名前が変更された場合、ファイルが削除された場合、またはファイルが移動された場合に、本システムはデータベース内にこのことを記録する。このシステムにより企業は集中してデータベース内に集められたデータを見ることが可能になる。このシステムは、既存の技術を採用するポータルを使用してデータを集約し、分析することができる。
【００３５】
図２に示すように、本システムは、この機能を実行するために複数のネットワークドメインにアクセスして、管理下にあるファイルを維持することができる。本システムは、ファイルのコンテンツをＸＭＬ（拡張マークアップ言語）の表示に変換し、このフォーマットを使用して比較を行う。本システムは、最も粒度の細かいレベルで機能して、ファイルを保存し、ファイルに対する変更を追跡し、識別し、キャプチャする。データベースは、この非常に粒度の細かいレベルでポピュレートされる。会社のユーザは、自身の報告ツールや分析選択肢をデータに適用することができる。本発明は、データの処理を限定するものではない。
【００３６】
図３に示すように、本システムは、アクセスが得られるドメイン内で機能する。本システムは、ファイル、イベントおよびイベント後のファイルのバージョンを探し出して、データベースを構築する。
【００３７】
図４に示すように、本システムは、対象のネットワーク環境をスキャンし、管理されるべき全てのファイルのリストを取り出す。本発明は、どこで直近の新規ファイルが作成されたか、またどこでファイルが削除されたか、ファイルの存在を識別する。スキャンは、自動ソフトウェアツールか、またはプロセスの開始（図１）かファイルの維持（図２）かのどちらか一方を行うためにファイルのキャプチャをトリガする業務のどちらか一方を可能にするレポートを生成する。
【００３８】
本実施形態において、行もしくは列が挿入される時、または行もしくは列がソートされる時、スプレッドシートが比較される前に、エンティティのシフトにより生じるノイズが除去される。このことを達成するために、スプレッドシート内のエンティティの全体の移動を判断するために、スプレッドシート内のエンティティのサンプルの移動が追跡される。次に、スプレッドシートの２つのバージョンの比較に先立って、スプレッドシートの２つのバージョン間のセル単位の比較が行われる前に、エンティティの判断された移動がスプレッドシートの前のバージョンに適用される。
【００３９】
本実施形態において、スプレッドシート内に生じる各数式を記憶するのではなく、固有の数式のみが記憶される。数式は、スプレッドシート内に生じる各固有の数式に対するエ

ントリを備える数式テーブル内に記憶される。数式テーブルはさらに、各数式に対する固有の数式ＩＤ（識別表示）を記憶する。こうして、スプレッドシート内に生じる数式は、それらの対応する数式ＩＤと置換することができる。スプレッドシート内の数式の各々のインスタンスではなく、各固有の数式のみが記憶されるので、このことが冗長性を取り除き、スプレッドシート内のスペースの大幅な節約につながる。さらに、数式ＩＤが文字列として数式を記憶するのに使用されるスペースの一部分を占めることで、さらにスペースの節約につながる。

【００４０】

図面の詳細な説明

図１に示すように、本発明の好適な実施形態は、例えば、既存の企業のデスクトップやネットワーク環境などの対象環境で機能する。システムの第１の機能は、対象環境内の全てのファイルを識別することである。図４で詳細に説明するように、このプロセスは、対象環境内の適切なドメインを選択し、その適切なドメインにアクセスすること、ファイルに関してネットワークをスキャンすること、業務レポートを作成することを含む。システムの次の機能は、管理の範囲を定義および設定することである。このことにより、特定のファイル、ファイルのネストもしくはドメイン、トリガイベントが管理業務により選択されるようにできる。これは、設定範囲内にある前ファイルのサブセットを含むことができる。システムの次の機能は、ネットワークを介してファイルをキャプチャすることである。このプロセスは、ネットワークにログオンされた個々のＰＣ内に位置するファイルをキャプチャすることを含む。図３でより詳細に説明するように、このプロセスは、適切なドメインを選択し、その適切なドメインにアクセスすること、種々の領域からファイルを検索すること、ファイルコンテンツをＸＭＬに変換すること、データベースをポピュレートすることを含む。システムの次の機能は、ファイルおよびファイルコンテンツのデータベースを構築することである。システムのさらなる機能は、データを集約し、分散データの集中ストアを作成することである。

【００４１】

同等に重要な機能は、図２により詳細に説明されたデータの維持である。管理下の種々のファイルは、ネットワークを介して複数の異なるＰＣ内に記憶することができるので、データの集中ソースがデータの維持を容易にする。データの維持は、管理下のファイルをキャプチャすること、ファイルコンテンツをＸＭＬに変換すること、変更を比較してキャプチャし、データベース内に変更を記憶することを含む。データを記憶して維持する時に、どのファイルが存在するか、ファイルに対してどの変更が発生するか、誰がデータを操作するのかを知ることが重要である。この情報を知るために、全てのデータの監査証跡(audit trail)およびアクティビティの履歴が構築され、データはシステムにより分析される。上述のプロセスから得られたデータを問い合わせするためのシステムの次の機能は、データに対するポータルアクセスを作成することである。このことにより、ポータルを介したデータへのアクセスが得られる。データへのアクセスは、報告ツールやファイルまたはアプリケーションを再構築する能力を提供することにより拡張できる。

【００４２】

図１のプロセスにおける記憶されたデータの維持に関する機能は、図２に示されている。第１のステップとして、ファイル保存機能などのトリガイベントが発生する。次に、図３で詳細に説明するように、管理下のファイルがキャプチャされる。管理下のファイルは、例えば、異なる領域に属するワークブックファイルを含んでもよい。いくつかのファイルは、複数の異なる位置で操作してもよい。例えば、図２に示すように、２つのワークブックは領域ａに位置するが、１つのワークブックは領域ｂに位置してもよい。データ維持の次のステップは、キャプチャされたファイルのコンテンツがＸＭＬに変換される。このフォーマットは、次の処理にとって都合の良いフォーマットである。次のステップで、発生した変更を見つけるために、キャプチャされた文書が文書の以前のバージョンと比較される。次に、これらの変更はキャプチャされ、データベースが変更を示す情報を使用してポピュレートされる。データへのポータルアクセスまたはクライアントアクセスは、ポー

タルまたはクライアントを介するデータへのアクセスが提供されるように作成される。システムにより提供される報告ツールは、ポータルアクセスを拡張させる。

【００４３】

ファイルのキャプチャに関する機能は、図３に示す。第１のステップでは、このシステムにより、適切なドメインが選択され、選択されたドメインへのアクセスが得られる。本システムは、アクセスが得られたドメイン内で機能する。次に、選択されたドメイン内のファイルの存在が識別される。このプロセスは、図４でより詳細に説明する。上述したように、ファイルは複数の位置で記憶してもよい。次のステップでは、本システムは、ファイルの全てを取って来る。次のステップでは、管理用のファイルが選択される。これらのファイルのコンテンツはＸＭＬに変換され、各ファイルの親バージョンのバイナリがキャプチャされる。次に、データベースがこのデータでポピュレートされる。

【００４４】

ネットワークにログオンされた個々のＰＣを含むネットワークを介するファイルのキャプチャに関する機能は、図４に示す。前述したように、適切なドメインが選択され、それらのドメインへのアクセスが得られる。次のプロセスでは、本システムは、監視されるべき全ての関連ファイルに対して対象ネットワーク環境をスキャンする。例えば、システムがＭｉｃｒｏｓｏｆｔ　Ｅｘｃｅｌスプレッドシート内の変更を追跡するのに使用される場合、ネットワークは全てのＥｘｃｅｌファイルに対してスキャンされる。このプロセスは、アクティビティをチェックし、作成された新規ファイルをチェックし、見つけられなかったファイルをチェックするために複数のスキャンを実行するサブプロセスを含む。新規ファイルは、最後のスキャン以降に作成されたファイルを含む。例えば、ファイルが削除された場合には、ファイルが見つからないかもしれない。このサブプロセスの結果は、見つけられたファイルおよび発生したアクティビティのレポートを構築するために使用される。ネットワークがファイルに対してスキャンされた後、業務用のレポートが作成される。レポートにより、図１に示されたプロセスの開始か図２に示されたファイルの維持かのどちらか一方のためのファイルのキャプチャのトリガが可能になる。

【００４５】

図６は、本発明の一実施形態の主要な機能コンポーネントを示すブロック図である。ファイルストア２０内に記憶されたスプレッドシート全体は、観察サービス（watching service）２２によりアクセスされる。制御サービスがさらに提供される。観察サービスおよびコントローラサービスは、メタストア２６と通信する。メタストア２６は、さらに監視サービス３２と処理サービス３４とを備える非同期フレームワークと通信する。図６の機能コンポーネントは、以下でより詳細に説明する。

【００４６】

非同期フレームワークに関連付けられたタスクは、図７でより詳細に説明するように、タスクオブジェクト３６で実装される。図７に示すように、各タスクオブジェクトは、タスク動作、タスク開始、タスク実行、タスク終了を含むさまざまな方法を公開する。図６の機能コンポーネントおよび図７の方法は、以下でより詳細に説明する。図８に示すように、タスクオブジェクト３６および公開される方法は、システム内で種々のプロセスを実行するタスクオブジェクト３６から派生した種々のクラスを引き継ぐ。このクラスには、ファイル検索オブジェクト、変換オブジェクト、マクロデコンポーザオブジェクト、バルクロードオブジェクト、ＸＭＬストレージオブジェクト、ＸＬＳストレージオブジェクト、制限チェッカーオブジェクトが含まれる。変換クラスは、比較クラスおよびマクロクラスを呼び出す。図７の方法および図８のクラスは、以下でより詳細に説明する。

【００４７】

図９～１５は、図８の種々のクラスの動作を示す図である。いずれの場合も、非同期フレームは、タスク動作方法を使用してオブジェクトを開始し、ＴＳＯ（トランザクション状態オブジェクト）をその処理内で使用するタスクの状態変数を含む入力として受信する。タスク動作方法はさらに、追跡コードへのログエントリを作成するログ追跡呼出機能を呼び出す。状態有効化動作も実行される。タスク開始方法は、起動変数設定動作において

実行設定変数をセットアップする。変数は、メタストア２６内に記憶される。タスク実行方法は、タスクを完了するための全ての機能を実行する。タスクの完了時に、タスク終了方法は、メタストア２６内の完了状態の設定を含む全ての終了動作を設定する。

【００４８】

図９に示すように、ファイル検索クラスのタスク実行方法は、監視サービス３２内に、ファイルを検索する機能、ファイルが存在するか否かをチェックする機能、ファイルを処理キューに追加する機能を含む。

【００４９】

図１０に示すように、変換クラスのタスク実行方法は、ＷＢ（ワークブック）を処理する機能を含む。この機能は、ワークブックを開く機能、ワークブックの非保護機能、ＷＳ（ワークシート）動作リストを計算する機能、ワークブックの指定範囲を作成する機能、ワークブックのマクロを分割する機能、ワークブックをＸＭＬファイルとして保存する機能を含む。タスク実行方法はさらに、ＸＭＬワークシートを分割する機能、ＸＭＬヘッダを処理する機能、各ワークシートをフレームワークに追加する機能を含む。この最後の機能では、異なる時点でスプレッドシートを比較するための比較タスクを提起するトランザクション状態オブジェクトが非同期フレームワークに対して発行される。非同期フレームワーク結果を取得する機能では、スプレッドシート内の変更を表す「デルタ」のＸＭＬファイル名を含む比較タスクの結果が、非同期フレームワークから戻される。各マクロは、トランザクション状態オブジェクトをマクロデコンポーザに対して発行することにより処理され、バルクロードがトランザクション状態オブジェクトをバルクロードクラスに対して発行することにより実行される。

【００５０】

図１１に示すように、ファイル検索クラスのタスク実行方法は、ファイルを検索する機能、ファイルが存在するか否かを（トランザクション状態オブジェクトを介して）チェックする機能、ファイルを非同期フレームワーク３０に追加する機能を含む。

【００５１】

図１２に示すように、マクロデコンポーザのタスク実行方法は、スクリプトのチェックサムが異なるか否かをチェックする機能を含む。トランザクション状態オブジェクトは、メタストア２６およびソースコード制御システム４０内にファイルを記憶するこの機能により発行される。非同期フレームワーク内のマクロの制限を要求する機能が提供される。

【００５２】

図１３に示すように、比較クラスのタスク実行方法は、メタストア２６からワークシートの時点バージョンを取得する機能を含む。さらに、チェックサムをチェックする機能、ワークシートを比較する機能、処理概要を設定する機能、比較から得たデルタファイルの位置を戻す機能を提供する。

【００５３】

図１４に示すように、バルクローダクラスのタスク実行方法は、スプレッドシート比較から生成された各デルタファイルを非同期フレームワーク３０に追加する機能を含む。ローダ機能は、データをメタストア２６に提供する。１つの機能は非同期フレームワークからの結果を待つ。

【００５４】

図１５に示すように、ＸＭＬストレージクラスのタスク実行方法は、ＸＭＬファイルを圧縮する機能を含む。圧縮ＸＭＬファイルは、ファイル記憶機能によりメタストア２６内に記憶される。メタストアから以前のファイルを削除する機能も実行される。

【００５５】

より詳細な実施形態

本発明のこの実施形態は、コンピュータのネットワーク内で実施される。このネットワークでは、任意に多数のクライアントがある。クライアントは、ユーザがスプレッドシートを展開および使用するためのスプレッドシートアプリケーションを実行するパーソナルコンピュータである。ユーザはまた、自分のパーソナルスプレッドシートファイルをこれ

らのコンピュータに記憶する場合もある。ファイルはネットワークにリンクされた任意のドライブで監視することができる。通常は、必ずしもとは限らないが、重要なスプレッドシートは中央ファイルサーバに位置する。さらに、ネットワーク上に監視サーバがあり、監視サーバはスプレッドシート上でユーザにより実行される動作を監視し、記録する。

【００５６】

図５に示すように、企業内のコンピュータのネットワークは、任意に多数のクライアントコンピュータ１０を備える。各クライアントコンピュータ１０は、ユーザによりスプレッドシートを含む個人的な文書を作成するのに使用される。典型的には、各クライアントコンピュータ１０は、個人的な嗜好や業務慣例をスプレッドシートに関連付ける１人のユーザを有する。

【００５７】

各クライアントコンピュータ１０は、ネットワーク１２に接続される。説明のために、これは統合ネットワークとして図１に示されている。しかし、実際の実施形態では、多くのサブセットやローカルエリアおよび広域エリアネットワークリンクを含むことが多い。

【００５８】

ネットワーク１２に接続されるのは、さらに１つまたは複数のファイルサーバ１４である。ネットワークポリシーに応じて、ユーザは自分のクライアントマシン１０かファイルサーバ１４かのいずれかにファイルを記憶することを選択することができる。各クライアントコンピュータ１０とファイルサーバ１４または各ファイルサーバ１４とにより提供されるローカルストレージは多様なファイルストアを構成する。

【００５９】

監視サーバ１６は、オペレータの方針に従って、ファイルストア全体またはファイルストアの一部にアクセスするようにネットワーク１２に接続される。本発明により提供される処理が実行されるのは、監視サーバ１６内である。監視サーバ１６の構成は、全体としてネットワークの性質によって異なる。監視サーバ１６は、クライアント１０上で実行されるソフトウェアの動作時に最小の遅延でタスクを実行することが可能でなければならない。時には、監視サーバ１６はモノリシックマシンである場合があるが、他の場合には、複数の文書の並行処理を可能にするためにサーバ４４により制御されるマシンのクラスタである場合がある。これらについては、以下で詳細に説明する。

【００６０】

図６では、ファイルストア内の全体のスプレッドシートファイルが２０で示されている。図６に示されたその他のアイテムは、監視サーバ１６の論理コンポーネントである。これらのコンポーネントは、観察サービス２２、コントローラサービス２４、メタストアデータベース２６、監視サービス３２および処理サービス３４を含む非同期フレームワーク３０である。全体として、システムの動作は、種々の論理コンポーネント間でタスクメッセージを渡すことにより制御される。このことは、クラスタにさらなるマシンを追加して、マシン間でタスクメッセージを配信することによりシステムを拡大することができる利点がある。さらに、クライアントとは非同期的に監視サーバがスプレッドシートファイル２０を処理できるようになり、これにより確実にユーザのスプレッドシートプログラムの動作を妨げるのを最小にできる。

【００６１】

観察サービス２２の機能は、スプレッドシートファイル２０を監視することおよびスプレッドシートファイルのいずれかが変更した時にイベントを提起することである。変更の検出時に、ファイルは読み出され、次に、最後に読み出された時のファイルのコンテンツと比較された現コンテンツと識別された変更とがデータベース内に記憶される。このようにして、ファイルストア内のファイルが変更されるときはいつでも記録が行われる。したがって、ファイルストア内のそのままのファイルで開始する際に、記録された変更を逆に適用し、そのことでファイルの展開における早い段階の状態のファイルにたどり着くことが可能である。観察サービス２２を提供することは、ファイルが変更される時に動作をトリガするためにスプレッドシートプログラムにフックする必要性を除去する。

【００６２】
概して、この実施形態はＭｉｃｒｏｓｏｆｔ　Ｅｘｃｅｌを使用して作成されたスプレッドシートの処理に関する。このアプリケーションは、従来はファイル名拡張子「ＸＬＳ」（便宜上、「ＸＬＳファイル」と呼ぶ）を有するファイル内のスプレッドシートを記憶する。これらはバイナリファイルである。しかし、本発明の原理は、他のアプリケーションがスプレッドシートデータを記憶する方法に関係なく、他のアプリケーションを使用して作成されたスプレッドシートにも同等に適用できる。バイナリファイルにおいてコンテンツ分析を実行するのが難しいのは周知であるので、この実施形態がとるアプローチは、まずＸＬＳファイルをＸＭＬ（拡張マークアップ言語）の表示に変換することである。他のスプレッドシートプログラム、特にＯｐｅｎＯｆｆｉｃｅ．ｏｒｇのＣａｌｃはネイティブファイルフォーマットとしてＸＭＬを使用する。したがって、これらのファイルを処理する時に、変換ステップは省略できる。
【００６３】
上述したように、この実施形態はコンピュータのクラスタ上に実装される。サーバにより行われる動作は個別タスクとして実行される。タスクオブジェクトが作成されて、タスクキュー内に配置される。タスクがキューのヘッドに達すると、クラスタ内の第１の利用可能なマシンにより処理される。
【００６４】
図３は、全てのタスクを派生する親クラスの構造を示す図である。４つの主要な方法がタスクオブジェクトごとに公開される。これらは、タスク動作方法、タスク開始方法、タスク実行方法、タスク終了方法である。タスク動作方法は、非同期フレームワークがオブジェクトを起動する方法である。タスク動作方法はさらに追跡コードへのログエントリを作成する。タスク動作方法は、処理内で使用するタスクの状態変数を含む入力ＴＳＯ（トランザクション状態オブジェクト）として受け取る。タスク開始方法は、実行設定変数をセットアップする。タスク実行方法は、タスクを完了する全ての機能を実行する。タスク終了方法は、全ての終了動作を設定する。
【００６５】
図３に示されたクラスから派生される主要なクラスは、図４に示されている。これらは、ファイルストアからファイルを検索するタスクオブジェクトを含み、ファイルコンテンツ、処理マクロ、データベース内の取扱いストレージ、取扱い変更情報を変換し、比較する。ここで、これらのクラスを簡潔に示す。
【００６６】
ファイル検索：このクラスは対象の変更された位置からファイルをコピーする。
【００６７】
変換：このクラスは、ＸＬＳファイルを受け取り、ＸＭＬワークシートに分解する。変換クラスは、比較クラスおよびマクロクラスを呼び出して、全てのデルタ動作を実行する。このクラスは、全ての指定範囲を分解して、いずれの変更も検出する。このクラスはさらに、非同期フレームワークに比較タスク、マクロタスク、デコンポーザタスクをもたらす。
【００６８】
比較：このクラスは、最新の状態と、ＸＭＬの２つの時点間の「デルタ」として既知である変更の記述とを計算し、分類する。このクラスは、全ての変更のタイプおよびセルのタイプを分類し、各デルタ変更を関連変更でタグ付けする。このクラスは、以下のアルゴリズムを使用する。
・以前のＸＭＬファイルと現ファイルのＸＭＬファイルから第１のセルのポインタを位置決めする。
・２つのセルのセル行列（行位置、列位置）を比較する。現セル位置と前セル位置が同じである場合、セルの値と数式を比較して差異をチェックする。比較は、変更のタイプを識別し、分類することになる。セルが数式を含む場合、セルは「数式」のタイプでタグ付けされて、不変の数値を含むセルと、表示した時に数式計算の結果として同じ値を示すセル

とをはっきりと区別する。さらに、セル変更の差異が定められる。整数値に対しては、差異は２つの整数の差異である。文字列の差異は、各文字列の長さに基づく。データの差異は、日数で計算される。現セル位置が前セル位置よりも大きい場合、「追加データ」記録をデルタＸＭＬファイルに追加する。前セル位置が現セル位置よりも大きい場合、「削除データ」記録をデルタＸＭＬファイルに追加する。
・ファイルからセル行列位置が最小である次のセルを取得し、セル行列位置を比較することにより再び比較プロセスを開始する。
・前ＸＭＬファイルと現ＸＭＬファイルがファイルの最後まで読み出されるまでこのプロセスを繰り返す。

【００６９】
完了したプロセスの結果、デルタＸＭＬファイルが得られる。これは、現時点と前時点との間のカテゴリ化された差異を含む。

【００７０】
バルクロード：このクラスは、変換／比較タスクから作成されたデルタファイルをバルク挿入する。

【００７１】
ＸＭＬストレージ：このクラスは、文書の変換から作成されたＸＭＬファイルを記憶する。このクラスは、生成された現ＸＭＬファイルを圧縮し、前ファイルを削除し、新規ファイルをデータベースにアップロードする働きをする。

【００７２】
マクロデコンポーザ：このクラスは、２つの時点の２つのプログラマチックスクリプト間の差異を計算する。キャプチャスクリプトは、既存のソースコード制御システム４０に組み込まれる。

【００７３】
制限チェッカー：このクラスは、データストア内で生じた変更タイプを監視し、指定された対象にアラートを提供する。

【００７４】
実施形態がＷｅｂベースのインタフェースを提供する場合、ユーザはこのインタフェースを介してデータベース内のデータへのアクセスを得られる。このことで、全てのスプレッドシートのアクティビティが監視される中心部が得られる。監視サーバ１６で実行するプロセスは、データベースに適用されるクエリを生成し、これらのクエリの結果からＷｅｂサーバ３８によりＷｅｂブラウザに供給されるページを生成する。生成されるクエリやレポートの範囲は、基本的に制限されない。これらは、システムの特定のユーザの固有の要件に合わせて調整される。該インタフェースはさらに、システムの構成や維持にも使用される。Ｗｅｂベースインタフェースをデータベースに提供することは、決まりきったことである。したがって、本明細書ではこれ以上説明しない。

【００７５】
さらに、クエリの結果に基づいてアラートを生成することができる。例えば、管理者は、特定のスプレッドシートのいくらかの部分の禁止行為の範囲を規定してもよいし、特定のセルの値が所定の範囲の外に移動する時に警告してもらえるよう要求してもよい。本システムは、アラートが出されたと検出する時に、電子メールを生成して所定のアドレスに添付する場合もある。

【００７６】
他のＷｅｂベースインタフェースにより提供されるメイン画面のいくつかを、ここで簡潔に説明する。

【００７７】
図１６は、ユーザがどのファイルが監視されているか、ファイルがどこへリンクされるか、リンクされたファイルの階層を見ることができる画面である。

【００７８】
図１７は、ユーザがスプレッドシート内の特定のセルの値がどのように時間と共に変化

するかを監視することができる画面である。値はこの画面ではグラフで示されている。

【００７９】

図１８は、ユーザにスプレッドシートに対して行われた変更または時間ごとの変更の概要を示す画面である。各変更の性質やその変更を行った人が識別される。

【００８０】

図１９は、本発明のソースコード制御の局面を示す図である。図１９では、右のペインが現バージョンのプログラムコードのセグメントを示し、左のペインがコードの前バージョンを示す。

【００８１】

行または列の挿入、削除、シフト

上述したように、典型的な実施形態では、比較手段は、ファイルストア内の現ファイルをファイルの前バージョンと比較する働きをする。現ファイル内の各セルは前バージョンの対応セルと比較され、スプレッドシート内の変更がセル単位で記録される。しかし、スプレッドシート編集の種々の一般的な動作は多数のエンティティ（値や数式）をスプレッドシート内の位置にシフトさせる。例えば、行が挿入または削除された時、この動作は、挿入された行または削除された行の下の全ての値や数式を１セルごとに上下にシフトさせる。同様に、列の挿入または削除は、挿入された列または削除された列の右のセルの全ての値や数式を１セルごとに左右にシフトさせる。別の例では、行または列ごとに値をソートすることは行の値または列の値を多種量ごとに並べ替えして、シフトさせる。

【００８２】

これらのタイプの動作と共に、スプレッドシートの現バージョンと前バージョンとのセル単位の差異を分析することで、時には多くのエンティティのシフトにより生じる大きなノイズを招く。比較により得られた典型的なデルタファイルは、多くの変更を特定セルの値の変更として示すが、これらの変更をエンティティのシフトによる変更として認識するのにより適している。したがって、スプレッドシート内の変更を分析する時に、ある変更は一回の動作により引き起こされたエンティティのシフトのために生じたという事実をハイライトすることが好ましい。このことで、スプレッドシートの変更のノイズがフィルタリングされる。

【００８３】

複数のセル内の変更が一回の動作により生じたか否かを判断するために、セル単位の比較スキームを維持しながら、図２１のフロー図を参照した以下の方法を採ってもよい。まず、スプレッドシート内のエンティティのサンプルの移動が、スプレッドシート内の全体的なエンティティの移動を判断するために追跡される。次に、スプレッドシートの２つのバージョンの比較に先立って、判断されたエンティティの移動が、スプレッドシートの１つのバージョン間のセル単位の比較の前にスプレッドシートの前バージョンに適用される。好適な実施形態では、移動が追跡されるエンティティのサンプルは、１つまたは複数の行および／または１つまたは複数の列のエンティティで形成される。これらのエンティティは、「キー」と呼ぶ場合がある。スプレッドシートの２つのバージョンの各キーの位置を判断することにより、２つのバージョン間の各キーの移動が判断できる。

【００８４】

明らかであるように、１つまたは複数の列を選択して、行の挿入もしくは削除、または垂直方向のセルのソートにより生じたエンティティの移動を検出することができる。１つまたは複数の行を選択して、列の挿入もしくは削除、または水平方向のセルのソートを検出することができる。以下に説明する例では、１つまたは複数の列が選択される。しかし、行に関して対応するプロセスは、代替例として、または列の選択と組み合わせて簡単に適用できる。例示的な方法の第１のステップでは、１つまたは複数の列を選択して、選択した列にエンティティを備える１つまたは複数のキーを定義する。

【００８５】

好適には、列は、こうして得られた各キーが固有値などの固有エンティティであるように選択される。２つのキーが同じである場合、スプレッドシートの１つのバージョンの２

つのキーのいずれがスプレッドシートの先のバージョンのキーのうちの固有のキーに対応するかを確実に判断することができない。したがって、このような複製されたキーの移動は確実に判断できない。多くのスプレッドシートでは、１つまたは複数の列が互いに全て異なる値を含む。しかし、エンティティの各々が固有である列がない場合、２つ以上の列を使用して、固有のキーのセットを定義してもよい。例えば、２つ以上の特定の行のエンティティを連結して１つのキーを形成してもよい。このプロセスは、他の行の各々に対しても繰り返されて、キーのセットを形成する。列の任意の行のエンティティが列の他の行のエンティティとは異なるセットを形成するように列のセットが選択される場合、このプロセスはキーの固有のセットを生成することがわかるであろう。

【００８６】

キーのセットを定義する１つまたは複数の列の選択は、エンティティセレクタを使用してシステムにより自動的に、またはユーザにより手動で行ってもよい。例えば、手動選択の場合、本システムは、ユーザが１つまたは複数の列をハイライトして、これによりこれらの列のエンティティをキーとして選択する構成である。好適な実施形態では、本システムは範囲マネージャ画面を提供し、これによりユーザは興味のある指定範囲を選択し、その後、指定範囲内のエンティティの一意性を定義する列または列の組み合わせを選ぶことができる。自動選択の場合、本システムは、分析される特定のスプレッドシートに基づいて最も適切な列を選択する構成である。

【００８７】

キーの選択は、最初に特定のスプレッドシート内の変更を追跡することが望まれると判断されると行われる。スプレッドシート内の変更を追跡した後の処理を説明する。

【００８８】

システムによるスプレッドシートの処理の間、各時点において、選択された列（単数または複数）内のエンティティをキャプチャし、データベース内に記憶する。

【００８９】

図２１に示すように、次のステップ１２３では、２つの時点における選択された列内のエンティティがキャプチャされる。このステップは、異なる時点における各キーの位置に関する情報を提供する。

【００９０】

２つの時点間のスプレッドシートの変更を追跡するために、本システムはトラッカを使用して、この時点間の各キーの移動を判断するように構成される。次のステップ１２５では、本システムは、どのキーがこの時点間にシフトしたかを判断する。これを達成するために、第１の時点でキャプチャされたデータは各キーを検索され、第１の時点の各キーの行位置が判断される。第２の時点でキャプチャされたデータは各キーを検索され、第２の時点での各キーの行位置が判断される。第１の時点での各キーの行位置は、オフセット判断器により第２の時点での対応キーの行位置から減じて、２つの時点間に各キーがシフトした行の数を示す値またはオフセットを定義する。キーがシフトしない場合、これらのキーの移動は「固定」としてカテゴリ化し、キーがシフトした場合、「非固定」としてカテゴリ化することができる。

【００９１】

２つの特定の時点間のキーの移動を表す情報は、データベース内に記憶してもよい。各キーのデータベースエントリが作成され、数個のバケットのうちの１つに配置される。各バケットはキーの移動の可能なタイプの１つに対応する。例えば、各固定キーのエントリは、「固定」バケット内に配置されるが、非固定キーのエントリは「非固定」バケット内に配置される。これらのエンティティの各々は、キーがある今の行と、キーが存在した以前の行とを含む場合がある。

【００９２】

２つの時点間で行が挿入または削除された時、挿入された行または削除された行のエンティティは１つの時点では存在するが、他の時点では存在しない。例えば、行が挿入された時、後の時点には存在するが先の時点では存在しない挿入された行に新規エンティティ

が作成される。同様に、行が削除された時、削除された行のエンティティは先の時点で存在するが、後の時点では存在しない。これらのプロセスが、選択された列の既存キーの削除や新規キーの作成をもたらす。

【００９３】

次のステップ１２７では、システムのエンティティ作成判断器は、どのキーが後の時点の新規キーであるかを判断する。さらなるステップ１２９では、システムのエンティティ削除判断器は、先の時点のどのキーが削除されたかを判断する。本システムは、データベース内に記憶された情報を分析して、先の時点のいずれのエンティティも後の時点で存在しないか否かを判断するように構成される。このようなエンティティは、「削除済」としてカテゴリ化されてもよい。本システムはさらに、先の時点で存在しない、後の時点の新規エンティティがあるか否かを判断するように構成される。このようなエンティティは、「新規」としてカテゴリ化されてもよい。各々の削除されたエンティティ用のデータベースエントリが作成されて、「削除済」バケット内に配置され、各々の新規エンティティ用のデータベースエントリが作成されて、「新規」バケット内に配置されてもよい。各エントリは、関連時点においてエントリが存在する行またはエントリが存在した行を含んでもよい。

【００９４】

本システムはさらに、データベース内に記憶された情報を分析して、２つの時点の任意のキーが同一であるか否かを判断するように構成される。このことは、例えば、選択された元のキーが全て固有でなかった場合、選択された列のエンティティが修正される場合、または新規の非固有のキーが作成される場合に生じる。キーが固有でない場合、そのキーは「複製」としてカテゴリ化される。次のステップ１３１では、本システムのエンティティ複製判断器が、複製が存在するか否かを判断する。各々の複製キーのデータベースエントリが作成されて、「複製」バケット内に配置されてもよい。

【００９５】

時点間の各キーの移動が関連バケット内に配置されると、２つの時点のスプレッドシートのセル単位の比較は上述したように行うことができるが、エンティティのシフトを考慮に入れるように修正できる。このことは、行の挿入や削除などの動作やソート動作により生じるノイズの影響を取り除く。

【００９６】

行挿入、行削除、行の垂直方向のソートなどのプロセスでは、各列内のエンティティは同じ量だけシフトする。例えば、行挿入動作では、エンティティがどの列にあるかに関係なく、挿入された行の上で生じるエンティティの全てはシフトせず、挿入された行の下で生じるエンティティの全ては１セルだけ下にシフトする。同様に、スプレッドシート内の値が第１の列の値に従ってソートされる時、第１の列の特定の行にある１つのエンティティが、例えば５セルだけ上にシフトする場合、他の列の同じ行にあるエンティティも同じ量だけ上にシフトする。したがって、時点間でキーが特定の量だけシフトした場合、他の列のそのキーと同じ行にあるエンティティがそのキーと同じようにシフトしたと推測することができる。

【００９７】

図２１に示すように、次のステップ１３３では、ステップ１３５でファイルが異なる時点のファイルのバージョンと比較される前に、オフセットがオフセットアプリケータによりスプレッドシートファイルに適用される。上述したようなセル単位の比較を使用する時、２つの時点間の２つＸＭＬファイルを比較する時、非固定キーに対応する各々の行に対して、適切なオフセットがＸＭＬで表示される行数に追加される。この結果、２つの時点でたまたま同じセルを占める２つの値ではなく、２つの同等なシフトをした値の比較をすることになる。前時点のファイルを無視して、全ての新規キーを現時点のファイルから追加することができる。現時点のファイルを無視して、全ての削除キーを前時点のファイルから追加することができる。２つの時点間の正確なキー識別を確実にできないため、複写キーとして識別されるどのキーも処理時には無視される。

【００９８】
上述した例では、キーを定義するために１つまたは複数の列を選択して、行挿入や行削除の動作および行の垂直方向のソート動作を考慮に入れる。しかし、キーを定義するために１つまたは複数の行を選択して、列挿入や列削除の動作および列の水平方向のソート動作を考慮に入れる同様の技術を使用してもよいことがわかる。行と列の両方を組み合わせて選択することも、当業者には理解できるだろう。
【００９９】
圧縮数式ストレージ
上述および図示した本システムはさらに、スプレッドシートを処理して圧縮数式ストレージを提供するように構成される。数式ストレージ圧縮プロセスは、本システムにより自動的に行われるのが好ましい。例えば、本システム内に記憶された各スプレッドシートは、監査段階で本発明に従って自動的に検索され、処理されてもよい。しかし、いくつかの実施形態では、比較プロセスは、具体的には、ユーザコマンドに応答して、場合によっては選択されたスプレッドシート上で行ってもよい。
【０１００】
スプレッドシートの圧縮数式ストレージ処理は、１つまたは複数のサーバ４４に記憶され、実行される適切なソフトウェアにより実行してもよい。
【０１０１】
スプレッドシートが処理すべきであると判断された時、関連スプレッドシートがサーバ４４の制御の下、ネットワーク１２内から検索される。上述したように、システム内のファイルストレージは、各クライアントコンピュータ１０と、ネットワーク１２上のファイルサーバ１４または各ファイルサーバ１４とにより提供されるローカルストレージを備える。スプレッドシートの処理時、判断ユニットが、スプレッドシート内で生じる各固有の数式を判断し、数式テーブルが構築される。数式テーブルは、第１の記憶領域にスプレッドシート内で生じる各固有の数式を記憶し、第２の記憶領域に各数式の固有数式ＩＤを一緒に記憶する。以下でより詳細に述べるように、このようにして、スプレッドシート内で生じる数式は、置換ユニットにより、数式テーブルで定義された時にそれらの対応するＩＤ値と置換してもよい。数式テーブル内のエントリは、スプレッドシート内で生じる数式の各インスタンスではなく、各固有の数式に対してのみ必要であるため、また数式ＩＤが数式そのものよりも明らかに小さいスペースを占めるため、ストレージのかなりの縮小になる。いくつかのスプレッドシートでは、スプレッドシート内の数式の最大で９９．５％までが複製される。スプレッドシート内で生じる数式の各インスタンスではなく、固有の数式のみを記憶することにより、ほとんど９９．５％のスペースの節約になる。
【０１０２】
好適な実施形態では、数式テーブルは１つまたは複数のエントリを備える。各エントリはスプレッドシート内の固有の数式に対応する。数式テーブル内の各エントリは、３つのフィールドを備える。３つのフィールドは、以下の形式を有する。
［ＣｕｒｒｅｎｔＦｏｒｍｕｌａ］［ｎｖａｒｃｈａｒ］（２０４８）
［ＦｏｒｍｕｌａＩＤ］［ｉｎｔ］ＮＯＴ　ＮＵＬＬ
［ＦｏｒｍｕｌａＣｈｅｃｋＳｕｍ］ＡＳ（ｂｉｎａｒｙ＿ｃｈｅｃｋｓｕｍ（［ＣｕｒｒｅｎｔＦｏｒｍｕｌａ］））
【０１０３】
ＣｕｒｒｅｎｔＦｏｒｍｕｌａとラベル付けされた第１のフィールドは、テーブルエントリに対応する数式を表す文字列を記憶する。この例では、ＣｕｒｒｅｎｔＦｏｒｍｕｌａは、最大で２０４８までの文字を有するｎｖａｒｃｈａｒ型（可変長の文字列を記憶するデータ型）の変数である。例えば、あるＣｕｒｒｅｎｔＦｏｒｍｕｌａは、文字列“＝Ａ２＋Ｂ３＋ＳＵＭ（Ｄ１：Ｄ３）”としてもよい。
【０１０４】
ＦｏｒｍｕｌａＩＤとラベル付けされた第２のフィールドは、ＣｕｒｒｅｎｔＦｏｒｍｕｌａフィールド内の数式を一意に識別する識別値を記憶する。したがって、スプレッド

シート内の数式ＩＤと固有の数式とは１対１に対応し、このためスプレッドシート内のどの数式も対応する数式ＩＤを使用して識別することができる。この例では、ＦｏｒｍｕｌａＩＤは、非ゼロ値を有するｉｎｔ型（可変の整数値を記憶するデータ型）の変数である。例えば、あるＦｏｒｍｕｌａＩＤは、値「３１」をとってもよい。

【０１０５】

ＦｏｒｍｕｌａＣｈｅｃｋＳｕｍとラベル付けされた第３のフィールドは、ＣｕｒｒｅｎｔＦｏｒｍｕｌａフィールドから導出された値を記憶する。この値は、以下でより詳細に述べるように、数式間の比較を容易にするのに使用される。

【０１０６】

どのスプレッドシートの処理の前も、数式テーブルは空である。次に、テーブルは、１つまたは複数のスプレッドシートの処理時にポピュレートされる。１つの例示的な方法では、図２０のフローチャートを参照して説明すると、スプレッドシートの処理は以下のステップを含む。スプレッドシート内のセルごとに検索ユニットにより検索が繰り返される。第１のステップ１０１では、数式判断ユニットにより第１のセルがチェックされ、セルが数式または別の形式のデータを含むか否かを判断する。一実施形態では、各セルは、各セル内に記憶されたデータの型を示す関連データ型情報を備える。この場合、セルのデータ型情報を分析して、セルが数式を含むか否かを判断する。

【０１０７】

セルが数式を含まないと判断される場合、次のセルが検索される（１０３）。セルが数式を含むと判断される場合、次のステップ１０５で、テーブル比較ユニットにより、その数式が数式テーブル内に記憶されている数式のセットと比較される。次のステップ１０７では、数式複製判断ユニットが、その数式のエントリがすでに数式テーブル内に存在するか否かを判断する。

【０１０８】

スプレッドシート内の数式と数式テーブル内の数式との比較は、任意の適切な比較アルゴリズムを使用して数式の文字列を比較することで、比較器により行ってもよい。好適な実施形態では、数式の比較を容易にするために、数式テーブル内の各エントリが、第１のフィールドに記憶されている数式文字列から導出された数値を記憶するように構成された第３のフィールドを備える。この値は、各固有の数式文字列が固有値を生成するように導出されるのが好ましい。一実施形態では、導出された値は、任意の適切な既知のチェックサム関数により数式の文字列から導出された数値を含むバイナリチェックサム値である。新規エントリが数式テーブルに追加されると、記憶されている数式のバイナリチェックサム値が計算され、数式文字列と一緒に記憶される。この例では、バイナリチェックサム値は、変数ＣｕｒｒｅｎｔＦｏｒｍｕｌａを引数とするシフトウェアのＢＩＮＡＲＹ＿ＣＨＥＣＫＳＵＭ関数を使用して計算される。異なる数式は、異なるチェックサム値を生成する。

【０１０９】

スプレッドシート内の数式が数式テーブル内の数式を含むか否かを判断するために、数式のバイナリチェックサム値がプロセッサにより計算され、数式テーブル内に記憶されているバイナリチェックサム値と比較される。２つの文字列を比較することより、２つの数値を比較することの方が簡単であるので、バイナリチェックサム値の使用は、計算効率をかなり向上させる。

【０１１０】

現セル内に含まれる数式のエントリがすでに数式テーブル内に存在する場合、スプレッドシート内の次のセルが検索される（１０３）。その数式のエントリが数式テーブル内に存在しない場合、次のステップ１０９で、エントリ作成ユニットにより新規エントリが作成される。このステップでは、数式を表す文字列が新規エントリの第１のフィールド内に記憶され、新規固有の数式ＩＤがその数式に割り当てられて、新規エントリの第２のフィールド内に記憶され、数式文字列のバイナリチェックサム値が計算されて、新規エントリの第３のフィールドに記憶される。

【０１１１】
次に、次のセルが検索され（１０３）、スプレッドシート内の各セルが処理されてしまうまで上述のプロセスが繰り返される。
【０１１２】
数式テーブルが構築されると、スプレッドシート内に含まれる各数式は、置換ユニットにより、数式テーブル内で定義された時に対応する数式ＩＤと置換される。あるいは、スプレッドシート内に含まれる数式は、数式テーブルが構築される時に対応する数式ＩＤと置換されてもよい。
【０１１３】
一実施形態では、数式テーブルを構築するためのスプレッドシートファイルの処理に伴う動作は、数式ＩＤヘルパークラスにより実行される。本明細書で説明したシステムにおいて、このクラスは、親クラスから派生したものであり、その構造は図３に示す。
【０１１４】
スプレッドシートが、圧縮数式ストレージを達成するために処理されると、元のスプレッドシートを再構築するために、スプレッドシート内の各セルが順に繰り返され、セルが数式ＩＤを含む場合、その数式ＩＤは、数式テーブルにより定義された時に対応する数式文字列と置換される。
【０１１５】
多くの場合、スプレッドシート内に現れる２つ以上の数式は、同じ形式を持つ（すなわち、同じ数学関数を示す）が、特定のセルに関して、数式内で生じる引数は異なる。しかし、場合によっては、各数式内では、引数における参照されたセル間の位置関係や数式を含むセルは同じである場合がある。これは、例えば、数式がある列内のセルに対して作成され、その数式が下へドラッグされ、すぐ下のセル内にコピーされる時のＭｉｃｒｏｓｏｆｔ　Ｅｘｃｅｌの場合である。下の表は、このプロセスが発生するスプレッドシートの一部を示す。
【０１１６】
【表１】

	...	D	E	
	⋮	⋮	⋮	
5	...	12	=2*(E4+D5)	...
6	...	15	=2*(E5+D6)	...
7	...	7	=2*(E6+D7)	...
8	...	64	=2*(E7+D8)	...
	⋮	⋮	⋮	

【０１１７】
列Ｅ内の各数式は、異なる引数が使用されているという点で異なるが、各数式は、$2^{*}(a_1+a_2)$の形式である。ここで、a_1とa_2とは引数である。さらに、各数式において、第１の引数は常に数式が位置するセルの真上のセルにあり、第２の引数は常に数式が位置するセルの真左のセルにある。好適な実施形態では、各数式において、引数における参照されるセル間の位置関係や数式を含むセルが同じである場合、２つ以上の数式が同じであると見なされる。これは、数式を表す文字列が同一である場合に２つ以上の数式が同じであると特定するよりも条件は厳しくない。しかし、より厳しい条件が使用された場合には、典型的なスプレッドシート内で生じるほとんどの数式は固有なものとなり、スペース

のほんのわずかな節約にしかならない。

【０１１８】

圧縮数式ストレージの効率を向上させるために、２つの数式の比較が行われる前に、これらの数式はまず、相対表記スタイル（Ｒ１Ｃ１または任意の他の適切なスタイル）で表示すべきである。このことにより、数式文字列が、スプレッドシート内の数式の大部分を固有なものにする特定のセルアドレス情報を含まなくなる。したがって、２つの数式間の比較が行われる際には、比較の前にまず数式の文字列が相対表記スタイルに変換される。バイナリチェックサム値が計算される際には、バイナリチェックサム値の計算の前にまず値が導出される数式文字列が相対表記スタイルに変換される。

【０１１９】

上述した実施形態では、本発明に従って既存のスプレッドシートが処理されて、圧縮数式ストレージを有するスプレッドシートが得られる。あるいは、本発明による圧縮数式ストレージ技術を、スプレッドシートが作成され、編集される時に同時に適用することもできる。例えば、ユーザがスプレッドシートを編集して、数式をセルに挿入する場合、本システムは、その時点で数式テーブル内のエントリがその数式用に存在するか否かを判断するように構成される。もし存在する場合、数式をスプレッドシートに挿入する代わりに、対応する数式ＩＤが挿入される。もし存在しない場合、その時点で数式テーブル内に新規エントリが作成される。

【０１２０】

数式テーブルは、ファイルサーバにより記憶され、管理されてもよい。好適な実施形態では、数式テーブルはスプレッドシートファイルとは別のデータファイル内に記憶される。しかし、代替の実施形態では、スプレッドシート用の数式テーブルは、スプレッドシートファイルのヘッダ内のようにスプレッドシートファイル自身の中に記憶してもよい。この構成により、スプレッドシートファイルが転送される時、必要な数式テーブル情報がすでにファイルの一部を形成するため、ファイルのエクスポートがより簡単になる。別個の数式テーブルファイルが提供される場合には、ファイルが転送される時、必要な数式テーブルファイルもスプレッドシートファイルと一緒に転送されて、元のスプレッドシートが再構築される。一実施形態では、別個の数式テーブルがそれぞれ個々のスプレッドシートに対して作成される。しかし、複数の異なるスプレッドシートで同じ数式を共有する場合もあるので、圧縮を最小にするために、１つの数式テーブルが複数または全てのスプレッドシートに使用される。

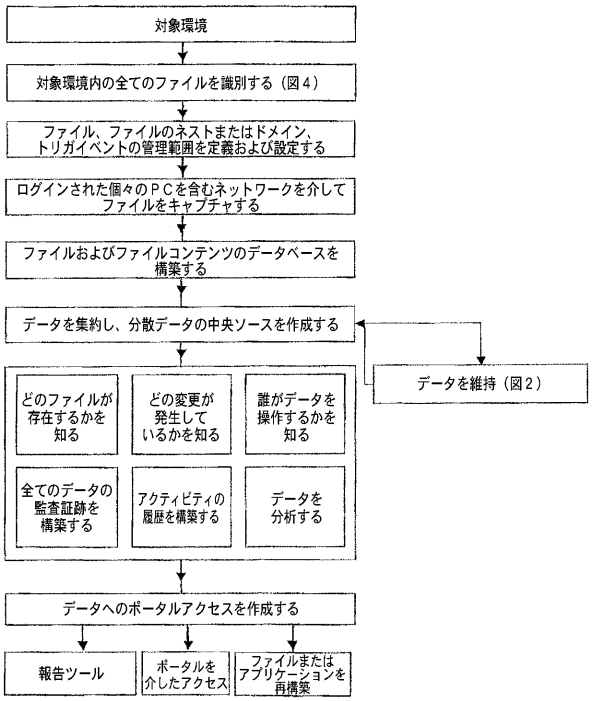
【０１２１】

上述した方法は、メモリ内に記憶された適切なソフトウェアにより実装してもよい。該ソフトウェアは、適切なプロセッサにより実行するためのコンピュータ実行可能コードを備える。

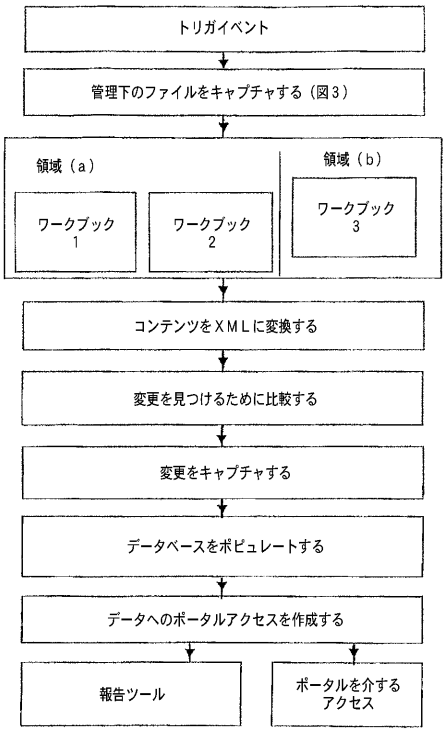
【０１２２】

ＭｉｃｒｏｓｏｆｔおよびＥｘｃｅｌは、Ｍｉｃｒｏｓｏｆｔ社の登録商標である。

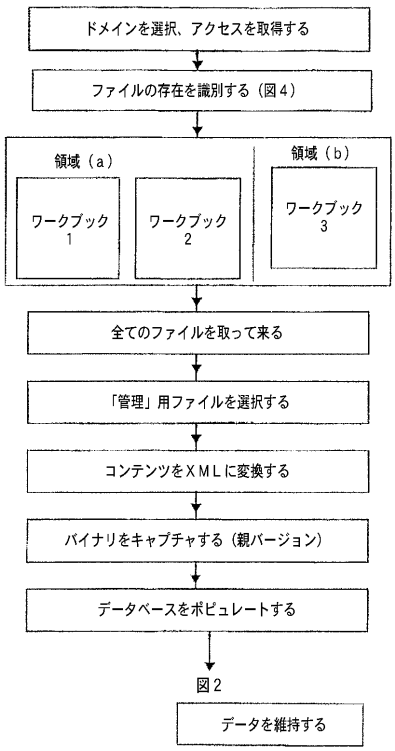
【図１】


対象環境
対象環境内の全てのファイルを識別する（図４）
ファイル、ファイルのネストまたはドメイン、トリガイベントの管理範囲を定義および設定する
ログインされた個々のＰＣを含むネットワークを介してファイルをキャプチャする
ファイルおよびファイルコンテンツのデータベースを構築する
データを集約し、分散データの中央ソースを作成する
データを維持（図２）
どのファイルが存在するかを知る
どの変更が発生しているかを知る
誰がデータを操作するかを知る
全てのデータの監査証跡を構築する
アクティビティの履歴を構築する
データを分析する
データへのポータルアクセスを作成する
報告ツール
ポータルを介したアクセス
ファイルまたはアプリケーションを再構築

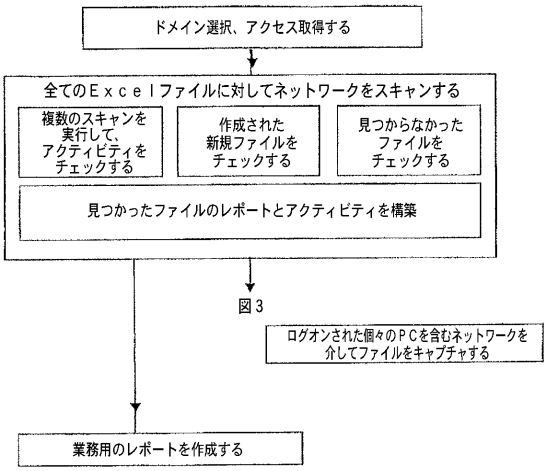
【図２】


トリガイベント
管理下のファイルをキャプチャする（図３）
領域（ａ）
領域（ｂ）
ワークブック 1
ワークブック 2
ワークブック 3
コンテンツをＸＭＬに変換する
変更を見つけるために比較する
変更をキャプチャする
データベースをポピュレートする
データへのポータルアクセスを作成する
報告ツール
ポータルを介するアクセス

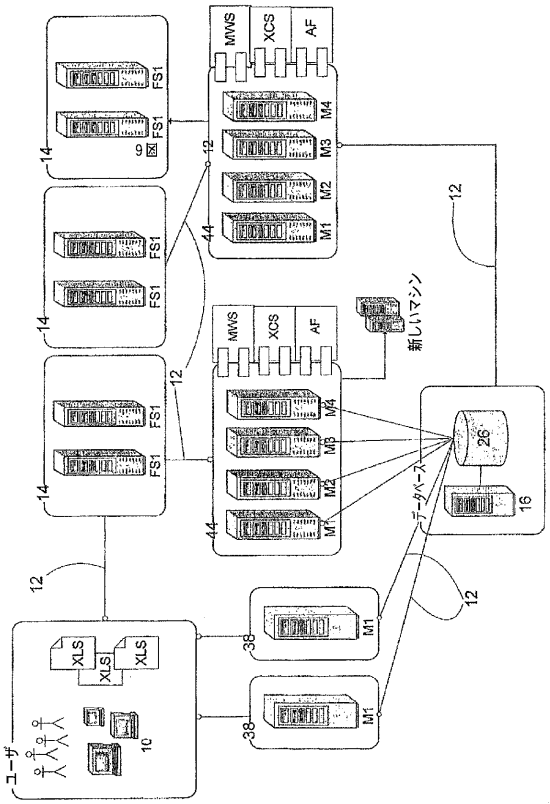
【図３】


ドメインを選択、アクセスを取得する
ファイルの存在を識別する（図４）
領域（ａ）
領域（ｂ）
ワークブック 1
ワークブック 2
ワークブック 3
全てのファイルを取って来る
「管理」用ファイルを選択する
コンテンツをＸＭＬに変換する
バイナリをキャプチャする（親バージョン）
データベースをポピュレートする
図２
データを維持する

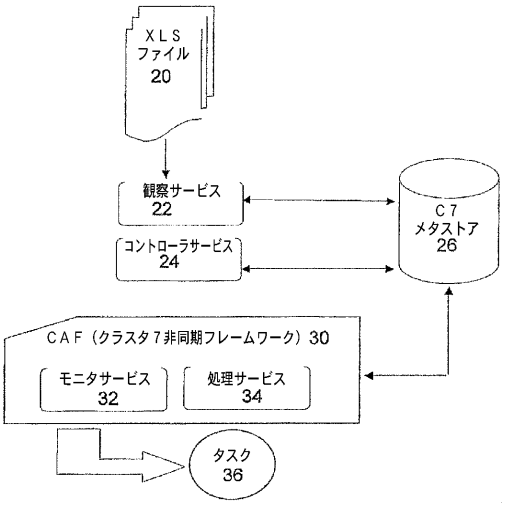
【図４】


ドメイン選択、アクセス取得する
全てのＥｘｃｅｌファイルに対してネットワークをスキャンする
複数のスキャンを実行して、アクティビティをチェックする
作成された新規ファイルをチェックする
見つからなかったファイルをチェックする
見つかったファイルのレポートとアクティビティを構築
図３
ログオンされた個々のＰＣを含むネットワークを介してファイルをキャプチャする
業務用のレポートを作成する

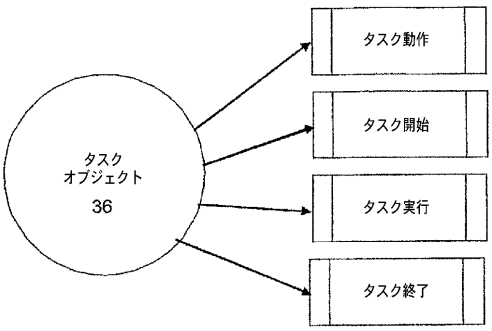
【図５】


MWS
XCS
AF
M4
M3
M2
M1
FS1
新しいマシン
データベース
ユーザ
XLS
12
14
16
26
38
44

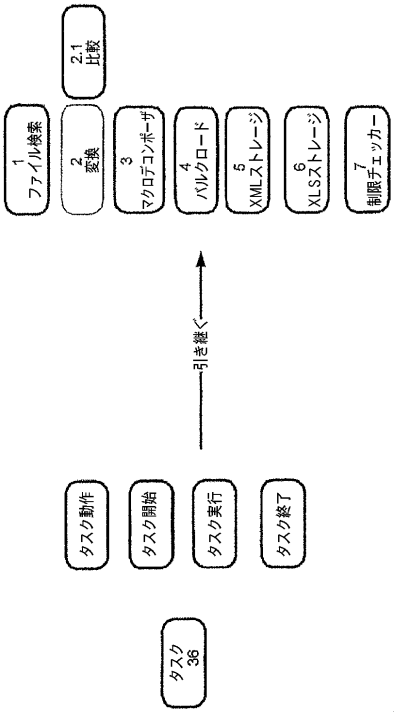
【図６】


XLS
ファイル
20
観察サービス
22
コントローラサービス
24
C7
メタストア
26
CAF（クラスタ7非同期フレームワーク）30
モニタサービス
32
処理サービス
34
タスク
36

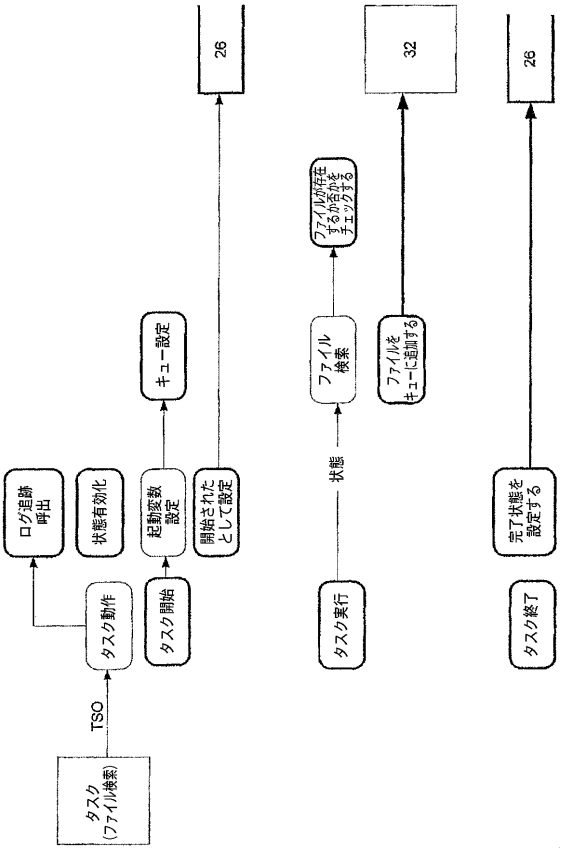
【図７】


タスク
オブジェクト
36
タスク動作
タスク開始
タスク実行
タスク終了

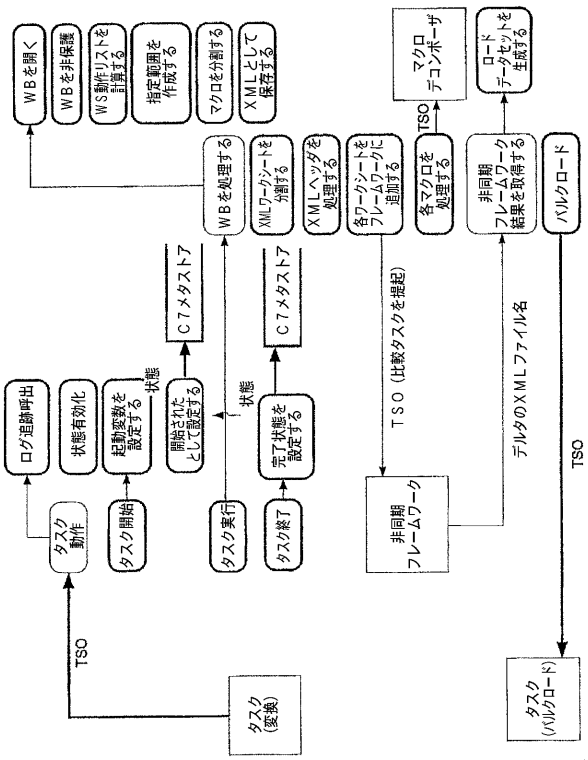
【図８】


1
ファイル検索
2
変換
2.1
比較
3
マクロデコンポーザ
4
バルクロード
5
XMLストレージ
6
XLSストレージ
7
制限チェッカー
引き継ぐ
タスク動作
タスク開始
タスク実行
タスク終了
タスク
36

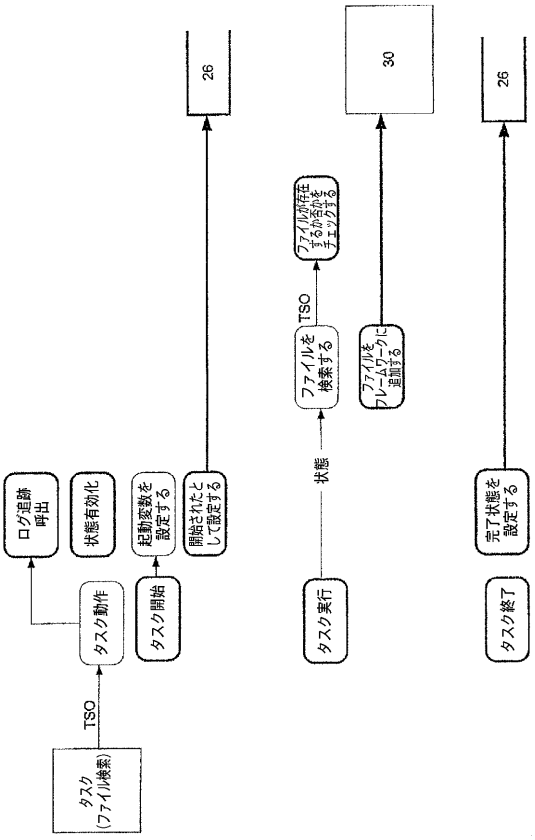
【図９】


26
32
26
ログ追跡
呼出
状態有効化
起動変数
設定
開始された
として設定
キュー設定
ファイル
検索
ファイルが存在
するか否かを
チェックする
ファイルを
キューに追加する
状態
完了状態を
設定する
タスク
(ファイル検索)
TSO
タスク動作
タスク開始
タスク実行
タスク終了

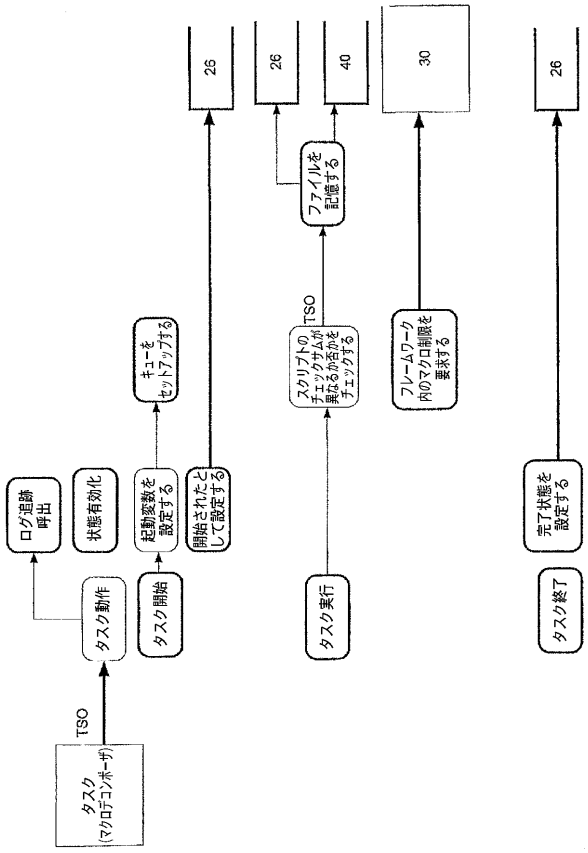
【図１０】



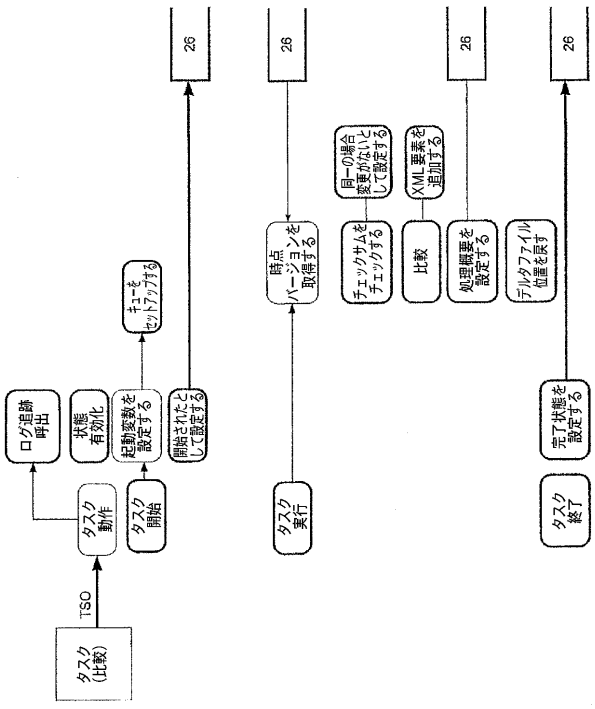
【図１１】



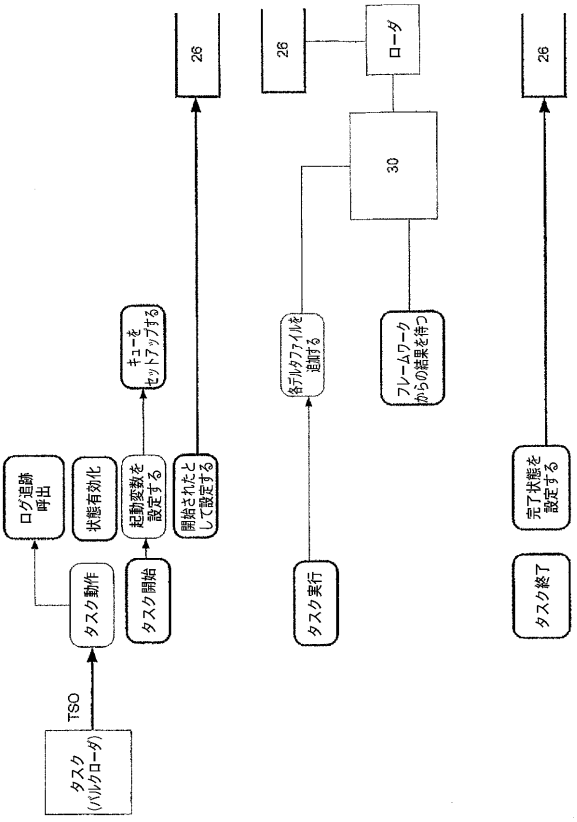
【図１２】



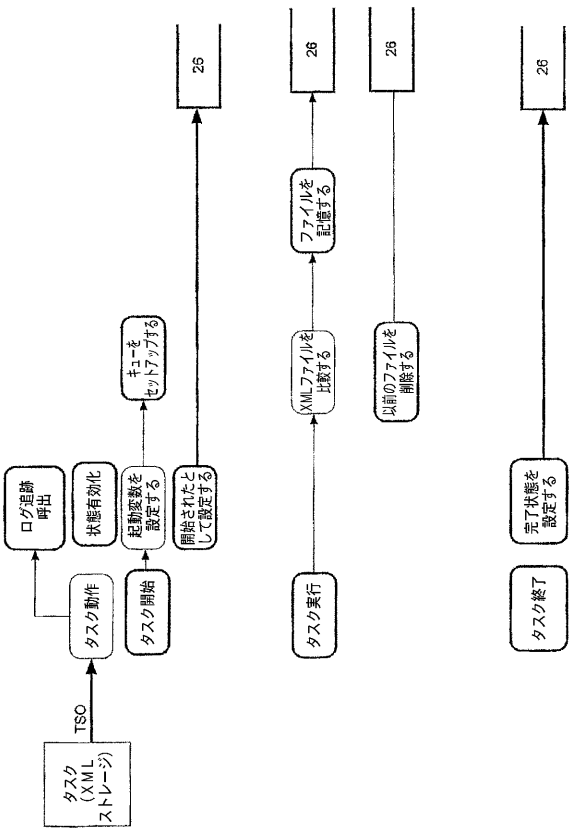
【図１３】



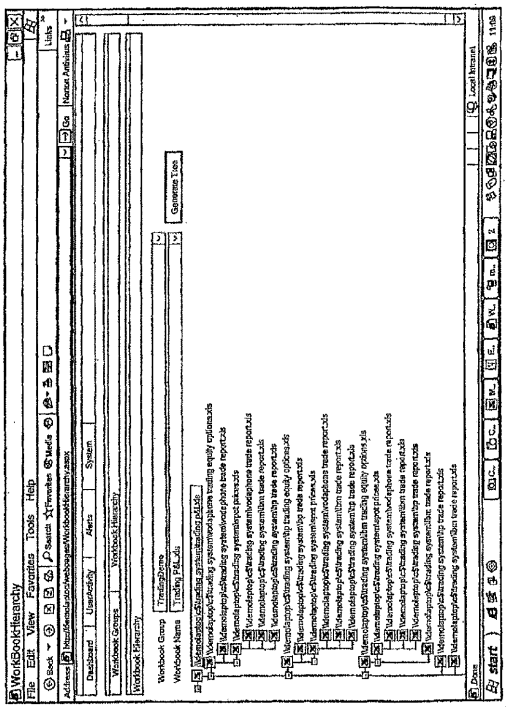
【図１４】



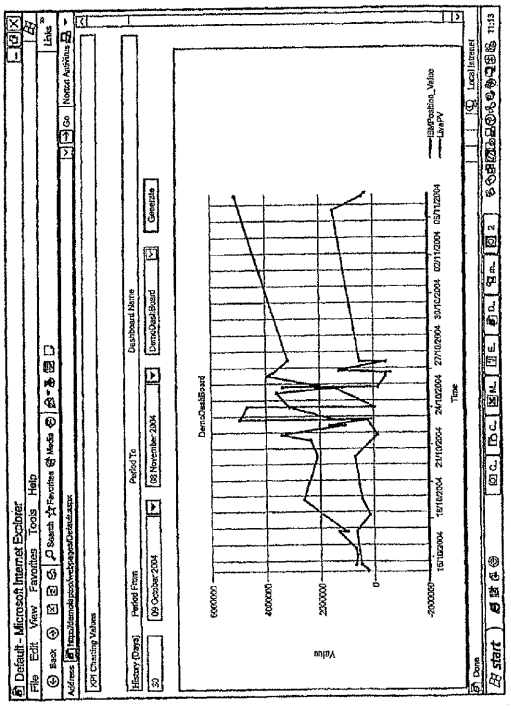
【図１５】



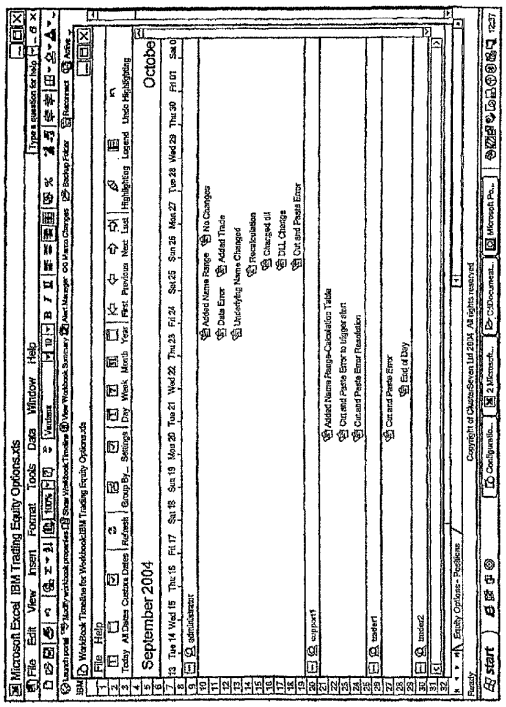
【図１６】



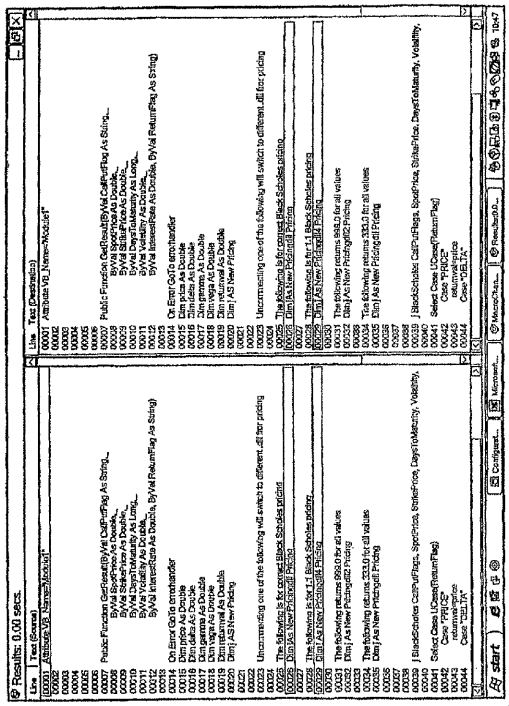
【図１７】



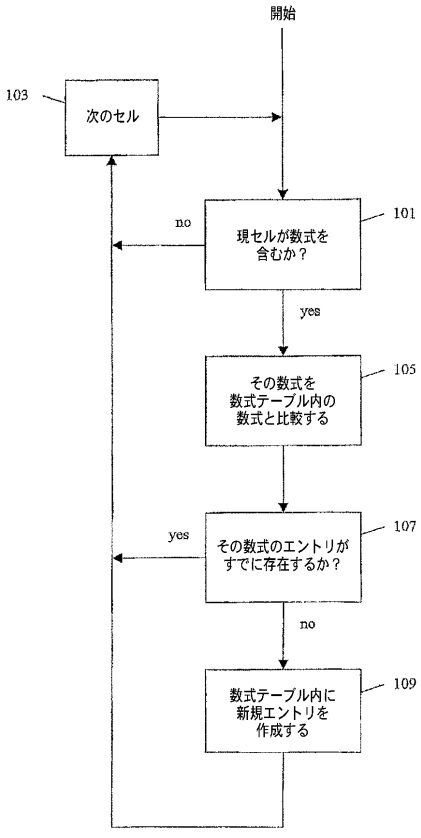
【図１８】



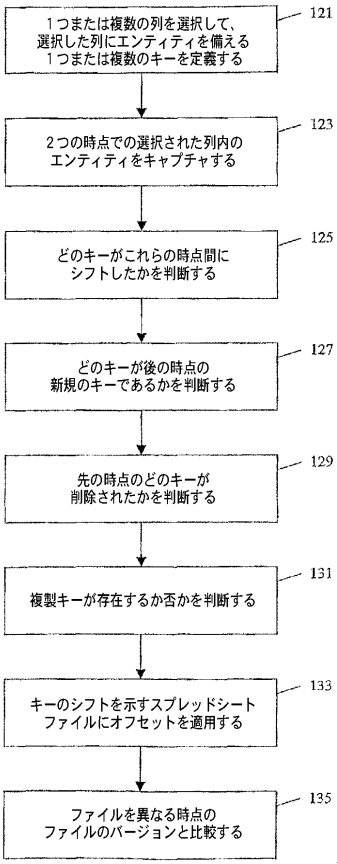
【図１９】



【図２０】



【図２１】



フロントページの続き

(72)発明者　ジェームズ　カルバーウェル
イギリス　ビーエイチ９　１エヌジー　ドーセット　ボーンマス　チャミンスター　ポートランド　ロード　１７０
(72)発明者　アーロン　ウィットマン
イギリス　ダブリュ２　６ディーエイチ　ロンドン　クリーブランド　スクエア　１０－１１　２２

審査官　久々宇　篤志

(56)参考文献　国際公開第２００５／０８１１２６（WO，Ａ１）
米国特許第０５３０３１４６（ＵＳ，Ａ）
特開２００１－１６２２６１（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F　17/21 - 17/28
G06F　12/00
G06Q　10/10
G06F　17/30