(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5021677号
(P5021677)
(45)発行日　平成24年9月12日(2012.9.12)　(24)登録日　平成24年6月22日(2012.6.22)

(51)Int.Cl.　F I
G06F 12/00 (2006.01)　G06F 12/00 535P

請求項の数 9　(全 41 頁)

(21)出願番号	特願2008-548512 (P2008-548512)
(86)(22)出願日	平成18年11月17日 (2006.11.17)
(65)公表番号	特表2009-522638 (P2009-522638A)
(43)公表日	平成21年6月11日 (2009.6.11)
(86)国際出願番号	PCT/US2006/044735
(87)国際公開番号	WO2007/078444
(87)国際公開日	平成19年7月12日 (2007.7.12)
審査請求日	平成21年11月10日 (2009.11.10)
(31)優先権主張番号	11/275,434
(32)優先日	平成17年12月30日 (2005.12.30)
(33)優先権主張国	米国(US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　ジョナサン　アール．ハウエル
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　インターナショナル　パテンツ内

最終頁に続く

(54)【発明の名称】デルタページャを使用した状態の管理

(57)【特許請求の範囲】
【請求項１】
複数のレコードを有するデータベースであって、各レコードはある値へのアドレスの割り当てを含み、前記データベースは前記データベースの現行状態を追跡するための現行状態ポインタを有する、データベースを格納する記憶装置を有するコンピュータが、
前記データベースに対する複数の命令を含むトランザクションを受け取るステップと、
第１の書き込みバッファを作成し、当該第１の書き込みバッファから、前記現行状態ポインタによって指示されるオブジェクトへのポインタを確立するステップと、
前記複数の命令の各命令について、当該命令が前記データベースに変更を適用する場合、追加の書き込みバッファを作成し、当該追加の書き込みバッファから、前記トランザクションの実行により作成された書き込みバッファのうち、直前に作成された書き込みバッファへのポインタを確立し、前記命令が前記データベースに適用しようとする変更を前記追加の書き込みバッファに格納するステップと、
前記現行状態ポインタに、前記トランザクションの実行により作成された書き込みバッファのうち、最後に作成された書き込みバッファを指示させることにより、前記トランザクションをコミットするステップと、
合体開始ポイントを識別するステップと、
前記合体開始ポイントと次の合体されていないオブジェクトとの間に合体済みオブジェクトを作成するステップと、
合体する可能性があるオブジェクトが存在しなくなるまで、前記合体済みオブジェクト

の次のオブジェクトに進み、当該次のオブジェクトに格納された変更を、前記合体済みオブジェクトに格納し、当該次のオブジェクトを解放するステップと、
前記現行状態ポインタに、前記合体済みオブジェクトを指示させるステップと
を含むことを特徴とする方法。
【請求項２】
前記データベースにアクセスし、特定のアドレスの値を求めるステップであって、
前記特定のアドレスの割り当てが見つかるまで、または前記データベースに到達するまで、前記現行状態ポインタによって指示されるオブジェクトから開始し、当該オブジェクトからのポインタによって指示される次のオブジェクトに進む、ステップをさらに含むことを特徴とする請求項１に記載の方法。
【請求項３】
前記コミットするステップは、
前記第１の書き込みバッファからのポインタ、および前記現行状態ポインタが同じオブジェクトを指示する場合、または
介入トランザクションが、前記トランザクションがアクセスしたアドレスの値を変更していない場合に、実行されることを特徴とする請求項１に記載の方法。
【請求項４】
前記トランザクションがコミットされない場合、前記トランザクションの実行により作成された書き込みバッファを破棄することによって、前記トランザクションを打ち切るステップをさらに含むことを特徴とする請求項３に記載の方法。
【請求項５】
現行状態を保持する命令に応答して、当該保持する命令を受けた時点で前記現行常状態ポインタによって指示されるオブジェクトを指示するスナップショットポインタを作成するステップを含み、
前記次のオブジェクトがスナップショットポインタによって指示される場合、前記スナップショットに前記合体済みオブジェクトを指示させ、前記合体済みオブジェクトと次の合体されていないオブジェクトとの間に追加の合体済みオブジェクトを作成することを特徴とする請求項１に記載の方法。
【請求項６】
前記記憶装置が低速ストアであり、前記トランザクションの実行により作成されたバッファおよび合体済みオブジェクトが高速ストアに格納されている場合に、
前記低速ストアに格納されたデータベースのキャッシュオブジェクトを前記高速ストアに作成するステップと、
前記キャッシュオブジェクトから前記データベースへのポインタを確立するステップと、
前記データベースを指示する、前記高速ストア内のオブジェクトからのポインタに、前記キャッシュオブジェクトを指示させるステップと
をさらに含むことを特徴とする請求項１に記載の方法。
【請求項７】
前記高速ストアはメモリを含み、前記低速ストアはディスクストレージを含むことを特徴とする請求項６に記載の方法。
【請求項８】
前記データベース内の少なくとも１つのアドレスの割り当てを前記キャッシュオブジェクトにコピーするステップと、
前記キャッシュオブジェクト内の少なくとも１つのアドレスの割り当てを前記データベースにコピーするステップと、
のうちの少なくとも１つをさらに含むことを特徴とする請求項６に記載の方法。
【請求項９】
新しいアドレスの割り当てを前記キャッシュオブジェクトに書き込むと同時に、前記新しい割り当てを前記データベースに書き込むステップをさらに含むことを特徴とする請求

項６に記載の方法。
【発明の詳細な説明】
【背景技術】
【０００１】
データベーストランザクションを実行する際の重要な問題は、データベースの整合性である。データベースの整合性を保持するためには、トランザクションが原子性（ａｔｏｍｉｃｉｔｙ）および分離性などの諸要件を確実に監視することが重要である。原子性要件は、トランザクションのタスクのすべてが実行されるか、またはそれらがまったく実行されないかの、いずれかであることを要求するため、不完全なトランザクションはデータベースに適用されない。分離性要件は、トランザクションが他のトランザクションとは別々に処理されることを要求するため、トランザクションが、他のトランザクションの中間状態にアクセスすること、またはこれに干渉することはない。原子性要件および分離性要件を監視することによって、トランザクションが互いに破損し合うこと、およびデータベース全体を破損することを防ぐ。
【０００２】
ネットワークシステムでは、多くのユーザおよび多くのアプリケーションが同時に同じデータ上でトランザクションを実行しようとする場合がある。たとえスタンドアロン型システムであっても、複数のアプリケーションが同時に同じデータ上でトランザクションを実行しようとする場合があり、各アプリケーションの複数の処理スレッドが競合し合う可能性がある。トランザクションが競合または矛盾する変更をデータベースに適用する可能性があるすべての状況を防止することはもちろん、予測することも困難であるため、データベースの整合性を維持することは、重要な課題である。
【０００３】
データベースの整合性を保持する方法の１つが、現行トランザクションによって使用されているデータベースの一部分をロックすることである。データベースの一部分をロックすることにより、現行トランザクションが完了するまで、他のトランザクションによるデータの読み取りまたは上書きを防止する。他のトランザクションによるデータベースの一部分の読み取りを防止することで、他のトランザクションが、現行トランザクションが変更する可能性のある値に基づく結果に達しないことが保証される。同様に、他のトランザクションがデータベースのその部分を上書きするのを防止することで、他のトランザクションが現行トランザクションに影響を与えないことが保証される。
【０００４】
しかしながら、データベースの一部分をロックすることは、データの整合性を保持するのを助ける可能性はあるが、それに対抗するコストが生じる可能性もある。いくつかのトランザクションがアクセスしようとしているデータベースの一部分をロックすることで、トランザクションのバックログが大量に生じる可能性がある。待機中のトランザクションの一部またはすべてが、データベースの現行状態を変更しない可能性があるか、または、現行トランザクションが使用している特定の値からの読み取り、この値への書き込み、またはこの値の変更を行わない可能性がある。それでもやはり、データベースの一部分をロックすることは、たとえデータベースへのアクセスが遅くなっても、データベースの整合性を保持することになる。
【０００５】
ロックの１つの形がリースである。リースとは、アパート、ビル、車、および設備のリースの場合とまったく同様の、限定期間にわたる独占許可である。したがって、あるトランザクションが選択されたデータに関するリースを認められた場合、そのトランザクションには、ある限定期間にわたる選択されたデータへの独占的なアクセス権が与えられる。データへの独占的アクセス権をある期間に限定することによって、トランザクションが完了した際にトランザクションがデータの解放に失敗した場合、またはトランザクションが実行されているシステムがクラッシュした場合、データはロック解除され、他のトランザクションはデータへのアクセスを無期限に待つ必要がなくなる。通常、選択されたデータ

を待っている各トランザクションは、事前にデータに関するキューに入れられたすべてのトランザクションがそのデータの使用を完了するまで、または各トランザクションに割り振られたリースが満了するまで、待つ必要がなくなる。

【０００６】

リースされたデータへのアクセスをいくつかのトランザクションが待っている場合、潜在的にかなりの遅延が存在する可能性がある。加えて、データのリースを制御しているシステムは、データにアクセスしようとすると障害を引き起こす可能性がある。このリースを制御しているシステムは、データアクセスに関する単一の制御ポイントとして、リース要求に応答しようとするとトランザクションの遅延を発生させる可能性がある。さらにこのシステムは、選択された同じデータに関する複数の要求を受信している可能性があるため、これら多数の潜在的に繰り返される要求を処理することは、計算サイクルの無駄となり、結果として、同じデータまたは任意の他のデータを求めるトランザクションにさらなる遅延を発生させる。

【発明の開示】

【課題を解決するための手段】

【０００７】

デルタページャ（ｄｅｌｔａ　ｐａｇｅｒ）とは、原子的な分離されたトランザクションによってデータベースを維持するものである。トランザクションがデータベースを変更しようとする際、デルタページャは変更を書き込みバッファに格納し、介入するトランザクションが、事実上または実質上、トランザクションに依拠してデータベースの状態を変更しない場合、この変更を適用する。デルタページャは変更を適用して、データベースの状態を表す新しいデータ構造を形成するように、書き込みバッファとデータベースの現行状態とを結合させることによって、トランザクションをコミットする。デルタページャは、選択されたデータベースの状態を保持するために、デルタページャが遵守するスナップショットに従って、効率を維持するために書き込みバッファを合体（ｃｏａｌｅｓｃｅ）する。デルタページャは、選択されたデータを永続ストアへ移動することによって、データベースの選択されたセクションを永続的にする。デルタページャは、データへの効率的なアクセスを促進するために、永続ストアと現行トランザクションとの間にキャッシュオブジェクトも提供する。

【０００８】

この課題を解決するための手段は、以下の発明を実施するための最良の形態でより詳細に説明する概念の選択を簡略化された形で紹介するために提供される。この課題を解決するための手段は、主張された主題の主要な特徴または不可欠な特徴を識別することを意図するものでなく、また、主張された主題の範囲を決定する際の一助として使用されることを意図するものでもない。

【０００９】

添付の図面を参照しながら、詳細な記述について説明する。図面では、３桁の参照番号の一番左の数字または４桁の参照番号の左側２つの数字が、その参照番号が最初に示された図面を識別するものである。同じ参照番号が異なる図面で使用される場合、同様または同一のアイテムであることを示す。

【発明を実施するための最良の形態】

【００１０】

概要

「デルタページャ」という用語は、データベースの整合性を維持し、データベースの選択された状態を保持しながら、データベースへの変更を処理するための、方法およびシステムの諸実施形態を記述する。認識されたオブジェクト間でのポインタの不変性を遵守することによって、デルタページャは、効率性も維持しながらデータベースの状態の整合性を保持する。

【００１１】

１つのモードでは、デルタページャは、データベースに格納されたすべてのデータの完

全マッピングを含む、データベースの現行状態を追跡するために、現行状態ポインタを使用することによって、トランザクションを管理する。完全マッピングでは、データベースに関して定義された各アドレスまたはアドレスの表現について、データ値またはヌル（ｎｕｌｌ）値が存在することになる。部分マッピングは、トランザクションが開始された時点で存在するデータベースの現行状態に対して、トランザクションが適用しようとする変更を格納するために、デルタページャが使用する、１つまたは複数のオブジェクトを含むことができる。トランザクションがデータベースにコミットされた場合、新しい現行状態を形成するために、データベースのオリジナルの現行状態に部分マッピングが付加される。新しく適用された部分マッピングを介して新しい現行状態にアクセスされた場合、部分マッピングにおけるアドレスへの値の割り当ては、オリジナル状態におけるそれらよりも優先されるため、データベースのオリジナル状態におけるデータの値は効果的に上書きまたは変更される。

【００１２】

本説明で使用される用語を明確にするためおよび区別するために、データベースのマッピングとは、アドレス値の、データ値への割り当てを言い表す。データベースの完全マッピングとは、データベース内のすべてのアドレスの、それらのアドレスで格納されたデータ値（またはヌル値）への割り当ての、完全なセットを言い表す。部分マッピングとは、トランザクションがデータを書き込もうとするアドレス、およびトランザクションがアドレスに書き込もうとするデータ値を含む、トランザクションがデータベースに対して実行するかまたは実行しようとする変更または上書きのセットを言い表す。トランザクションがデータベースにコミットされた場合、部分マッピングを以前の完全マッピングに追加することが可能であり、結果としてデータベースの現在のマッピングが更新される。したがって、データベースの現在の完全マッピングは一連の部分マッピング（ａ　ｓｅｒｉｅｓ　ｏｆ　ｐａｒｔｉａｌ　ｍａｐｐｉｎｇｓ）を含むことが可能であり、後で追加された部分マッピングは、以前の部分マッピングに含まれるアドレスの割り当てを上書きまたは変更することができる。

【００１３】

データベースの状態とは、データベースのマッピングが格納される、メモリ、ディスク、または大容量記憶装置に格納されたオブジェクトの集まりを言い表す。状態内のオブジェクトは、トランザクションがデータベースに書き込もうとする変更、ならびにバッファを内部に集めることができる他のオブジェクトを格納するために、デルタページャがトランザクションに対して作成するバッファを含む。バッファ、他のオブジェクト、ならびに、これらのオブジェクトが結合、合体、および他の方法で処理される方法について、以下で詳細に説明する。データベースの現行状態は、データベースの現在の完全マッピングを提示するそれらのオブジェクトを含む。以下で説明するように、デルタページャは、トランザクションが開始された時点で存在するデータベースの状態などの、他の状態も維持する。デルタページャは、トランザクションをデータベースにコミットすべきであるかどうかを判別するために、この状態を追跡する。デルタページャは、それらの状態、ならびに後で使用するためにそれらの状態が表すデータベースのマッピングを保持するために、選択された状態も維持する。

【００１４】

各トランザクションが開始されると、デルタページャは、トランザクションが開始された時点で存在していたデータベースの状態に基づいて、トランザクションを実行させることができる。その後、トランザクションは、書き込みバッファなどの少なくとも１つのオブジェクトを作成し、トランザクションが開始された時点で存在していたデータベースの状態に書き込みバッファをポイントさせる。デルタページャは、一連のバッファを使用することが可能であり、ここで各バッファは、トランザクションがデータベースに適用しようとする単一の変更を格納し、各バッファは、データベースの状態を保持するために先行バッファをポイントする。別の方法として、デルタページャは、トランザクションが実行しようとするすべての変更を、トランザクションが開始された時点のデータベースの現行

状態をポイントする累積書き込みバッファに格納することもできる。トランザクションが開始された時点の状態をポイントするバッファまたは他のオブジェクトは、トランザクションがデータベースに適用しようとする変更の部分マッピングを表す。

【００１５】

１つまたは複数の介在トランザクションが現行トランザクションの適用を妨げているというインディケーションを、デルタページャが見つけなかった場合、デルタページャは、データベースの現行状態の追跡にデルタポインタが使用する現行状態ポインタを、トランザクション用に作成された最後のバッファまたは唯一のバッファをポイントするように変更することによって、トランザクションをコミットする。このバッファをポイントするように現行状態ポインタを変更することによって、データベースの新しい更新された状態を生成し、データベースの完全マッピングにおけるデータ値を変更することができる。

【００１６】

デルタページャは、読み取りバッファおよび書き込みバッファを使用してトランザクションを処理することができる。読み取りバッファは、トランザクションがアクセスするデータを追跡する。現行トランザクションが完了した場合、デルタページャは読み取りバッファを使用して、トランザクションをコミットするかどうかを決定する。読み取りバッファが、介入トランザクションによって上書きまたは変更されたデータを現行トランザクションが読み取ったことを示す場合、デルタページャはトランザクションを打ち切ることができる。

【００１７】

デルタページャがデータベースへの複数の変更をコミットした後、デルタページャは効率性のために、データベースの現行状態においてバッファまたは他のオブジェクトを合体することができる。デルタページャは、１つの部分マッピングにおける複数のオブジェクトを、他の部分マッピングにおけるオブジェクトと合体することができる。２つの部分マッピングのうちの新しい方におけるアドレスの割り当ては、古い方の部分マッピングに含められた割り当てに追加されるか、またはこれと置き換えられることになる。合体の結果、メモリオブジェクトまたは永続オブジェクトを含む１つまたは複数の合体オブジェクトが生じる可能性があるか、あるいはデルタページャは、バッファに格納された変更をデータベースのオリジナル状態に適用することによってバッファを合体することができる。デルタページャは、選択された状態を保持できるようにするためにスナップショットポインタを提供する。スナップショットポインタに関連付けられたいかなる状態も、スナップショットポインタによって保持された状態に対して実行された変更を表すバッファと合体されることはない。

【００１８】

デルタページャは、データベースおよびデータベースに対して実行された変更を、永続的にすることができる。デルタページャは、データベースの現行状態を維持するために使用された１つまたは複数のオブジェクトを、永続的な不揮発性ストレージにコピーすることができる。デルタページャは、デルタページャが永続的にするために不揮発性ストレージに移動した現行状態の一部を含む、大容量記憶媒体に格納されたデータベースの現行状態の一部へのより高速なアクセスを提供するための、キャッシュオブジェクトも備える。デルタページャは、領域照会を容易にするために、このキャッシュオブジェクトを使用してヌルデータの領域をデータベースの状態に格納することができる。

【００１９】

書き込みバッファを使用したトランザクションの処理

図１は、データベースの現行状態１００を示す。現行状態１００は、データベースの例示的なオリジナル状態１１０を含む。データベースは、揮発性メモリまたは不揮発性メモリに常駐することができる。デルタページャは、現行状態ポインタ１２０を使用して、図１ではオリジナル状態１１０のみを含む現行状態をポイントすることができる。

【００２０】

オリジナル状態１１０は、レコード０　１３０、レコード１　１４０、レコード２　１

５０、レコード３　１６０、およびレコード４　１７０の、それぞれが現在はヌル値を格納している５つのレコードを維持するオブジェクトである。ヌル値は、実際にオリジナル状態に格納されている必要がないことに留意されたい。またオリジナル状態１１０は、トランザクションが変更を適用する可能性のある、ヌルおよび非ヌルの値を含む、任意の初期状態を含むことができる。

【００２１】

図２は、複数の命令２１０～２７０を含む例示的トランザクションである、トランザクション１　２００を示す。命令１　２２０、「ｘ＝Ｒｅａｄ（３）」などのいくつかの命令は、データベースからデータを読み取ろうとする。命令３　２４０、「Ｗｒｉｔｅ（２，ｘ）」などの他の命令は、結果としてデータベース内のアドレスに値を割り当てるかまたは再割り当てすることになる、データベースへの変更を適用しようとする。

【００２２】

図３は、データベースに対してトランザクション１　２００などのトランザクションを開始する、デルタページャの１つのモードを示す。トランザクション１　２００は、トランザクション開始命令、ｔｒａｎｓ＿ｓｔａｒｔ　２１０で開始される。トランザクション１　２００が開始された後、現行状態ポインタ１２０は、トランザクションが開始された時点のデータベースの状態、すなわちオリジナル状態１１０を、引き続きポイントする。デルタページャは第１の書き込みバッファ３１０を作成し、デルタページャは、トランザクション１　２００が適用しようとする変更をここに格納することになる。デルタページャは、現行状態ポインタ１２０によって示された、トランザクション２　２００が開始された時点の状態に、書き込みバッファ３１０をポイントさせる。デルタページャはトランザクションポインタ３２０も作成し、このトランザクションポインタ３２０を第１の書き込みバッファ３００にポイントさせ、この第１の書き込みバッファは現行状態をポイントする。

【００２３】

図４～６は、デルタページャがトランザクション１　２００を処理する方法を示す。図２を参照すると、命令１　２２０、「ｘ＝Ｒｅａｄ（３）」は、ヌルである現行値ｘをアドレス３から読み取る。命令２　２３０、「ｘ＝ｘ＋１」は、ｘの値を１だけ増分し、このケースではｘの値を１に変更する。命令３　２４０、「Ｗｒｉｔｅ（２，ｘ）」は、コンピューティングシステムにｘの値をアドレス２に書き込むように命じる。

【００２４】

デルタページャは値１をオリジナル状態１１０のアドレス２には書き込まない。その代わりに図４に示されるように、デルタページャは第２の書き込みバッファ４１０を作成する。第２の書き込みバッファ４１０は第１の書き込みバッファ３１０をポイントし、デルタページャは、トランザクション２　２００が、トランザクションが開始された時点のデータベースの状態を修正する方法を示す、データベースの部分マッピングを維持するために、トランザクションポインタ３２０が第２の書き込みバッファ４１０をポイントするように切り替える。デルタページャは、このアドレス２への値１の書き込みの変更を、第２の書き込みバッファ４１０に格納する。

【００２５】

図５は、トランザクション１　２００がデータベースに実行しようとするデルタページャの第２の変更への応答を示す。命令４　２５０、「Ｗｒｉｔｅ（４，「ＤＯＧ」）」は、文字列「ＤＯＧ」をアドレス４に格納するようにコンピューティングシステムに命じる。ここでも、変更をオリジナル状態１１０に書き込む代わりに、デルタページャは、第２の書き込みバッファ４１０をポイントする第３の書き込みバッファ５１０を作成する。さらにデルタページャは、トランザクション１　２００がこれを修正しようとした場合にデータベースの状態を保持するために、トランザクションポインタ３２０が第３の書き込みバッファ５１０をポイントするように変更し、文字列「ＤＯＧ」を第３の書き込みバッファ５１０に格納する。したがって、トランザクションポインタ３２０から読み取る、トランザクション２１２００に関するデータベースの部分マッピングは、アドレス２が値１を

格納し、アドレス４が文字列「ＤＯＧ」を格納するという修正を含む。しかしながら、トランザクションは、オリジナル状態１１０も、現行状態ポインタ１２０によって示される現行状態も変更していない。

【００２６】

図６は、デルタページャがトランザクション１　２００をデータベースにコミットする方法を示す。トランザクションポインタ３２０は、書き込みバッファ５１０、４１０、および３１０を通るか、または現行状態ポインタ１２０がポイントするのと同じ状態をポイントするため、デルタページャは、トランザクション１　２００がデータベースにコミットされないようにするために、いかなる介入トランザクションもデータベースの現行状態を変更していないものと判別する。したがってデルタページャは、書き込みバッファ３１０、４１０、および５１０によって修正された場合にオリジナル状態１１０を含む、トランザクション１　２００に対して作成された部分マッピングをポイントするように現行状態ポインタ１２０を変更することによって、トランザクション１２００をデータベースにコミットする。したがってデルタページャは、トランザクション１　２００によって追加された最新の書き込みバッファである、第３の書き込みバッファ５１０をポイントするように、現行状態ポインタ１２０を変更する。次にデルタページャは、トランザクションポインタ３２０を削除する。これで現行状態ポインタ１２０は、書き込みバッファ３１０、４１０、および５１０に格納された変更によって更新されたオリジナル状態１１０を含む、データベースの更新された状態をポイントすることになる。

【００２７】

図６の現行状態を使用して、トランザクションがデータベースにアクセスする場合、デルタページャは、現行状態ポインタ１２０によって識別される、トランザクションが開始される時点で存在するデータベースの状態を使用する。この現行状態を使用して、トランザクションがバッファ５１０内でアドレス４のコンテンツを識別しようとした場合、文字列「ＤＯＧ」を見つけることになる。他方で、トランザクションがアドレス２のコンテンツを求めた場合、バッファ５１０内には見つからないことになる。したがって、トランザクションはバッファ４１０に進み、ここでトランザクションは値１を見つけることになる。図６の例で、トランザクションが他の値を求めた場合、オリジナル状態１１０が他のアドレスについてヌル値のみを含むことを見つけるまで、トランザクションはすべてのバッファにアクセスすることになる。このようにしてデルタページャは、データベースの現行状態にバッファまたは他のオブジェクトを追加することによって、データを変更する。

【００２８】

デルタページャの不変条件（ｉｎｖａｒｉａｎｔ）とは、デルタページャが作成したそれぞれのポインタの不変性を遵守することである。不変性の不変条件とは、各オブジェクトが依存する状態、したがって各オブジェクトがポイントする状態が、ポインタがそのオブジェクトをポイントする限りは変更されないことを維持するものである。たとえば、第１の書き込みバッファ３１０は、データベースのオリジナル状態１１０をポイントする。その後の変更にかかわらず、トランザクション１　２００はデータベースに適用し、第１の書き込みバッファ３１０は、オリジナル状態１１０のみを含むデータベースの状態を常にポイントする。同様に、第２の書き込みバッファ４１０は第１の書き込みバッファ３１０をポイントする。第２の書き込みバッファ４１０は、オリジナル状態１１０およびオリジナル状態１１０を修正しない書き込みバッファ３１０を含む、データベースの不変状態もポイントする。第３の書き込みバッファ５１０は、第２の書き込みバッファ４１０をポイントするため、第２の書き込みバッファ４１０によって修正された場合のオリジナル状態１１０を含む、データベースのそれ独自の部分マッピングをポイントする。これらのポインタの不変性は、以下で説明する利点を有する。

【００２９】

累積書き込みバッファおよび単一書き込みバッファの再書き込み

今説明したモードでは、デルタページャは、トランザクションを開始するため、およびトランザクションがデータベースに適用しようとする各変更を格納するために、新しい書

き込みバッファを作成する。別の方法として、デルタページャは、累積バッファなどの異なるタイプのオブジェクトを採用することができるか、またはデルタページャは、１つまたは複数の変更を格納するために既存のバッファを上書きすることができる。

【００３０】

図７は、累積書き込みバッファを使用してトランザクションを処理するデルタページャを示す。図３の例と同様に、デルタページャは、現行状態ポインタ１２０によって示されるように、トランザクションが開始されたときに存在した現行状態をポイントする第１の累積バッファ７１０を作成することによって、トランザクション１　２００を開始する。デルタページャは、トランザクションポインタ３２０を再度作成し、これを、最も新しく作成された書き込みバッファにポイントさせて、トランザクション１　２００によって修正されるようにデータベースの部分マッピングを維持する。

【００３１】

図８は、第１の累積バッファ７１０をポイントする第２の累積バッファ８１０を追加することによってトランザクションの処理を続行する、デルタページャを示す。デルタページャは、トランザクションポインタ３２０を、最も新しく作成された書き込みバッファである第２の累積バッファ８１０にポイントさせる。デルタページャは、トランザクション１　２００がデータベースに適用しようとする、アドレス２の値を１に設定する変更を、第２の累積バッファ８１０に格納する。トランザクション１　２００はこれまで１つの変更しか試みてこなかったため、第２の累積バッファ８１０は累積的には見えない。

【００３２】

図９は、第２の累積バッファ８１０をポイントする第３の累積バッファ９１０を追加することによってトランザクション１　２００の処理を続行する、デルタページャを示す。デルタページャは、トランザクションポインタ３２０を、最も新しく作成された書き込みバッファである第３の累積バッファ９１０にポイントさせる。ここでデルタページャは、トランザクション１　２００がデータベースに適用しようとする、アドレス２の値を１に設定する変更と、アドレス４で文字列「ＤＯＧ」を格納する変更の両方を格納する。

【００３３】

図１０および図１１は、デルタページャが累積書き込みバッファを使用するモードの利点を示す。図１０は、第３の累積バッファ９１０に格納されたトランザクション１　２００によって求められたすべての変更と共に、トランザクション１　２００がデータベースに適用することになるすべての変更の部分マッピングが、単一のオブジェクトである累積書き込みバッファ９１０に含められることを示す。図１０の例に示されるように、部分マッピングは、たとえ累積バッファ７１０および８１０が除外された場合であっても保持される。

【００３４】

不変性の不変条件により、いかなるトランザクションもトランザクションの中間状態に干渉またはアクセスすることができないため、トランザクションポインタ３２０によって示される状態の整合性を維持するために書き込みバッファ７１０および８１０を保存する必要はない。結果として、累積書き込みバッファ７１０および８１０が格納した変更が、最も新しく作成された累積バッファ９１０にも格納された後で、累積書き込みバッファ７１０および８１０を維持する必要はない。

【００３５】

したがって、図１１が示すように、デルタページャが新しい累積バッファを作成し、そのそれぞれに書き込む場合、デルタページャは先行するバッファを破棄することができる。第１の段階１１００は、アドレス２の値を１に設定するという第１の変更が第２の累積バッファ８１０に書き込まれた後の、トランザクション１　２００の現行状態を示す。第２の累積バッファ８１０は、第１の書き込みバッファ７１０に格納されたいずれの変更も含むため、第２の段階１１１０でデルタページャは、第２の累積バッファ８１０を、第１の累積バッファ７１０がポイントした状態にポイントさせ、トランザクションポインタ３２０によって示された状態を変更せずに、第１の累積バッファ７１０およびそのポインタ

を解放することができる。トランザクションポインタ３２０がポイントする状態は、第２の累積バッファ８１０に格納された同じデータのサブセットのみを含むバッファを通過しないため、より短い。第３の段階１１２０で、デルタページャは、第１の累積バッファ７１０を格納するために使用されたメモリを解放し、他に使用するためにメモリを確保することができる。デルタページャが、トランザクション１　２００の次の変更を格納するために、図１１に示されたものと同様の段階を適用した場合、結果は図１０に示されたものと同じになる。

【００３６】

図１２は、デルタページャがトランザクションを処理するために使用するモードを示す。流れ図１２００は、ブロック１２０２がトランザクションを受け取ると開始される。ブロック１２０４は、トランザクションがデータベースに適用しようとする任意の変更をポイントすることになる、トランザクションポインタを作成する。ブロック１２０６は、第１の書き込みバッファを作成し、そのバッファを、トランザクションが受け取られた時点で存在する現行状態にポイントさせる。ブロック１２０８は、トランザクションポインタを第１の書き込みバッファにポイントさせ、第１の書き込みバッファはトランザクションが開始された時点の現行状態をポイントする。

【００３７】

ブロック１２１０は、トランザクションが、たとえばデータベースからデータを読み取るだけではなく、データベースに変更を適用しようとするかどうかを判別する。変更を適用しようとする場合、ブロック１２１２は追加の書き込みバッファを作成し、これを、以前作成されたバッファにポイントさせる。追加の書き込みバッファは、単一書き込みバッファまたは累積書き込みバッファとすることができる。１つのモードで累積書き込みバッファが使用される場合、ブロック１２１２は、追加の累積バッファを、以前のバッファがポイントした状態にポイントさせ、図１１を参照しながら説明したように以前のバッファを解放する。

【００３８】

ブロック１２１４は、トランザクションがデータベースに適用しようとする変更を、追加の書き込みバッファに格納する。ブロック１２１６は、トランザクションポインタを追加の書き込みバッファにポイントさせる。ブロック１２１８は、トランザクションが任意の追加の変更をデータベースに適用しようとするかどうかを判別する。適用しようとする場合、流れ図１２００はブロック１２１２にループし、ブロック１２１２は追加のバッファを作成する。他方で、ブロック１２１８が、トランザクションは追加の変更を適用しようとしないものと判別した場合、後で説明するように、ブロック１２２０は変更のコミットを試行することになる。ブロック１２２４は、他のトランザクションの受け取りを待つ。

【００３９】

他方で、ブロック１２１０が、トランザクションがいかなる変更もデータベースに適用しようとしなかったものと判別した場合、第１の書き込みバッファおよびトランザクションポインタは不要であるため、ブロック１２２２はこれらを解放する。ブロック１２２４は、再度、他のトランザクションの受け取りを待つ。

【００４０】

デルタページャは、一連の累積バッファを使用する代わりに、図１３に示されたような単一の再書き込み可能書き込みバッファを使用することもできる。第１の段階１３００で、デルタページャはトランザクションを受け取り、単一書き込みバッファ１３１０を作成して、これを現行状態にポイントさせる。次にデルタページャは切り替えトランザクションポインタ１３２０を作成し、これを、単一書き込みバッファ１３１０にポイントさせる。切り替えポインタ１３２０は、以下で説明するように、デルタページャ内のポインタの不変性を保持する。このようにバッファに上書きまたは再書き込みすることにより、それぞれの変更を格納するために複数のバッファを使用する代わりに、トランザクションが適用しようとする変更を格納するために単一の累積バッファを使用して、デルタページャに

よって認識される不変性の不変条件を保持する。

【００４１】

単一書き込みバッファ１３１０の使用の第２の段階１３３０で、デルタページャは、切り替えポインタ１３２０を単一書き込みバッファ１３１０を避けてポイントさせ、トランザクションによって示された変更または追加の変更で単一書き込みバッファを更新する。前述のように、デルタページャはすべてのポインタの不変性を保持する。したがって技術的には、切り替えポインタが引き続き単一書き込みバッファ１３１０をポイントする場合、バッファ１３１０に変更を書き込むと、不変性の不変条件が破られることになり、切り替えポインタによって示される状態は変化するように見える。したがって１つのモードでは、デルタページャは、単一書き込みバッファ１３１０を避けてポイントすることが可能な切り替えポインタ１３２０を使用する。

【００４２】

第３の段階１３４０で、デルタページャは単一書き込みバッファを更新すると、トランザクションがデータベースに適用しようとする変更の部分マッピングを維持するために、再度切り替えポインタ１３２０を単一書き込みバッファ１３１０にポイントさせる。第４の段階１３５０で、デルタページャは、再度、切り替えポインタ１３２０を単一書き込みバッファ１３１０を避けてポイントさせ、トランザクションによって示された他の変更を追加するように単一書き込みバッファを更新する。第５の段階１３６０で、デルタページャが追加の変更を単一書き込みバッファに追加すると、デルタページャは部分マッピングを維持するために、再度切り替えポインタ１３２０を単一書き込みバッファ１３１０にポイントさせる。

【００４３】

デルタページャは、実際にはバッファ１３１０を避けてポイントされる切り替えポインタ１３２０を、事実上使用する必要がないことに留意されたい。不変性の不変条件が論理的に保持されるのを示すために、デルタページャは、切り替えポインタ１３２０によって示された再書き込み可能バッファを使用するものとして説明される。別の方法として、システムのどの部分も再書き込みが発生した間ポインタを逆参照（ｄｅｒｅｆｅｒｅｎｃｅ）しないように、再書き込みが原子的に発生した場合、不変条件が依然として有効に保持されている間、通常のトランザクションポインタ３２０を代わりに使用することが可能であり、ポインタは、だれもその値を監視していない場合、ポインタではない。したがってこの限られたケースでは、デルタページャは、不変性の不変条件を保持することからインプレース再書き込みを使用する。

【００４４】

図１４は、デルタページャが単一書き込みバッファを使用することによってトランザクションを処理するために使用する、図１３に示されたモードを示す。流れ図１４００は、ブロック１４０２がトランザクションを受け取ると開始される。ブロック１４０４は、トランザクションがデータベースに適用しようとする変更の部分マッピングを示すことになる、切り替えポインタを作成する。ブロック１４０６は、単一書き込みバッファを作成し、そのバッファを、トランザクションが受け取られた時点の現行状態にポイントさせる。ブロック１４０８は、トランザクションがデータベースに適用しようとする部分マッピングを示すために、切り替えポインタを単一書き込みバッファにポイントさせる。

【００４５】

ブロック１４１０は、トランザクションがデータベースに変更を適用しようとするかどうかを判別する。適用しようとする場合、ブロック１４１２は切り替えポインタを単一書き込みバッファを避けてポイントさせる。ブロック１４１４は、変更を単一書き込みバッファに格納する。図１３に示されるように、単一書き込みバッファは累積的であるため、複数の変更を単一書き込みバッファに追加することができる。ブロック１４１６は、再度切り替えポインタを単一書き込みバッファにポイントさせる。

【００４６】

ブロック１４１８は、トランザクションがデータベースに任意の追加の変更を適用しよ

うとするかどうかを判別する。適用しようとする場合、流れ図１４００はブロック１４１２にループし、ブロック１４１２は追加のバッファを作成する。他方で、ブロック１４１８が、トランザクションが追加の変更を適用しようとしないものと判別した場合、後で説明するように、ブロック１４２０は変更のコミットを試行することになる。ブロック１４２４は、他のトランザクションの受け取りを待つ。

【００４７】

他方で、ブロック１４１０が、トランザクションがデータベースにいかなる変更も適用しようとしなかったものと判別した場合、単一書き込みバッファおよび切り替えポインタは不要であるため、ブロック１４２２はこれらを解放する。ブロック１４２４は他のトランザクションの受け取りを待つ。

【００４８】

トランザクションのコミット

デルタページャは、トランザクションをコミットするタイミングを決定するための異なるモードをサポートする。デルタページャは、複数のトランザクションが、各トランザクションが開始された時点で存在するデータベースの状態に同時にアクセスできるようにする。しかしながら、複数のトランザクションが同時に実行された場合、最初のデータベースへのコミットにより、他の同時に実行しているトランザクションが依拠するデータベースの状態が変更される可能性がある。古くなったかまたは取って代わられたデータベースの状態に基づくトランザクションがデータベースにコミットされると、無効の結果につながる可能性がある。したがってデルタページャは、現行トランザクションが依拠したデータベースの状態に介入トランザクションが干渉しなかった場合にのみ、現行トランザクションをコミットする。

【００４９】

図１５は、デルタページャがトランザクション１　２００と同時に処理し、結果としてトランザクションの競合を発生させる可能性のある、第２のトランザクションである、トランザクション２（Ｔ×２）１５００を示す。競合する可能性のあるトランザクションを考えた場合、デルタページャは、どのトランザクションをデータベースにコミットし、どのトランザクションを打ち切るかを決定する。トランザクション２　１５００は、複数の命令１５１０～１５７０を呼び出す。命令１　１５２０、「ｘ＝Ｒｅａｄ（０）」などのいくつかの命令は、データベースからデータを読み取ろうとするが、命令４　１５５０、「Ｗｒｉｔｅ（３，「ＣＡＴ」）」などの他の命令は、変更をデータベースに適用しようとする。

【００５０】

図１６は、デルタページャがトランザクション１　２００およびトランザクション２　１５００の両方を受け取り、同時に処理する状況を示す。２つのトランザクションのうちのどちらかを、どちらかのトランザクションがデータベースにコミットされる前に、他方よりも先に開始することができる。どちらのトランザクションも、現行状態ポインタ１２０が引き続きオリジナル状態１１０、すなわちその時点での現行状態をポイントするものとして開始された。

【００５１】

デルタページャは、前述のように、現行状態ポインタ１２０によって示されたオリジナル状態１１０をポイントする第１の書き込みバッファ３１０を作成することによって、トランザクション１　２００を開始し、そのトランザクションに関して最も新しく作成された書き込みバッファをポイントするトランザクションポインタ３２０を作成する。デルタページャは、トランザクション１　２００がデータベースに対して実行しようとする変更を格納する、追加の書き込みバッファ４１０および５１０を作成する。

【００５２】

トランザクション２　１５００の場合、デルタページャは、オリジナル状態１１０をポイントする第１の書き込みバッファ１６１０と、そのトランザクションに関して最も新しく作成された書き込みバッファを示すためのトランザクションポインタ１６２０とを作成

する。同様にデルタページャは、トランザクション２　１５００に関する追加の書き込みバッファを作成する。図１６では、命令１　１５２０である「ｘ＝Ｒｅａｄ（０）」、および命令２　１５３０である「ｙ＝Ｒｅａｄ（１）」は、どちらもオリジナル状態１１０からヌル値を読み取る。したがって条件付き命令３　１５４０、「ｉｆ　ｙ＝ｘ　ｔｈｅｎ」は真であり、命令４　１５５０、「Ｗｒｉｔｅ（３，「ＣＡＴ」）」が実行される。デルタページャは、第１の書き込みバッファ１６１０をポイントする追加の書き込みバッファ１６３０を作成し、命令４　１５５０によって示された変更をこの追加の書き込みバッファ１６３０に格納し、トランザクションポインタ１６２０を、この追加の書き込みバッファ１６３０にポイントさせる。トランザクションポインタ３２０およびトランザクションポインタ１６２０は、どちらも、各トランザクションがデータベースに適用することになる変更を含む部分マッピングをポイントする。

【００５３】

トランザクション１　２００およびトランザクション２　１５００は、データベースに対して競合する変更を行う可能性があるか、あるいは、命令３　１５４０などの条件付き命令は、他のトランザクションが変更する可能性のあるデータに依拠する可能性がある。デルタページャのモードは、これらの潜在的な競合を様々な方法で説明する。

【００５４】

介入トランザクションが現行状態を変更した場合のトランザクションの打ち切り

図１７および図１８は、デルタページャがデータベースの状態を変更する介入トランザクションをコミットした後にトランザクションを打ち切ることによって、デルタページャが競合する可能性のあるトランザクションを処理するために使用する１つのモードを示す。図１７では、図６に関して前述したように、デルタページャは、トランザクション１　２００をデータベースにコミットしており、トランザクションポインタ３２０は除去され、現行状態ポインタ１２０は、トランザクションに関して作成された最新の書き込みバッファ５１０をポイントするように切り替えられる。トランザクション１　２００をデータベースにコミットした後、デルタページャは、トランザクション２　１５００が実行を完了したものと決定し、データベースへコミットしようとする。

【００５５】

トランザクション２　１５００をデータベースにコミットするかどうかを決定する場合、デルタページャは、トランザクション１　２００などの介入トランザクションが、現行のトランザクション、すなわちトランザクション２　１５００が開始された時点で存在したデータベースの状態を変更したかどうかを判別する。この判別を行う場合、デルタページャは、第１の書き込みバッファ１６１０がポイントする状態を決定することによって、トランザクション２　１５００が開始された時点のデータベースの状態と比較する。次にデルタページャは、これを、現行状態ポインタ１２０がポイントするデータベースの現行状態と比較する。言い換えれば、デルタページャは、それらが両方とも同じオブジェクトをポイントしているかどうかを判別するために、現行状態ポインタ１２０と、トランザクションポインタ１６２０ならびに挿入された書き込みバッファ１６１０および１６３０とを比較する。トランザクション１　２００をデータベースにコミットした後、デルタページャは、現行状態ポインタ１２０が書き込みバッファ５１０をポイントする一方で、トランザクション２　１５００は、その第１の書き込みバッファ１６１０を通ってオリジナル状態１１０をポイントすることを決定する。

【００５６】

トランザクション２　１５００は、現行状態ポインタ１２０によって示される現行状態以外の状態をポイントするため、デルタページャはトランザクション２　１５００を打ち切る。図１８は、デルタページャが、データベースの現行状態の一部とはならないメモリを解放するために、書き込みバッファ１６１０および１６３０ならびにトランザクションポインタ１６２０を削除することによって、トランザクション２　１５００を打ち切ることを示す。

【００５７】

現行状態におけるいずれかの変更のためにトランザクションを打ち切ることは、競合するトランザクションのコミットを避けるためのリスクを嫌った手法である。図１６～図１８の例に示されるように、トランザクション１　２００は、トランザクション２　１５００が実行された時点で存在したデータベースの状態におけるいかなる値も変更しなかった。しかしながら、介入するトランザクション１　２００が、トランザクション２　１５００が依拠したデータを変更した可能性はある。トランザクション１　２００はデータベースの現行状態を変更したため、トランザクション２　１５００が依拠したデータを変更した可能性がある。

【００５８】

トランザクション２　１５００は打ち切られたが、図１９および図２０に示されるように、このトランザクションは再開することができる。図１９で、デルタページャは、トランザクション１　２００がコミットした後に、トランザクション２　１５００を開始する。したがって、トランザクション２　１５００が開始された場合、デルタページャは、トランザクション２　１５００が再開された時点で存在する状態をポイントする、第１の書き込みバッファ１９１０を作成する。この状態は、トランザクション１　２００によって状態に追加された書き込みバッファ５１０をポイントする、現行状態ポインタ１２０によって示される。次にデルタページャは、元々は書き込みバッファ１９１０をポイントしていたトランザクションポインタ１９２０を作成する。デルタページャは、トランザクション２　１５００がデータベースに適用しようとする変更をデルタページャが格納する追加の書き込みバッファ１９３０を追加し、その後、デルタページャは、トランザクション２　１５００がデータベースに適用しようとする変更を示す部分マッピングを示すために追加の書き込みバッファ１９３０をポイントするように、トランザクションポインタ１９２０を変更する。

【００５９】

今回は、トランザクション２　１５００が完了したものとデルタページャが決定し、データベースにトランザクションをコミットしようとすると、デルタページャには、トランザクション２　１５００が依拠した状態からデータベースの現行状態を変更した介入トランザクションがないことがわかる。トランザクションポインタ１９２０は、書き込みバッファ１９３０および１９１０を通って、書き込みバッファ５１０をポイントする。現行状態ポインタが依然として書き込みバッファ５１０をポイントしているため、トランザクションポインタ１９２０は依然として現行状態をポイントする。

【００６０】

図２０は、デルタページャがトランザクション２　１５００をデータベースにコミットしていることを示す。デルタページャは、トランザクション２　１５００によって追加された最新のバッファ、すなわち書き込みバッファ１９３０をポイントするように、現行状態ポインタ１２０を変更する。次にデルタページャは、トランザクションポインタ１９２０を削除する。現行状態ポインタが現在ポイントしている現行状態は、トランザクション１　２００によって適用された書き込みバッファ４１０および５１０と、トランザクション２　１５００によって適用された書き込みバッファ１９３０とに格納された変更によって修正された場合、オリジナル状態１１０を含む。

【００６１】

図２１は、デルタページャがトランザクションをコミットするかまたは打ち切るかどうかを決定するために使用する、第１のモードを示す。流れ図２１００は、ブロック２１０２で開始される。ブロック２１０２は、完了されたトランザクションがデータベースへのコミットを求める場合を決定し、流れ図２１００は、トランザクションがデータベースへのコミットを求めるまで、ブロック２１０２にループする。ブロック２１０２が、コミットしようとする完了されたトランザクションを検出した場合、ブロック２１０４は、現行状態が、たとえば現行トランザクションに関してトランザクションポインタによって示されるように、トランザクションが開始された時点で存在した状態と同じであるかどうかを判別する。言い換えれば、ブロック２１０４は、トランザクションのために作成された第

１の書き込みバッファが、現行状態ポインタがポイントするのと同じ状態をポイントするかどうかを判別する。ポイントする場合、ブロック２１０６は、現行状態ポインタに、トランザクションポインタもポイントするバッファである、トランザクションのために作成された最新のバッファをポイントさせ、トランザクションがコミットされる結果としてデータベースの更新を完了する。ブロック２１０８はトランザクションポインタを削除し、関連するメモリを解放する。流れ図２１００は、データベースにコミットしようとする次の完了されたトランザクションを検出するために、ブロック２１０２へとループする。

【００６２】

他方で、ブロック２１０４が、現行状態が、トランザクションが開始された時点で存在した状態と同じでないと決定した場合、ブロック２１０４は、現行状態が変更されたものと決定する。ブロック２１１０は、トランザクションの書き込みバッファを解放する。ブロック２１１２は、トランザクションの打ち切りが完了したトランザクションポインタを削除する。前述のように、打ち切られたトランザクションは再開することが可能であり、デルタページャは、トランザクションが完了し、コミットしようとする場合、トランザクションがデータベースにコミットされるべきであるかどうかを判別することになる。

【００６３】

トランザクションが不必要に打ち切られるのを防ぐための読み取りバッファ

トランザクションが現行状態と一致しない状態をポイントする場合に必ず、デルタページャが現行トランザクションを打ち切る代わりに、デルタページャは、現行トランザクションが読み取ったデータを介入トランザクションが書き込んだ場合に、現行トランザクションを打ち切ることができる。現行状態が変更された場合に必ず現行トランザクションを打ち切ることによって、競合する変更がデータベースに適用されないことが保証される。しかしながら、同時に多くのトランザクションが実行している可能性がある場合、および、それらのトランザクションの一部が長いかまたは複雑な計算を含む可能性がある場合、トランザクションの打ち切りは計算リソースを無駄にする。したがって、介入トランザクションが、現行トランザクションが読み取ったかまたは依拠したいかなるデータにもアクセスしなかったかまたはこれを変更しなかった場合、デルタページャが現行トランザクションを打ち切る必要はない。

【００６４】

デルタページャのあるモードは、各トランザクションに対して読み取りバッファを作成する。読み取りバッファは、トランザクションがアクセスする任意のデータを追跡する。完了したトランザクションがデータベースへのコミットを試行するが、介入トランザクションがデータベースの状態を変更したことがわかった場合、デルタページャは、読み取りバッファとデータベースの現行状態とを比較して、現行トランザクションの打ち切りが必要であるかどうかを判別することができる。他のモードでは、デルタページャは、現行状態が変更され、現行トランザクションが読み取ったデータを介入トランザクションが変更または上書きした場合に、現行トランザクションを打ち切る。このモードでは、デルタページャは、介入トランザクションが現行トランザクションによって読み取られたアドレスにデータを書き込んだ場合に、トランザクションを打ち切ることができるか、または、デルタページャは、データの値が実際に変更したかどうか判別するために、現行トランザクションによって読み取られた実際のデータと、介入トランザクションによって書き込まれたデータとを、比較することができる。

【００６５】

図２２～２４は、現行トランザクションによって読み取られたデータを介入トランザクションが書き込んだ場合に、読み取りバッファを使用してトランザクションを打ち切る、デルタページャのモードを示す。図２２は、データベースの現行状態に対して実行される、トランザクション１　２２００およびトランザクション２　２２５０という２つのトランザクションを示す。現行状態ポインタ１２０はオリジナル状態１１０をポイントし、これによって現行状態またはデータベースの現行状態を示す。トランザクション１　２２００の場合、デルタページャは、読み取りバッファ２２１０、書き込みバッファ２２２０、

およびトランザクションポインタ２２３０を作成する。前述のように、デルタページャは、各トランザクションについて１つまたは複数の書き込みバッファを作成することができる。デルタページャは、読み取りバッファがサイズまたはその他の理由で制限されている場合、複数の読み取りバッファを作成することもできる。デルタページャは、読み取りバッファ２２１０を現行状態にポイントさせ、書き込みバッファ２２２０を読み取りバッファ２２１０にポイントさせ、トランザクションポインタ２２３０を書き込みバッファ２２２０にポイントさせる。これに対応して、トランザクション２　２２５０の場合、デルタページャは読み取りバッファ２２６０、書き込みバッファ２２７０、およびトランザクションポインタ２２８０を作成する。デルタページャは、読み取りバッファ２２６０を現行状態にポイントさせ、書き込みバッファ２２７０を読み取りバッファ２２６０にポイントさせ、トランザクションポインタ２２８０を書き込みバッファ２２７０にポイントさせる。

【００６６】

トランザクション１　２２００についての読み取りバッファ２２１０は、トランザクション１　２２００がアドレス３でデータを読み取ったことを示す。トランザクション２　２２５０についての読み取りバッファ２２６０も、トランザクション２　２２５０がアドレス３でデータを読み取ったことを示す。しかしながら、たとえそれぞれのトランザクションが同じデータを読み取ったとしても、どちらのトランザクションもそのアドレスのデータを変更しようとしない。したがって、デルタページャはどちらのトランザクションも打ち切らなくてよい。

【００６７】

空の書き込みバッファと同様に、読み取りバッファは、データベースの状態を変更しない。したがって、デルタページャの諸実施形態は、データが格納されたオブジェクトであるかのように、データベースの現行状態にない読み取りバッファを採用することができる。その代わりに、デルタページャは、トランザクションが適用しようとする変更を格納するためにデルタページャが作成できる部分マッピングの外部に格納することができる。読み取りバッファは、別の場所に格納することが可能であり、トランザクションポインタは、読み取りバッファの場所を示すために別のポインタを維持することができる。

【００６８】

介入トランザクションが現行トランザクションによって読み取られたアドレスのデータを変更する場合にのみ、デルタページャが現行トランザクションを打ち切る場合、図２５～図２７を参照しながら以下で説明するように、読み取りバッファは、読み取ったアドレスおよび値を格納するはずである。他方で、介入トランザクションが現行トランザクションによって読み取られたデータを上書きした場合に、デルタページャがトランザクションを打ち切る場合、たとえ介入トランザクションが同じ値を書き込んだとしても、読み取りバッファは、図２２～図２４に示されるように、トランザクションが読み取る１つまたは複数のアドレスを含むだけでよい。

【００６９】

図２３は、デルタページャがトランザクション１　２２００をデータベースにコミットした後のデータベースの状態を示す。デルタページャは、トランザクション１　２２００について作成された最新の書き込みバッファ２２２０にポイントするように、現行状態ポインタ１２０を変更し、トランザクションポインタ２２３０を削除する。トランザクション２　２２５０が完了すると、デルタページャはトランザクション２　２２５０をコミットすべきかどうかを判別する。ここでも、デルタページャがトランザクション２　２２５０をコミットする前にデータベースの現行状態が変更されているため、図１５～図２１を参照しながら前述したトランザクションをコミットするかどうかを判別するモードで、デルタページャはトランザクション２　２２５０を打ち切ることになる。これに対して、現行のモードは、より現実的な評価をする。

【００７０】

このモードでは、たとえデータベースの現行状態が変更されていても、介入トランザク

ションが現行トランザクションによってアクセスされるアドレスにデータを書き込んでいない限り、デルタページャは現行トランザクションを打ち切らない。デルタページャは、トランザクション２　２２５０がデータを読み取ったアドレスに、介入トランザクションであるトランザクション１　２２００が書き込まれたかどうかを判別するために、トランザクション２　２２５０の読み取りバッファ２２６０を比較し、それによって、トランザクション２　２２５０の処理が依拠した状態を変更する。トランザクション２　２２５０は、読み取りバッファ２２６０に従って、アドレス３からデータを読み取るだけである。しかしながら、現行状態ポインタ１２０によって示された現行状態をチェックすると、トランザクション１　２２００は、書き込みバッファ２２２０に従ってアドレス４にデータを書き込んだのみである。したがってデルタページャは、データベースにトランザクション２　２２５０をコミットする。

【００７１】

図２４は、データベースの現行状態にトランザクション２　２２５０をコミットしているデルタページャを示す。デルタページャは、トランザクション２　２２５０の読み取りバッファ２２６０を、現行状態ポインタによって示されたオブジェクトである、トランザクション１　２２００の書き込みバッファ２２２０にポイントさせる。次にデルタページャは、トランザクション２　２２５０の書き込みバッファ２２７０を指示するように、現行状態ポインタ１２０を変更し、トランザクション２　２２５０をデータベースにコミットする。

【００７２】

図２５～図２７は、介入トランザクションがデータを書き込んだアドレスに介入トランザクションが書き込むだけでなく、現行トランザクションが読み取ったデータ値を介入トランザクションが実際に変更する場合に、デルタページャが現行トランザクションを打ち切るだけの、それほどリスクを嫌わないモードを示す。図２５は、データベースの現行状態に対して実行される、トランザクション１　２５００およびトランザクション２　２５５０の、２つのトランザクションを示す。現行状態は、データベースのオリジナル状態１１０をポイントする、現行状態ポインタ１２０によって示される。トランザクション１　２５００の場合、デルタページャは、読み取りバッファ２５１０、書き込みバッファ２４２０、およびトランザクションポインタ２５３０を作成する。デルタページャは、読み取りバッファ２５１０をその時点での現行状態にポイントさせ、書き込みバッファ２５２０を読み取りバッファ２５１０にポイントさせ、トランザクションポインタ２２３０を書き込みバッファ２５２０にポイントさせる。これに対応して、トランザクション２　２５５０の場合、デルタページャは読み取りバッファ２５６０、書き込みバッファ２５７０、およびトランザクションポインタ２５８０を作成する。デルタページャは、読み取りバッファ２５６０をその時点での現行状態にポイントさせ、書き込みバッファ２５７０を読み取りバッファ２５６０にポイントさせ、トランザクションポインタ２５８０を書き込みバッファ２５７０にポイントさせる。

【００７３】

図２２～図２４の例とは対照的に、トランザクション１　２５００についての読み取りバッファ２５１０は、トランザクション１　２５００がアドレス３の値を読み取ったこと、およびアドレス３から読み取られた値がヌルであったことを示す。トランザクション２　２５５０についての読み取りバッファ２５６０は、トランザクション２　２５００もアドレス３の値を読み取ったこと、および、アドレス３から読み取られた値もヌルであったことを見つけたことを示す。トランザクション２　２５５０についての書き込みバッファ２５７０は、アドレス３の値への変更を格納し、格納された値を文字列「ＣＡＴ」に変更する。

【００７４】

図２６は、デルタページャがトランザクション２　２５５０をデータベースにコミットした後のデータベースの現行状態を示す。したがってデルタページャは、トランザクション２　２５５０がデータベースに適用するいずれかの変更を格納している書き込みバッフ

ァ２５７０をポイントするように、現行状態ポインタ１２０を変更し、トランザクションポインタ２５８０を破棄する。このモードでは、トランザクション１　２５００が完了すると、デルタページャは、トランザクション２　２５５０が変更した値をトランザクション１　２５００が読み取ったかどうかを判別する。

【００７５】

デルタページャは、トランザクション１　２５１０の読み取りバッファ２５１０と、現行状態ポインタ１２０によって示されるデータベースの現行状態とを比較する。読み取りバッファ２５１０によれば、トランザクション１はアドレス３を読み取り、値がヌルであることがわかった。さらにデルタページャは現行状態ポインタ１２０によって示されるデータベースの現行状態を追い、トランザクション２　２５５０がアドレス３で格納された値を「ＣＡＴ」に変更したことを見つける。したがって、トランザクション１　２５００がデータベースに適用しようとするいかなる変更も、今では古くなったデータに基づいている可能性がある。

【００７６】

したがって、デルタページャは、図２７に示されるようにトランザクション１　２５００を打ち切る。デルタページャは、読み取りバッファ２５１０、書き込みバッファ２５２０、およびトランザクションポインタ２５３０を解放または削除し、デルタページャは、現行状態ポインタ１２０を、トランザクション２　２５５０がデータベースに適用した変更を格納している書き込みバッファ２５７０をポイントしたままにする。所望であれば、トランザクション１　２２００を前述のように再開することができる。

【００７７】

たとえトランザクション２　２５５０がトランザクション１　２２００によって読み取られたアドレスに新しい値を書き込んだとしても、トランザクション２　２２５０が書き込んだデータがトランザクション１　２５００の結果を変更している可能性はないことに留意されたい。しかしながら、トランザクション１　２５００の結果が、トランザクション２　２５５０によって書き込まれたデータの結果として変更されるかどうかを判別するために、トランザクション１　２５００を再実行しなければならない。デルタページャのこのモードは、現行状態が変更されているというだけで、トランザクション１　２５００を打ち切らず、トランザクション１　２５５０の再実行にかかる時間が足りない現行トランザクションを打ち切るべきであるかどうかのいくつかの実質的な分析を提供する。

【００７８】

図２８は、デルタページャがトランザクションをコミットすべきであるかどうかを判別するより実質的なモードを容易にするために、読み取りバッファおよび単一書き込みバッファを作成することによって、デルタページャがトランザクションを処理するために使用するモードを示す。デルタページャは、読み取りバッファの使用と共に単一書き込みバッファの使用に限定されておらず、単一書き込みバッファの選択は、単なる１つの可能な代替である。

【００７９】

流れ図２８００は、ブロック２８０２がトランザクションを受け取ると開始される。ブロック２８０４は、トランザクションがデータベースに適用しようとする変更の部分マッピングを示すことになる、切り替えポインタを作成する。ブロック２８０６は読み取りバッファを作成し、これを、トランザクションが開始された時点で存在した状態にポイントさせる。ブロック２８０８は単一書き込みバッファを作成し、この単一書き込みバッファを読み取りバッファにポイントさせる。ブロック２８１０は、トランザクションが適用しようとする変更の部分状態を示すために、切り替えポインタを単一書き込みバッファにポイントさせる。

【００８０】

ブロック２８１２は、トランザクションによって読み取られたすべてのデータを読み取りバッファに格納する。前述のように、介入トランザクションが現行トランザクションと同じデータを読み取った場合、デルタページャがトランザクションを打ち切ることができ

るモードでは、読み取りバッファは単に読み取ったデータのアドレスを格納すればよい。他方で、現行トランザクションによって読み取られたデータを介入トランザクションが書き込んだかどうかに基づいて、トランザクションをコミットするかどうかをデルタページャが判別する場合、読み取りバッファはアドレスとそのアドレスから読み取ったデータとの両方を格納すべきである。

【００８１】

ブロック２８１４は、トランザクションがデータベースに変更を適用しようとするかどうかを判別する。適用しようとする場合、ブロック２８１６は、切り替えポインタを単一書き込みバッファを避けてポイントさせる。ブロック２８１８は、単一書き込みバッファへの変更を格納する。ブロック２８２０は、切り替えポインタを再度単一書き込みバッファにポイントさせる。

【００８２】

ブロック２８２２は、トランザクションが完了するかまたは実行を継続するかを判別する。トランザクションが完了しない場合、流れ図２８００はブロック２８１２にループして、トランザクションによって読み取られたデータがあれば読み取りバッファに格納する。他方で、ブロック２８２２がトランザクションを完了すると判別した場合、ブロック２８２４はトランザクションによって実行された変更のコミットを試行することになる。ブロック２８２６は他のトランザクションの受け取りを待つ。

【００８３】

図２９は、デルタページャが読み取りバッファを使用して、トランザクションをコミットするかまたは打ち切るかを判別するモードを示す。流れ図２９００はブロック２９０２で開始される。ブロック２９０２は、完了されたトランザクションがデータベースをコミットしようとする場合を決定し、流れ図２９００はトランザクションがデータベースにコミットしようとするまでブロック２９０２へとループする。ブロック２９０２が、完了されたトランザクションがコミットしようとするのを検出すると、ブロック２９０４は、トランザクションが開始されて以来、現行状態が変更されたかどうかを判別する。

【００８４】

現行状態が変更されていない場合、ブロック２９１２は現行状態ポインタを、単一書き込みバッファ、または、トランザクションポインタも指示しているバッファである、トランザクションについて作成された複数の書き込みバッファのうちの最新のバッファにポイントさせる。ブロック２９１４は、トランザクションポインタを削除し、データベースの更新を完了する。データベースの現行状態が変更されていない場合、デルタページャは、読み取りバッファの検査またはデータベースへの適用の検索なしに、トランザクションをコミットすることができる。現行状態が変更されていない場合、いかなる介入トランザクションも現行トランザクションが依拠するデータを変更することはない。流れ図２９００は、データベースにコミットしようとしている次の完了されたトランザクションを検出するために、ブロック２９０２へとループする。

【００８５】

他方で、ブロック２９０４が、トランザクションが開始されて以来、現行状態が変更されていると決定した場合、ブロック２９０６は、読み取りバッファ内に列挙されたデータが現行状態で変更されているかどうかを判別する。ここでも、デルタページャのこのモードは、デルタページャが介入トランザクションをコミットする結果として、現行状態が変更されているかどうか、読み取りバッファを検査するのみである。読み取りバッファ内に列挙されたデータが現行状態で変更されていない場合、デルタページャはトランザクションをコミットすることになり、流れ図２９００はブロック２９１２へと進む。

【００８６】

他方で、ブロック２９０６が現行状態で、読み取りバッファ内のアドレスにデータが書き込まれたことを見つけた場合、または、そのアドレスでのデータの値が読み取りバッファ内に格納されたものと異なる場合、デルタページャはトランザクションを打ち切る。ブロック２９０８は、トランザクションについての読み取りおよび書き込みバッファを解放

する。次に、ブロック２９１０は、現行トランザクションについてのトランザクションポインタを解放または削除し、トランザクションの打ち切りを完了する。次に、流れ図２９００は、ブロック２９０２へとループして、コミットしようとする次のトランザクションを待つ。

【００８７】

トランザクションの合体および状態の保持

デルタページャの書き込みバッファは、多くの利点を提供する。一例を挙げると、トランザクションがコミットできず、打ち切らなければならない場合、デルタページャは、そのトランザクションについて作成した書き込みバッファを無視または削除し、デルタページャは、そうでなければデータベースに適用された可能性のある誤った変更または競合する変更を再書き込みするかまたは取り消す必要はない。しかしながら、デルタページャが多くの書き込みバッファを現行状態に追加した後では、現行状態におけるすべてのデータへのアクセスは非効率的になる可能性がある。たとえ比較的単純な図２０の例であっても、トランザクションは、トランザクションが読み取ろうとしてデータの値を決定するために、書き込みバッファ１９３０、１９１０、５１０、４１０、３１０、およびオリジナル状態１１０を通って再度読み取らなければならない可能性がある。デルタページャの１つのモードでは、データベースの現行状態への効率的なアクセスを保証するために、書き込みバッファを合体する。

【００８８】

図３０は、デルタページャが図２０の書き込みバッファを合体することができる１つのモードを示す。図３０で、デルタページャは、書き込みバッファに格納された変更をデータベースのオリジナル状態１１０に適用することによって、書き込みバッファを合体する。第１の段階３０００で、デルタページャは、書き込みバッファが合体されることになるポイントを示す合体ポインタ３０１０を認識する。合体ポインタ３０１０の利点について、以下で詳細に説明する。デルタページャは、次に、オリジナル状態１１０をポイントする第１の書き込みバッファ内に格納された変更を適用する。図３０に示されるように、第１の書き込みバッファは空の書き込みバッファ３１０であった。デルタページャは、この空の書き込みバッファを現行状態から除外することによって、書き込みバッファ３１０などの空のバッファを合体する。

【００８９】

空の書き込みバッファ３１０は、トランザクションがデータベースに適用しようとする変更を表す部分マッピングを変更しないことに留意されたい。したがって、デルタページャが、トランザクションが適用しようとする変更を格納する書き込みバッファを作成するとすぐに、空の書き込みバッファが破棄される可能性があるか、またはデルタページャが空の書き込みバッファ３１０を作成さえしない可能性もある。たとえば、アドレス４の値を変更するために書き込みバッファ５１０が追加されると、デルタページャの実施形態は、書き込みバッファ５１０を、トランザクションが開始された時点で存在する状態にポイントさせ、空の書き込みバッファ３１０を破棄する可能性がある。

【００９０】

第２の段階３０２０で、デルタページャは、次の書き込みバッファ４１０に格納された変更をオリジナル状態１１０に適用し、オリジナル状態を更新済み状態３０３０に置き換える。書き込みバッファ４１０に格納された、アドレス２の値を１に設定する変更は、更新済みバッキングストア３０３０を作成するために、アドレス２　３０４０に適用される。

【００９１】

最終結果３０５０は、データベースの新しい状態３０６０を示す。新しい状態３０６０では、書き込みバッファ５１０に格納された変更が、アドレス４　３０７０で格納された値を文字列「ＤＯＧ」に変更するために適用され、書き込みバッファ１９３０に格納された変更が、アドレス３　３０８０で格納された値を文字列「ＣＡＴ」に変更するために適用された。デルタページャは、ここで、新しい状態３０６０をポイントするように現行状

態ポインタ１２０を切り替える。その後デルタページャは、合体ポインタ３０１０を除外する。

【００９２】

デルタページャのバッファの合体に関して注目する４つのポイントがある。第１に、たとえ合体プロセス中であっても、各ポインタはデータベースの同じ状態を不変的にポイントし続ける。たとえば第２の段階３０２０で、デルタページャが、更新済み状態３０３０を作成するために、書き込みバッファ４１０内の変更をオリジナル状態１１０に適用した場合、他のポインタによって示される状態も同様に維持された。現行状態ポインタ１２０、書き込みバッファ５１０、１９１０および１９３０、ならびに合体ポインタ３０１０は、すべて、すべて同じ値を提示するデータベースの状態を依然としてポイントしていた。

【００９３】

第２に、現行のデータベースを変更するために１つまたは複数の新しいトランザクションが開始されコミットされた場合、新しいトランザクションはデータベースの不変状態をポイントし続けることになった。新しいトランザクションを開始すると、デルタページャは、現行状態ポインタ１２０によって示された現行状態をポイントする書き込みバッファを作成する。デルタページャがトランザクションをコミットする場合、デルタページャは、それぞれの後続トランザクションについて作成された最新の書き込みバッファをポイントするように、現行状態ポインタ１２０を切り替える。したがってデルタページャは、たとえ書き込みバッファの合体中であっても、新しいトランザクションに関する状態の不変性を保持する。

【００９４】

第３に、合体ポインタ３０１０は合体プロセスが停止するポイントを示す。デルタページャが追加の書き込みバッファを現行状態にコミットする場合、デルタページャは、デルタページャが他の合体に従事するまで追加のバッファを合体しない。合体は無限に続けることができるが、数個の可能性がある新しい書き込みバッファを継続的に合体するために計算リソースを充当することは、計算リソースの効率的な使用ではない可能性がある。

【００９５】

第４に、図３１に示されるように、デルタページャは、データベースのオリジナル状態１１０を上書きすることなしにバッファを合体することができる。たとえばオリジナル状態１１０は、アクセスが非効率的な可能性があるディスクストレージに格納することができるか、または後続のトランザクションによって変更が適用されることなく、オリジナル状態１１０を維持することが望ましい可能性がある。したがって、デルタページャは、１つまたは複数の中間オブジェクトに書き込みバッファを合体することができる。

【００９６】

図３１で、書き込みバッファを中間オブジェクトに合体する第１の段階３１００では、デルタページャは、合体される第１の書き込みバッファである書き込みバッファ３１０と、オリジナル状態１１０との間に、中間の合体済みオブジェクト３１１０を作成する。合体済みオブジェクト３１１０の挿入は、データベース内のいかなる現行状態の不変性も変更せず、現行状態ポインタ１２０および書き込みバッファ３１０、４１０、５１０、１９１０、および１９３０は、依然として同じデータベースの状態をポイントする。合体済みオブジェクト３１１０ならびにオリジナル１１０は、キャッシュメモリ、メモリ、ディスク、または他の形のストレージを含む、任意の所望のストレージ内に維持することができる。

【００９７】

最終結果３１２０は、デルタページャが、書き込みバッファ４１０、５１０、および１９３０に格納されたすべての変更を、合体済みオブジェクト３１１０に合体したことを示す。したがって、書き込みバッファ４１０に格納された、アドレス２の値を１に変更するという変更は、オリジナル状態１１０に適用される代わりに、アドレス２　３１３０で合体済みオブジェクト３１１０に格納される。同様に、デルタページャは、書き込みバッファ５１０および１９３０からの変更を、それぞれアドレス３　３１４０およびアドレス４

３１５０で、合体済みオブジェクト３１１０に格納する。デルタページャは、合体済みオブジェクト３１１０を指示するように現行状態ポインタ１２０を変更し、この合体済みオブジェクト３１１０はオリジナル状態１１０をポイントする。したがって、データベースの状態は決して変更されず、現行状態ポインタ１２０は、デルタページャが書き込みバッファを合体済みオブジェクト３１１０に合体する前に存在したデータベースと同じ状態を引き続きポイントする。

【００９８】

有利なことに、データを読み取ろうとしている新しいトランザクションは、ここで、合体済みオブジェクト３１１０およびオリジナル状態１１０という２つのデータストアをチェックするだけでよい。後者が望ましい場合、デルタページャは、合体済みオブジェクト３１１０のコンテンツをオリジナル状態に合体し、更新済みバッキングストアを作成することができる。別の方法として、デルタページャは、合体済みオブジェクトを維持し、同じ合体済みオブジェクト３１１０への書き込みバッファの後続の合体を実行することができる。さらに別の方法として、スナップショットに関して以下で詳細に説明するように、デルタページャは追加の合体済みオブジェクトを作成することができる。

【００９９】

前述のように、デルタページャはポインタの不変性を維持する。データベース内の合体済みオブジェクトでは、各オブジェクトが追加された時点で存在した中間状態を表すバッファ間のポインタは、現行状態が保持される限り消去される。しかしながら、後続バッファからのポインタ以外のポインタがバッファをポイントする場合、デルタページャによって認識される不変性は、ポインタによって示される状態がバッファの合体後も存続しなければならないことを保証する。デルタページャのポインタの不変性を利用して、デルタページャは、選択された状態をポイントするスナップショットポインタを遵守し、たとえスナップショットポインタの先行または後続のバッファをデルタページャが合体する場合であっても、その状態を保持することを遵守する。

【０１００】

図３２は、デルタページャがスナップショットポインタ３２１０と共に保持することになるデータベースの現行状態を示す。現行状態ポインタ１２０は、書き込みバッファ３２２０、３２３０、および３２４０によってオリジナル状態１１０が修正された、現行状態をポイントする。第１の書き込みバッファ３２２０は、文字列「ＰＥＡＲ」をアドレス０に書き込む。第２の書き込みバッファ３２３０は、値１をアドレス２に書き込む。第３の書き込みバッファ３２４０は、文字列「ＤＯＧ」をアドレス４に書き込む。

【０１０１】

デルタページャが、現行状態を保持するように命じられた場合、デルタページャは、現行状態をポイントするスナップショットポインタ３２１０を作成する。デルタページャは、たとえば定期的なバックアップまたはユーザ要求を自動的に保持するプログラムが、現行状態などの状態のスナップショットを要求した場合、状態を保持するためにスナップショットポインタ３２１０を挿入する。スナップショットポインタ３２１０は選択された状態をポイントすることから、デルタページャによって遵守されるポインタの不変性の結果として、デルタページャはポインタおよび対応する状態を維持することになる。

【０１０２】

図３３は、追加のトランザクションが現行状態を変更した後の、データベースのその後の状態を示す。第４の書き込みバッファ３３１０は、文字列「ＰＥＡＣＨ」をアドレス０に書き込む。第５の書き込みバッファ３３２０は、文字列「ＣＡＴ」をアドレス４に書き込む。デルタページャは、第５の書き込みバッファ３３２０をポイントするように現行状態ポインタ１２０を切り替える。図３３で、第５の書き込みバッファ３３２０は、アドレス４に格納された書き込みバッファ３２４０の文字列「ＤＯＧ」を、文字列「ＣＡＴ」で上書きする。しかしながら、スナップショットポインタ３３１０は、図３２の状態を表す書き込みバッファ３２４０を依然としてポイントする。

【０１０３】

図３４は、デルタページャによって許可された合体を示す。デルタページャがオリジナル状態１１０を上書きしないケースを想定すると、デルタページャは、書き込みバッファ３２２０、３２３０、および３２４０によって適用された変更を集めることによって、スナップショットポインタ３２１０と、第１の合体済みオブジェクト３４１０内のオリジナル状態のバッキングストアとの間で、データベースの現行状態におけるオブジェクトを合体する。次にデルタページャは、書き込みバッファ３３１０および３３２０によって適用された変更を第２の合体済みオブジェクト３３２０内に集める。現行状態ポインタ１２０は、この第２の合体済みオブジェクト３４２０をポイントする。第２の合体済みオブジェクト３４２０は、第１の合体済みオブジェクト３４１０をポイントし、これはオリジナル状態１１０をポイントする。

【０１０４】

デルタページャは、スナップショットオブジェクト３２１０のポインタの不変性を保持するために、別々の合体済みオブジェクト３４１０および３４２０を使用する。スナップショットオブジェクト３２１０が存在する限り、デルタページャは図３２の選択された状態を維持することになる。したがって、たとえば図３３および図３４の現行状態では、デルタページャは、書き込みバッファ３２４０によってアドレス４に格納された文字列「ＤＯＧ」を、書き込みバッファ３３２０によってアドレス４に格納された文字列「ＣＡＴ」で置き換える。しかしながら、スナップショットポインタ３２１０によって保持された選択された状態では、文字列「ＤＯＧ」は依然として第１の合体済みオブジェクト３４１０内のアドレス４に格納されたままである。

【０１０５】

スナップショットポインタ３２１０に関して注目する５つのポイントがある。第１にデルタページャは、スナップショットポインタを追加することによって、任意の選択された時点でデータベースの状態を保持することができる。第２にデルタページャは、複数の状態を保持するために複数のスナップショットポインタを含むことができる。第３に、合体に関して前述したように、デルタページャは、ポインタの不変性により、１つまたは複数の以前の状態を保持しながら、トランザクションの受け取りおよびデータベースへの変更のコミットを続行することができる。第４に、スナップショットポインタ３２１０によって保持された状態がもはや必要でない場合、スナップショットポインタ３２１０は除去され、後続の合体によって保持された状態が解放される。

【０１０６】

第５に、スナップショットポインタは、デルタページャが複数の異なる状態を単一のデータストア内で維持できるようにする。現行状態を維持するために従来のデータベースのバックアップイメージが保存される場合、後続のイメージが以前のイメージを上書きするため、各イメージは別々に格納される。複数の状態は、大量のストレージを消費する可能性があり、バックアップイメージは、高速デバイス内のストレージスペースを保持するために、しばしば低速ストレージデバイスへ追いやられる。しかしながら、デルタページャは以前の状態の上に後続の状態を構築することから、スナップショットポインタは単一のデータストア内に以前の状態を保持する。

【０１０７】

図３５は、デルタページャが、バッファまたは、以前に作成されてここでさらに合体されることになる合体済みオブジェクトなどの、他のオブジェクトを合体する際に使用する１つのモードを示す。流れ図３５００はブロック３５０２で開始される。ブロック３５０２は、データベースの現行状態でオブジェクトの合体を開始する。プログラムが周期的に合体を開始するか、またはユーザが合体を開始することができる。ブロック３５０４は、図３４の例にあるようなオリジナル状態、最新の合体済みオブジェクト、またはデルタページャがデータベースに追加されたバッファを合体することになる他のポイントを含むことができる、合体開始ポイントを識別する。ブロック３５０６は、次の合体されていないオブジェクトに進む。図３３の例で、ブロック３５０４は、オリジナル状態１１０から始まり、次の合体されていないオブジェクトである書き込みバッファ３２２０として、合体

のための開始ポイントを識別する。

【０１０８】

ブロック３５０８は、次に作成されるバッファから以外のバッファへのいずれかのポインタが存在するかどうかを判別する。デルタページャは、データベースの状態が変更されない場合、一連のバッファまたは他のオブジェクトを合体する際にポインタを消去することができるが、デルタページャは、スナップショットポインタ３２１０からのポインタなどの他のポインタは消去しない。ブロック３５０８が、次のバッファから以外のポインタが存在しないと判別した場合、ブロック３５１０は、このバッファと次に作成されるバッファとを合体済みオブジェクト内に組み入れる。ブロック３５１２は、合体する可能性のある追加のオブジェクトが存在するかどうかを判別する。追加のオブジェクトは次に作成されるバッファを含む可能性があるか、または、データベース内のオブジェクトが事前に合体されている場合は、さらに合体されることになる合体済みオブジェクトが存在する可能性がある。合体する可能性のある追加のオブジェクトが存在する場合、流れ図３５００はブロック３５０６へとループして、次の合体されていないオブジェクトに進む。しかしながらブロック３５１２は、合体する他のオブジェクトがないと決定する場合がある。たとえば、デルタページャがデータベースを完全に合体した場合、デルタページャは、合体されていない現行状態ポインタ１２０に到達することになる。ブロック３５１２が、合体する追加のオブジェクトが存在しないと決定した場合、ブロック３５１４は合体を完了する。合体の完了により、たとえば動作の完了を確認するメッセージを送信またはログ記録する場合がある。

【０１０９】

他方で、ブロック３５０８は、状態内の次のオブジェクト以外からのポインタが存在すると決定する場合がある。たとえば図３２～図３４に示されるように、スナップショットポインタ３２１０は、オブジェクトが表す状態を保持するためのオブジェクトをポイントすることができる。ブロック３５０８が、他のポインタが存在すると決定した場合、流れ図３５００はブロック３５１２へと進み、合体する可能性のある追加のオブジェクトが存在するかどうかを判別する。

【０１１０】

状態の永続化

デルタページャは、データベースの選択された部分を永続的にすることができる。デルタページャは、データベースの選択された部分を保持するために、不揮発性ストレージにコミットすることになる。

【０１１１】

図３６は、永続媒体であるディスク３６１０上に部分的に、および揮発性媒体であるメモリ３６２０内に部分的に格納された、オブジェクトを含む、データベースの現行状態を示す。現行状態ポインタ１２０は、ディスク３６１０上に格納されたオリジナル状態１１０を含み、デルタページャが書き込みバッファ３６３０および３６４０内の変更をコミットした、現行状態をポイントする。

【０１１２】

１つのモードで、デルタページャは、永続ポインタ３６５０を挿入することによって、プログラムまたはユーザがデータベースの選択した部分を永続的にできるようにする。永続ポインタ３６５０は、永続的にされるオブジェクトを含むデータベースの状態をポイントする。図３６で、永続ポインタ３６５０は、第１の書き込みバッファ３６３０を含む状態をポイントする。

【０１１３】

図３７が示すように、デルタページャは、永続ポインタ３２５０によって示される状態に含まれたオブジェクトがあれば、ディスクストレージ３７１０などの永続格納可能ストレージに移動する。この例では、第１の書き込みバッファ３６３０はディスクストレージ３７１０に移動される。デルタページャは、データベースの状態を、任意数のストレージ媒体にわたって常駐させることができる。したがって、デルタページャが第１の書き込み

バッファ３７３０を書き込むディスクストレージ３７１０は、オリジナル状態１１０が常駐するディスクストレージ３６１０と同じストレージデバイスとするか、または別のストレージデバイスとすることができる。永続ポインタ３６５０がポイントする状態に含まれていない第２の書き込みバッファ３６４０は、メモリ３６２０内に残される。

【０１１４】

デルタページャが、永続ポインタ３６５０によって示されるデータベースの現行状態の一部を永続ストレージに格納すると、デルタページャは永続ポインタを解放することができる。デルタページャは、データベースの選択された部分が永続的にされていることを確認するメッセージを生成することができる。

【０１１５】

図３８は、デルタページャがバッファを永続的にする際の１つのモードを示す。流れ図３５００はブロック３８０２で開始される。ブロック３８０２は永続要求を受け取る。ブロック３８０４は、永続的にするデータベースの部分を決定するために永続ポインタを配置する。ブロック３８０６は、永続ポインタと、永続的にされるデータベース内の最新の永続ストレージ部分との間の、すべてのオブジェクトを識別する。ブロック３８０８は、永続ポインタと最新の永続ストアとの間のすべてのオブジェクトを、永続ストレージオブジェクトにコピーする。ブロック３８１０は、永続的にされる現行状態の部分をポイントするいかなるポインタも、永続ストレージオブジェクトに変更する。図３７の例では、デルタページャは、書き込みバッファ３６４０のポインタを永続オブジェクトに変更する。別の方法として、たとえば、現行状態ポインタ１２０またはスナップショットポインタ３２１０などの、永続ポインタ３６５０がポイントする状態への他のポインタが存在する場合、また、これらのポインタは、ブロック３８０８によって作成された永続オブジェクトへポイントされる。

【０１１６】

永続ポインタ３６５０がポイントする状態内のオブジェクトが永続オブジェクトにコピーされ、この状態へのポインタが変更されると、ブロック３８１２は、メモリを他の用途に解放するために、永続ポインタがポイントするメモリ内のオブジェクトを解放する。ブロック３８１４は、永続ポインタを削除する。ブロック３１６は、たとえば、永続要求を実行するメッセージをユーザに送信すること、または、メッセージをシステムログに追加することによって、永続要求の完了を確認する。

【０１１７】

データのキャッシュ

デルタページャは、性能を向上させるためにデータのキャッシュも提供する。デルタページャは、図７～図９を参照しながら説明した一連の累積バッファを使用するなどにより、データの重複ストレージを可能にする。状態内でのデータの重複ストアが状態を変更することはない。したがってデルタページャは、状態を変更することなく、ディスクストレージまたは他の大容量記憶装置などの低速ストア上に格納されたデータを、メモリなどの高速ストア内にキャッシュすることができる。

【０１１８】

図３９は、キャッシュオブジェクト３９１０を含むデータベースの現行状態を示す。キャッシュオブジェクトポイント３９１０は、ディスクストレージ３９５０に格納された永続オブジェクト３９３０などの、キャッシュされるオブジェクトをポイントする。デルタページャは、このケースでは読み取りバッファ３９６２である、永続オブジェクト３９３０に格納されていない第１のオブジェクトを、キャッシュオブジェクト３９１０にポイントさせる。

【０１１９】

現行状態では、トランザクションがアドレス０の値を検索すると、トランザクションは、文字列「ＣＡＴ」を現行状態に書き込む書き込みバッファ３９６８内のアドレス０の現行値を見つけることになる。トランザクションは、書き込みバッファ３９６８以外は探す必要がなく、トランザクションはメモリ取り出し速度でアドレス０の値を受け取ることに

なる。他方で、トランザクションがアドレス２に格納された値を求める場合、デルタページャはこれを見つけるまで現行状態全体にわたってこの値を求めることになる。キャッシュオブジェクト３９１０がない場合、デルタページャはディスクストレージ３９５０内の永続オブジェクト３９３０まで探しに行くことになる。ディスクストレージ３９５０からのデータへのアクセスは、メモリ３９６０へのアクセスに比べて遅い。キャッシュオブジェクト３９１０をメモリ３９３０内に作成することで、アクセス効率を向上させることができる。

【０１２０】

図３９の例では、永続オブジェクト３９３０は、値１を格納するアドレス２　３９３２、文字列「ＣＡＴ」を格納するアドレス３　３９３４、および文字列「ＤＯＧ」を格納するアドレス４　３９３６などの、アドレスの範囲で値を格納する。永続オブジェクト３９３０は、文字列「ＲＯＣＫ」を格納するアドレス９９　３９３８までの、データが格納されないヌル値の範囲３９４０も格納する。

【０１２１】

デルタページャは、メモリ３９６０内にキャッシュオブジェクト３９１０を作成する。デルタページャは、キャッシュオブジェクト３９１０を、このケースでは永続オブジェクト３９３０である、キャッシュされているオブジェクトにポイントさせる。次にデルタページャは、第１の非永続オブジェクトである読み取りバッファ３９６２を、キャッシュオブジェクト３９１０にポイントさせる。

【０１２２】

デルタページャがキャッシュオブジェクトを作成し、現行状態にキャッシュオブジェクトを挿入するようにポインタを変更すると、デルタページャはキャッシュを読み込む（ｐｏｐｕｌａｔｅ）。１つのモードでは、デルタページャが永続オブジェクト３９３０からデータを取り出す場合、デルタページャは同じデータをキャッシュオブジェクト３９１０に格納する。また、永続オブジェクトへのアクセスに時間を費やした後、デルタページャは永続オブジェクト３９３０からデータのブロックを取り出し、これをキャッシュオブジェクト３９１０に格納することもできる。たとえば、キャッシュオブジェクト３９１０が読み込まれる前に、デルタページャが読み取りバッファ３９６２を作成したトランザクションはアドレス４からのデータを求め、これが永続オブジェクト３９３０内のアドレス４　３９３６に格納される。永続オブジェクト３９３０にアクセスしている間、デルタページャは、アドレス２　３９３２、アドレス３　３９３４、およびアドレス９９　３９３８までの他のアドレスからのデータを含むブロックに関するデータを取り出し、そのデータをキャッシュオブジェクト３９１０内のそれらのアドレスに格納することもできる。

【０１２３】

デルタページャが永続オブジェクト３９３０から取り出したデータをキャッシュオブジェクト３９１０に格納した後、キャッシュオブジェクト３９１０にコピーされたデータに関する後続の要求は、かなり速いメモリ速度で実行されることになる。したがって、その後、読み取りバッファ３９６２が作成されたトランザクションがアドレス９９にデータを要求した場合、デルタページャはキャッシュオブジェクト３９１０内のアドレス９９　３９１８からデータを取り出すことができる。同様に、読み取りバッファ３９６４が作成されたトランザクションがアドレス３にデータを要求した場合、デルタページャは、ディスクストレージ３９５０からデルタページャがデータを取り出すのを待つことなく、キャッシュオブジェクト内のアドレス３　３９１４からデータを取り出すことができる。

【０１２４】

デルタページャのキャッシュオブジェクト３９１０に関して注目する３つの特徴がある。第１に、キャッシュオブジェクト３９１０の挿入によって、デルタページャがデータ状態を作成および使用する方法が変更されないことである。たとえば、たとえキャッシュオブジェクト３９１０内のアドレス４　３９１６が文字列「ＤＯＧ」を格納している場合でも、書き込みバッファ３９６６は、書き込みバッファ３９６６内のアドレス４に文字列「ＢＩＲＤ」を書き込むことによって、データベースの状態を変更することができる。後続

のトランザクションでは、現行状態ポインタ１２０がポイントする現行状態は、キャッシュオブジェクト３９１０がアドレス４　３９１６に何を格納しているかにかかわらず、文字列「ＢＩＲＤ」がアドレス４に格納されていることを見つけることになる。

【０１２５】

第２に、キャッシュオブジェクト３９１０を使用する状態が永続的になっている場合、デルタページャは、キャッシュオブジェクト３９１０を再開および再読み込みする必要がない。その代わりに、データが永続ストレージにコピーされるため、ディスクにコピーされている状態内のデータとキャッシュオブジェクト３９１０内に格納されたデータとが異なる場合、デルタページャはキャッシュオブジェクト３９１０内のデータを更新することができる。別の方法としてデルタページャは、永続ストレージにコピーされている状態と比べた場合、もはや現行ではない、キャッシュオブジェクト３９１０内のエントリを無効にすることができる。どちらの技法も、古い状態のキャッシュを修正して、更新済み状態の有効キャッシュとする。このように実行することによって、デルタページャはキャッシュオブジェクトを無期限に再使用する。キャッシュが更新全体にわたって保持できない場合、更新が永続的にされた場合に必ずキャッシュを再構築することで効率性が失われる。

【０１２６】

第３に、デルタページャは、範囲の照会を容易にするためにキャッシュオブジェクト３９１０を使用することができる。キャッシュオブジェクトは、範囲の照会をバッキングストアに発行することによって、ヌルのラン（ｒｕｎｓ　ｏｆ　ｎｕｌｌ）を効率的に発見し、デルタページャによって維持されるキャッシュオブジェクトまたは他のオブジェクトの一部でこうしたランをコンパクトに表すことができる。たとえば、ヌルのランの表現をキャッシュオブジェクトのフィールドに含めることができる。メモリに格納されたオブジェクト内でこの表現を維持することによって、範囲の照会を容易にするためのヌルのランの表現へのアクセス速度が改善される。たとえば、トランザクションが「ＢＩＲＤ」、「ＣＡＴ」、または「ＤＯＧ」の後に次のペットの文字列を見つけようとすると、「ＤＯＧ」の後の永続オブジェクト３９３０内にヌルデータの長い文字列３９４０が存在した場合、キャッシュオブジェクト３９１０は、範囲の照会をサポートするために、アドレス５　３９２０のように、ヌル値を格納しているラン内のアドレスの最後、または、非ヌル値を格納しているアドレスの最初を示す、第１のヌルアドレスフィールド内のエントリを含むことができる。

【０１２７】

図４０は、デルタページャがキャッシュオブジェクトを維持する際の１つのモードを示す。流れ図４０００はブロック４００２で開始される。ブロック４００２は、メモリまたは他の高速アクセスストレージ内にキャッシュオブジェクトを作成する。ブロック４００４はこのキャッシュオブジェクトを、キャッシュされているオブジェクトにポイントさせる。ブロック４００６は、キャッシュされているオブジェクトをポイントするオブジェクトを識別し、このオブジェクトをキャッシュオブジェクトにポイントさせる。ブロック４００８は、キャッシュオブジェクト内にキャッシュされているオブジェクトからトランザクションによって読み取られたアドレスおよびデータを格納する。前述のように、デルタページャが、キャッシュされているオブジェクトからデータを取り出す場合、好ましくは、デルタページャは、後続のディスクアクセス動作の必要性を潜在的に回避するために、読み取られたアドレスに隣接するアドレスのブロックからデータを取り出すことになる。

【０１２８】

ブロック４０１０は、キャッシュオブジェクトをポイントする状態が永続的とされるかどうかを判別する。永続的とされない場合、流れ図４０００は、トランザクションによって読み取られたアドレスおよびデータを引き続きキャッシュオブジェクトに格納するために、ブロック４００８へとループする。しかしながら、ブロック４０１０が、キャッシュをポイントする状態が永続的とされるものと決定した場合、ブロック４０１２は、永続的とされるオブジェクト内の変更を使用してキャッシュエントリを更新する。別の方法として、デルタページャは、古くなったキャッシュエントリを更新する代わりに無効化するこ

とができる。ブロック４０１２がキャッシュエントリを更新すると、ブロック４０１４は、永続的とされない第１のオブジェクトをキャッシュオブジェクトにポイントさせる。

【０１２９】

デルタページャによって使用されるオブジェクト

要約すると、図４１は、前述の例示的動作を容易にするためにデルタページャが使用するオブジェクトを示す。この要約に表された汎用オブジェクトを参照するために、新しい参照番号が使用されている。図４１は、永続ディスクストレージ４１２０およびメモリ４１３０に格納されたようなオブジェクトの範囲を含む、例示的状況４１００を示す。

【０１３０】

デルタページャは、ディスクストレージ４１２０内の１つまたは複数の永続オブジェクト４１２２および４１２４を使用することができる。これらのオブジェクトは、オリジナル状態および前述のように永続的とされた任意の他の状態を格納することができる。たとえば永続ポインタ４１２６は永続的とされており、それがポイントする状態が永続オブジェクト４１２４内に顕在化（ｍａｎｉｆｅｓｔ）された後に除去されることになるため、破線で示されている。永続オブジェクトは、１つまたは複数のデバイスに格納することができる。

【０１３１】

デルタページャは、メモリ４１３０に格納された複数のオブジェクトを含むことができる。デルタページャ４１４０は、キャッシュ可能な永続オブジェクト４１２２および４１２４と後続のオブジェクトとの間に挿入された、キャッシュオブジェクト４１４０を含むことができる。スナップショットポインタ４１５２によって保持された状態を含むスナップショットオブジェクト４１５０は、キャッシュオブジェクト４１４０をポイントする。スナップショットオブジェクト４１５０は、合体済みオブジェクトまたは一連の合体されていないオブジェクトを含む、選択された状態を含む。前述のように、スナップショットポインタ４１５２は、たとえ保持された状態に含まれるかまたは保持された状態に後で追加されるオブジェクトが合体される場合であっても、ある状態を保持することができる。

【０１３２】

最新の合体動作で合体済みオブジェクト４１６０が生成されて以来、読み取りバッファ４１７０および書き込みバッファ４１８０などの複数のバッファが現在の状態に追加されている。現行状態ポインタ４１９０はこの現行状態を示す。

【０１３３】

例示的諸実施形態を実施するための動作環境

図４２は、デルタページャを実施するための例示的動作環境４２００を示す。動作環境４２００は好適な動作環境の一例に過ぎず、前述のようなデルタページャの諸実施形態例または他の諸実施形態の使用または機能の範囲に関して、いかなる制限をも示唆することを意図するものではない。さらに動作環境４２００は、例示的動作環境４２００に示された構成要素のうちのいずれか１つまたはいずれかの組み合わせに関する、いかなる依存性または要件を有するものとも解釈されるべきではない。

【０１３４】

デルタページャの実施プロセスは、動作環境４２００内で実行される、プログラムモジュールなどのコンピュータ実行可能命令の一般的なコンテキストで説明することができる。通常、プログラムモジュールは、特定のタスクを実行するかまたは特定の抽象データ型を実施する、ルーチン、プログラム、オブジェクト、構成要素、データ構造などを含む。さらに、当業者であれば、デルタページャを実施するプロセスが、ハンドヘルドデバイス、マルチプロセッサシステム、マイクロプロセッサベースかまたはプログラム可能な大衆消費電化製品、ミニコンピュータ、メインフレームコンピュータ、などを含む、様々なコンピュータシステム構成を使用して実施可能であることを理解されよう。デルタページャを実施するプロセスは、通信ネットワークを介してリンクされたリモート処理デバイスによって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。分散コンピューティング環境では、メモリストレージデバイスを含むローカルとリモートの両方の

コンピュータストレージ媒体内に、プログラムモジュールを配置することができる。

【０１３５】

図４２を参照すると、デルタページャのプロセスを実施するための例示的動作環境４２００は、処理ユニット４２２０と、システムメモリ４２３０と、システムメモリ４２３０を含む様々なシステム構成要素を処理ユニット４２２０に結合するシステムバス４２２１とを含む、コンピュータ４２１０を含む。

【０１３６】

コンピュータ４２１０は、通常、様々なコンピュータ読み取り可能媒体を含む。例を挙げると、コンピュータ読み取り可能媒体は、コンピュータストレージ媒体および通信媒体を備えることができるが、これらに限定されることはない。コンピュータストレージ媒体の例には、ランダムアクセスメモリ（ＲＡＭ）、読み取り専用メモリ（ＲＯＭ）、電気的消去可能プログラム可能読み取り専用メモリ（ＥＥＰＲＯＭ）、フラッシュメモリまたは他のメモリ技術、ＣＤ　ＲＯＭ、デジタル汎用ディスク（ＤＶＤ）あるいは他の光またはホログラフィディスクストレージ、磁気カセット、磁気テープ、磁気ディスクストレージ、または他の磁気ストレージデバイス、あるいは、所望の情報を格納するために使用することおよびコンピュータ４２１０によってアクセスすることが可能な任意の他の媒体、が含まれるが、これらに限定されることはない。システムメモリ４２３０は、ＲＯＭ　４２３１およびＲＡＭ　４２３２などの揮発性および／または不揮発性メモリの形のコンピュータストレージ媒体を含む。コンピュータ４２１０内の諸要素間で（起動時などの）情報の転送を助ける基本ルーチンを含む、基本入出力システム４２３３（ＢＩＯＳ）は、通常、ＲＯＭ　４２３１に格納される。ＲＡＭ　４２３２は、通常、処理ユニット４２２０によって即時アクセス可能および／または現在その上で動作中の、データおよび／またはプログラムモジュールを含む。例を挙げると、図４２は、オペレーティングシステム４２３４、アプリケーションプログラム４２３５、他のプログラムモジュール４２３６、およびプログラムデータ４２３７を示すが、これらに限定されることはない。

【０１３７】

コンピュータ４２１０は、他の取り外し可能／取り外し不能で揮発性／不揮発性のコンピュータストレージ媒体も含むことができる。単なる例として挙げると、図４２は、取り外し不能で不揮発性の磁気媒体から読み取るかまたはこれに書き込むハードディスクドライブ４２４１と、取り外し可能で不揮発性の磁気ディスク４２５２から読み取るかまたはこれに書き込む磁気ディスクドライブ４２５１と、ＣＤ－ＲＯＭまたは他の光媒体などの取り外し可能で不揮発性の光ディスク４２５６から読み取るかまたはこれに書き込む光ディスクドライブ４２５５と、を示す。例示的動作環境で使用可能な他の取り外し可能／取り外し不能で揮発性／不揮発性のコンピュータストレージには、磁気テープカセット、フラッシュメモリユニット、デジタル汎用ディスク、デジタルビデオテープ、ソリッドステートＲＡＭ、ソリッドステートＲＯＭ、などが含まれるが、これらに限定されることはない。ハードディスクドライブ４２４１は、通常、インターフェース４２４０などの取り外し不能メモリインターフェースを介して、システムバス４２２１に接続される。磁気ディスクドライブ４２５１および光ディスクドライブ４２５５は、通常、インターフェース４２５０などの取り外し可能メモリインターフェースによってシステムバス４２２１に接続される。

【０１３８】

前述し、図４２に示された、ドライブおよびそれらに関連付けられたコンピュータストレージ媒体は、コンピュータ読み取り可能命令、データ構造、プログラムモジュール、およびコンピュータ４２１０向けの他のデータの、ストレージを提供する。たとえばハードディスクドライブ４２４１は、オペレーティングシステム４２４４、アプリケーションプログラム４２４５、他のプログラムモジュール４２４６、およびプログラムデータ４２４７を格納するように示される。これらの構成要素は、オペレーティングシステム４２３４、アプリケーションプログラム４２３５、他のプログラムモジュール４２３６、およびプログラムデータ４２３７と同じであるか、または異なるものとすることができることに留

意されたい。通常、ＲＡＭに格納されたオペレーティングシステム、アプリケーションプログラムなどは、対応するシステム、プログラム、またはハードディスクドライブ４２４１から読み取られたデータの一部であり、この部分は、所望の機能に応じてサイズおよび範囲が異なる。オペレーティングシステム４２４４、アプリケーションプログラム４２４５、他のプログラムモジュール４２４６、およびプログラムデータ４２４７には、それらが少なくとも異なるコピーであることが可能なことを示すために、ここでは異なる番号が与えられている。ユーザは、キーボード４２６２、一般にはマウス、トラックボール、またはタッチパッドと呼ばれるポインティングデバイス４２６１、無線入力受信構成要素４２６３、あるいは、リモートコントロールなどの無線ソースなどの、入力デバイスを介して、コマンドおよび情報をコンピュータ４２１０に入力することができる。他の入力デバイス（図示せず）は、マイクロフォン、ジョイスティック、ゲームパッド、衛星放送用パラボラアンテナ、スキャナなどを含むことができる。これらおよび他の入力デバイスは、しばしば、システムバス４２２１に結合されるユーザ入力インターフェース４２６０を介して、処理ユニット４２２０に接続されるが、パラレルポート、ゲームポート、ＩＥＥＥ４２９４ポート、またはユニバーサルシリアルバス（ＵＳＢ）４２９８、または赤外線（ＩＲ）バス４２９９などの、他のインターフェースおよびバス構造による接続が可能である。前述のように、入力／出力機能は、通信ネットワークを介した分散様式で容易にすることができる。

【０１３９】

ディスプレイデバイス４２９１も、ビデオインターフェース４２９０などのインターフェースを介してシステムバス４２２１に接続される。ディスプレイデバイス４２９１は、モニタ、ＬＣＤスクリーン、ＴＦＴスクリーン、フラットパネルディスプレイ、従来型テレビジョン、またはスクリーンプロジェクタに限定されることのない、コンピュータ４２１０の出力を表示するための任意のデバイスとすることができる。コンピュータは、ディスプレイデバイス４２９１に加えて、出力周辺インターフェース４２９５を介して接続可能な、スピーカ４２９７およびプリンタ４２９６などの他の周辺出力デバイスを含むこともできる。

【０１４０】

コンピュータ４２１０は、リモートコンピュータ４２８０などの１つまたは複数のリモートコンピュータへの論理接続を使用する、ネットワーク化環境で動作することになる。リモートコンピュータ４２８０は、パーソナルコンピュータとすることが可能であり、通常、コンピュータ４２１０に関して上記で説明した要素のうちの多くまたはすべてを含むが、図４２ではメモリストレージデバイス４２８１のみが示されている。図４２に示された論理接続は、ローカルエリアネットワーク（ＬＡＮ）４２７１およびワイドエリアネットワーク（ＷＡＮ）４２７３を含むが、メトロポリタンエリアネットワーク（ＭＡＮ）、イントラネット、またはインターネットへの接続などの、他のネットワークも含むことができる。

【０１４１】

ＬＡＮネットワーキング環境で使用される場合、コンピュータ４２１０は、ネットワークインターフェースまたはアダプタ４２７０を介してＬＡＮ　４２７１に接続される。ＷＡＮネットワーキング環境で使用される場合、コンピュータ４２１０は、通常、モデム４２７２、またはインターネットなどのＷＡＮ　４２７３を介した通信を確立するための他の手段を含む。モデム４２７２は内蔵型または外付け型とすることが可能であり、ネットワークインターフェース４２７０または他の適切なメカニズムを介してシステムバス４２２１に接続される。モデム４２７２は、ケーブルモデム、ＤＳＬモデム、または他のブロードバンドデバイスとすることができる。ネットワーク化環境では、コンピュータ４２１０に関して示されたプログラムモジュール、またはその一部を、リモートメモリストレージデバイスに格納することができる。たとえば、図４２はリモートアプリケーションプログラム４２８５をメモリデバイス４２８１上に常駐するものとして示しているが、これに限定されることはない。図示されたネットワーク接続は例示的なものであり、コンピュー

タ間に通信リンクを確立する他の手段も使用可能である。

【０１４２】

コンピュータ４２１０の多くの他の内部構成要素は示されていないが、当業者であれば、こうした構成要素および相互接続は周知であることを理解されよう。たとえば、テレビジョンチューナカードおよびネットワークインターフェースカードなどの様々な拡張カードをコンピュータ４２１０内に含めることが一般的である。したがって、デルタページャの実施プロセスの例示的諸実施形態を説明する上で、コンピュータ４２１０の内部構造に関する追加の細部を開示する必要はない。

【０１４３】

コンピュータ４２１０が電源投入またはリセットされた場合、ＲＯＭ　４２３１に格納されたＢＩＯＳ　４２３３は、オペレーティングシステムまたはその必要部分を、ハードディスクドライブ４２４１からＲＡＭ　４２３２へとロードするように、処理ユニット４２２０に命じる。オペレーティングシステム４２４４として指定されたオペレーティングシステムのコピーされた部分がＲＡＭ　４２３２にロードされると、処理ユニット４２２０はオペレーティングシステムコードを実行し、オペレーティングシステム４２３４のユーザインターフェースに関連付けられた視覚要素をディスプレイデバイス４２９１上に表示させる。通常、アプリケーションプログラム４２４５がユーザによって開かれた場合、プログラムコードおよび関連データがハードディスクドライブ４２４１から読み取られ、必要な部分がＲＡＭ　４２３２にコピーされるが、このコピーされた部分は、本明細書では参照番号４２３５によって表される。

【０１４４】

結論

以上、構造的特徴および／または方法論的動作に特有の言葉で例示的諸実施形態について説明してきたが、添付の特許請求の範囲は、必ずしも前述の特定の特徴または動作に限定されるものでないことを理解されよう。むしろ、この特定の機能および動作は、例示的諸実施形態として開示されている。

【図面の簡単な説明】

【０１４５】

【図１】データベースの初期の、現行状態を示す図である。

【図２】データベースを対象とする第１のトランザクションを示す図である。

【図３】バッファを使用してデータベースに対して実行およびコミットされているトランザクションを示す図である。

【図４】バッファを使用してデータベースに対して実行およびコミットされているトランザクションを示す図である。

【図５】バッファを使用してデータベースに対して実行およびコミットされているトランザクションを示す図である。

【図６】バッファを使用してデータベースに対して実行およびコミットされているトランザクションを示す図である。

【図７】累積書き込みバッファを使用して実行されているトランザクションを示す図である。

【図８】累積書き込みバッファを使用して実行されているトランザクションを示す図である。

【図９】累積書き込みバッファを使用して実行されているトランザクションを示す図である。

【図１０】累積書き込みバッファを使用して実行されているトランザクションを示す図である。

【図１１】累積書き込みバッファを使用して実行されているトランザクションを示す図である。

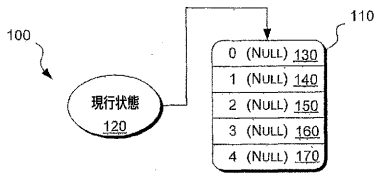
【図１２】書き込みバッファを使用するトランザクションの実行モードを示す図である。

【図１３】単一書き込みバッファを使用して実行されているトランザクションを示す図で

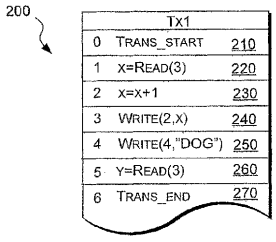
ある。
【図１４】単一書き込みバッファを使用するトランザクションの実行モードを示す図である。
【図１５】第１のトランザクションが介入トランザクションとして実行される間に、データベースに対して実行されている第２のトランザクションを示す図である。
【図１６】介在トランザクションを受けての、現行トランザクションのコミットメントおよび打ち切りを示す図である。
【図１７】介在トランザクションを受けての、現行トランザクションのコミットメントおよび打ち切りを示す図である。
【図１８】介在トランザクションを受けての、現行トランザクションのコミットメントおよび打ち切りを示す図である。
【図１９】介在トランザクションを受けての、現行トランザクションのコミットメントおよび打ち切りを示す図である。
【図２０】介在トランザクションを受けての、現行トランザクションのコミットメントおよび打ち切りを示す図である。
【図２１】介在トランザクションを受けて、トランザクションをコミットするかまたは打ち切るタイミングを決定するモードを示す図である。
【図２２】読み取りおよび書き込みのバッファを使用してデータベースに対して実行およびコミットされている、競合する可能性のあるトランザクションを示す図である。
【図２３】読み取りおよび書き込みのバッファを使用してデータベースに対して実行およびコミットされている、競合する可能性のあるトランザクションを示す図である。
【図２４】読み取りおよび書き込みのバッファを使用してデータベースに対して実行およびコミットされている、競合する可能性のあるトランザクションを示す図である。
【図２５】読み取りおよび書き込みのバッファを使用してデータベースに対して実行およびコミットされている、競合する可能性のあるトランザクションを示す図である。
【図２６】読み取りおよび書き込みのバッファを使用してデータベースに対して実行およびコミットされている、競合する可能性のあるトランザクションを示す図である。
【図２７】読み取りおよび書き込みのバッファを使用してデータベースに対して実行およびコミットされている、競合する可能性のあるトランザクションを示す図である。
【図２８】読み取りおよび書き込みのバッファを使用するトランザクションの実行モードを示す図である。
【図２９】読み取りおよび書き込みのバッファを使用する介在トランザクションを受けて、トランザクションをコミットするかまたは打ち切るタイミングを決定するモードを示す図である。
【図３０】バッファの合体を示す図である。
【図３１】バッファの合体を示す図である。
【図３２】データベースの選択された状態が後続の状態に合体されないように保持するための、スナップショットオブジェクトの使用を示す図である。
【図３３】データベースの選択された状態が後続の状態に合体されないように保持するための、スナップショットオブジェクトの使用を示す図である。
【図３４】データベースの選択された状態が後続の状態に合体されないように保持するための、スナップショットオブジェクトの使用を示す図である。
【図３５】１つまたは複数の選択された状態を保持しながらバッファを合体するモードを示す図である。
【図３６】永続的にされているバッファを示す図である。
【図３７】永続的にされているバッファを示す図である。
【図３８】オブジェクトを永続的にするモードを示す図である。
【図３９】キャッシュオブジェクトの使用を示す図である。
【図４０】キャッシュオブジェクトを作成および維持するモードを示す図である。
【図４１】前述のオブジェクトの集まりを示す図である。

【図４２】前述のデータベースのトランザクションおよび動作を実行するために好適な、例示的動作環境を示す図である。

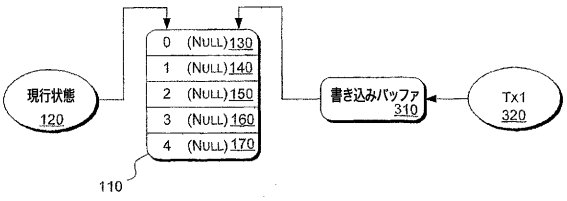
【図１】



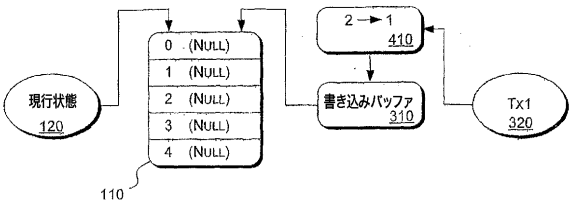
【図２】



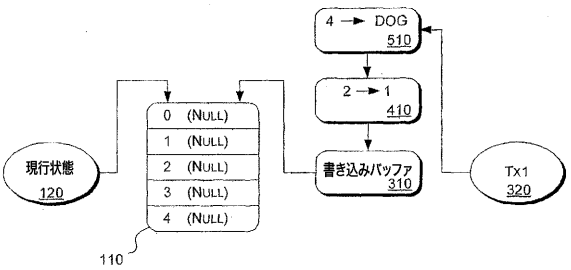
【図３】



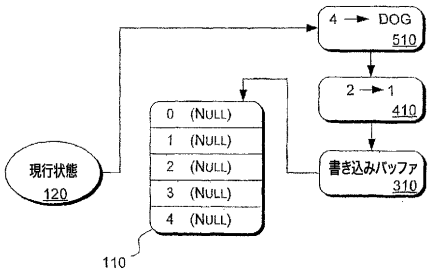
【図４】



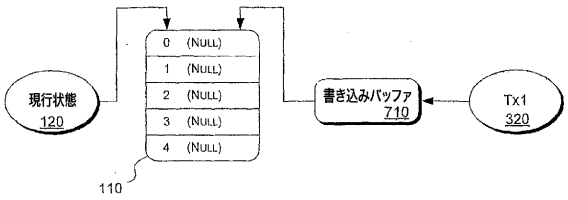
【図５】



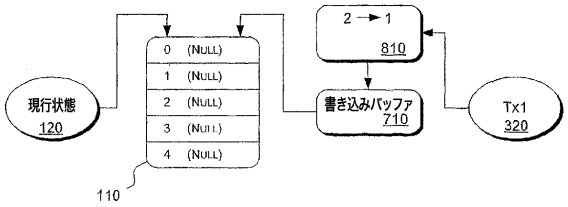
【図６】



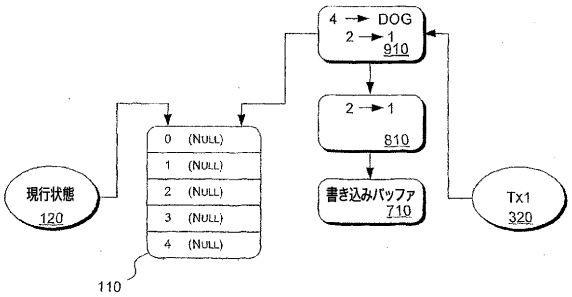
【図７】



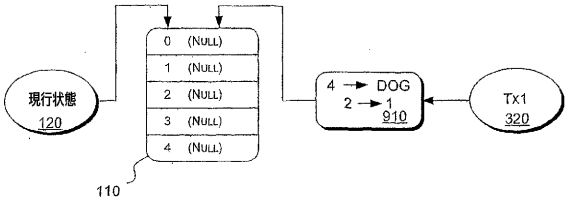
【図８】



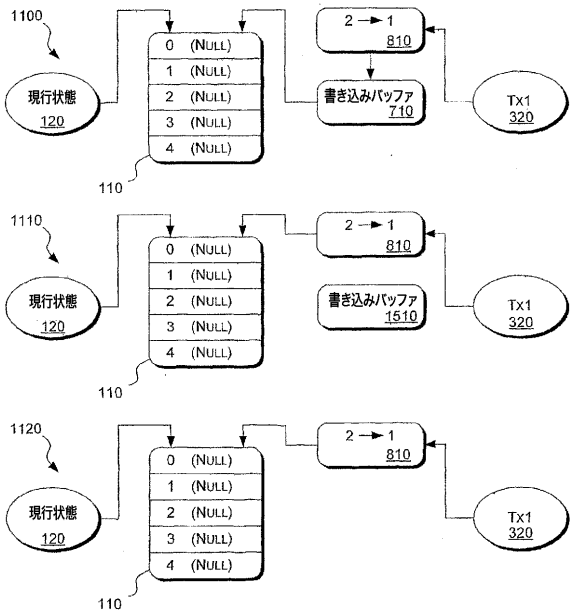
【図９】



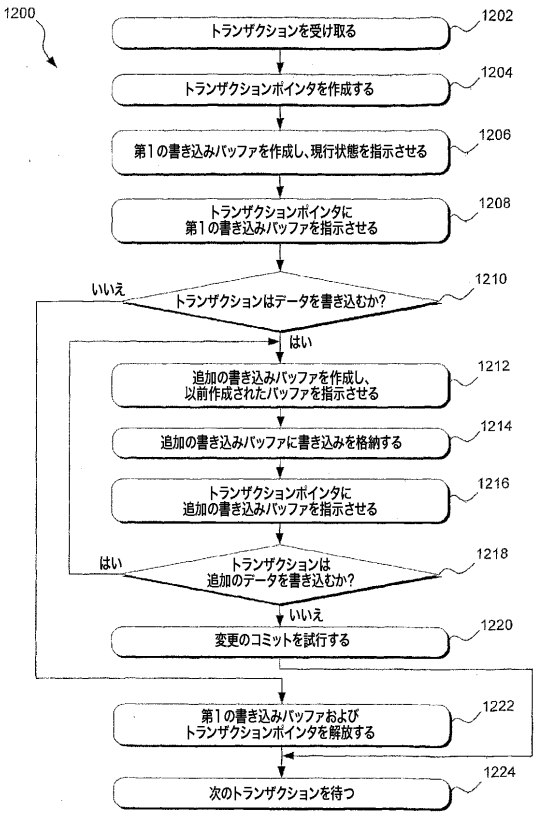
【図１０】



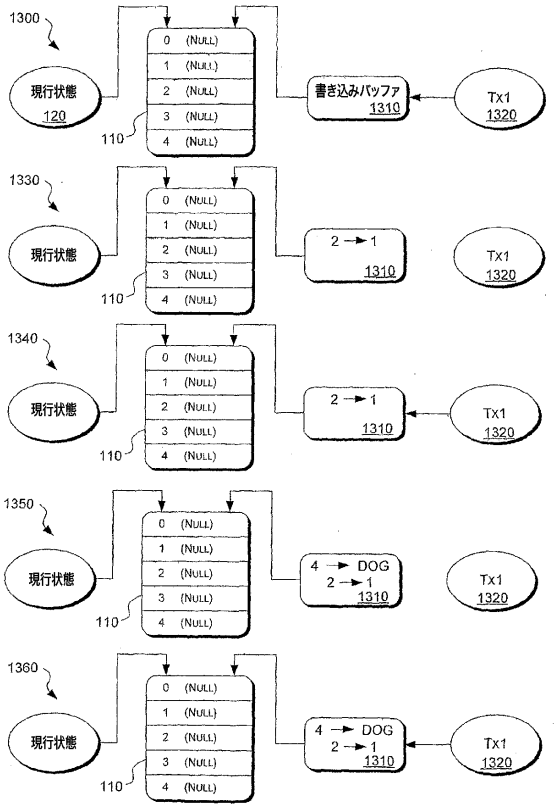
【図１１】



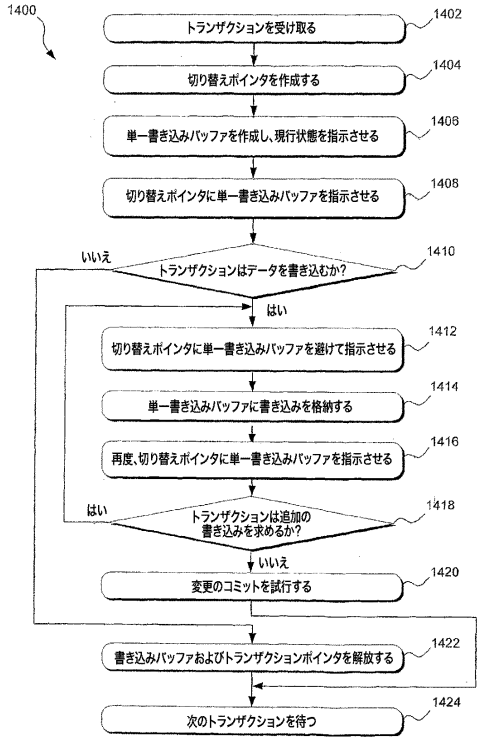
【図１２】



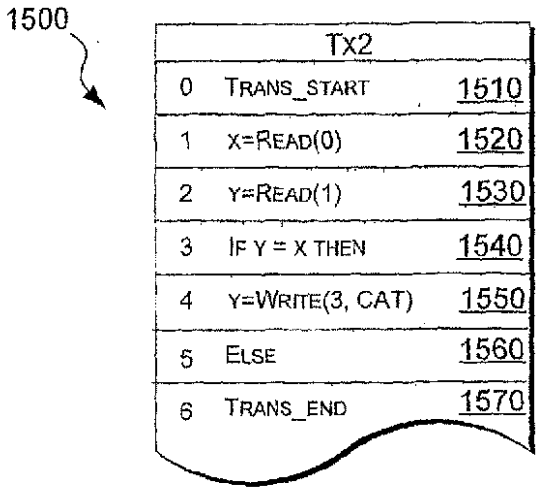
【図１３】



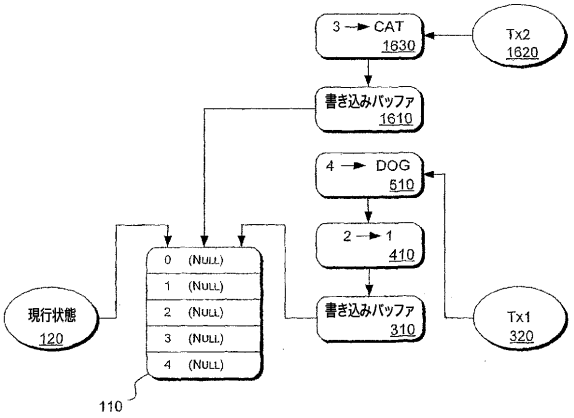
【図１４】



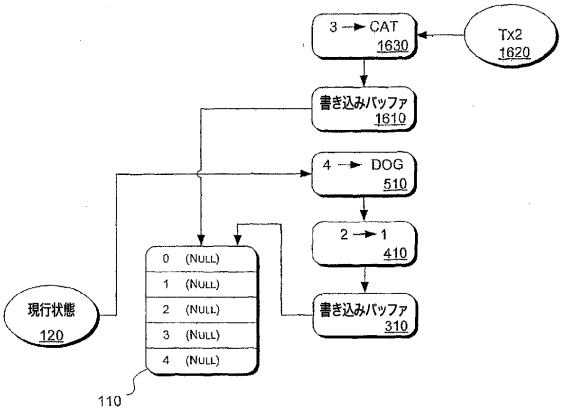
【図１５】



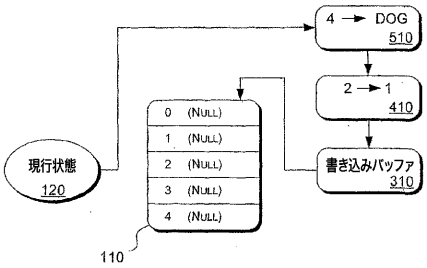
【図１６】



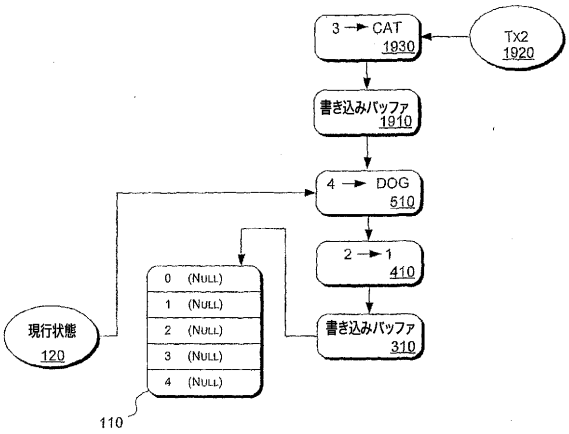
【図１７】



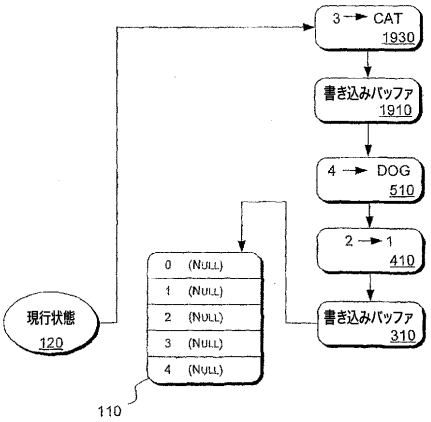
【図１８】



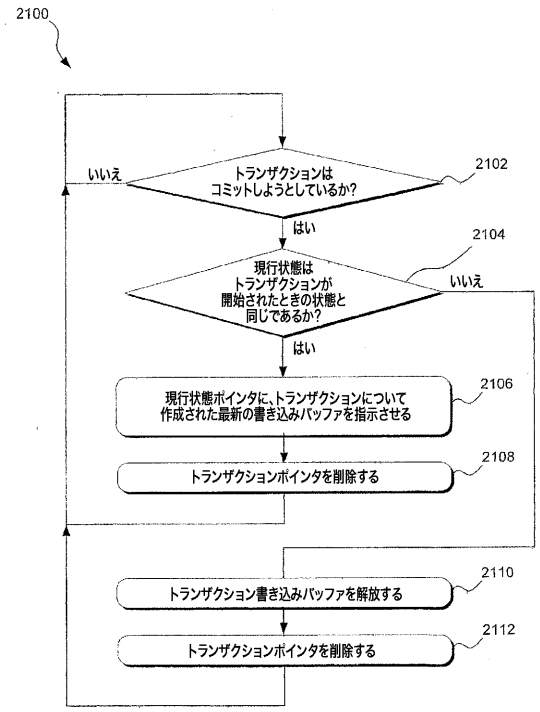
【図１９】



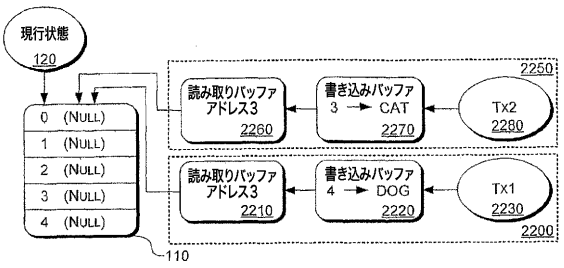
【図２０】



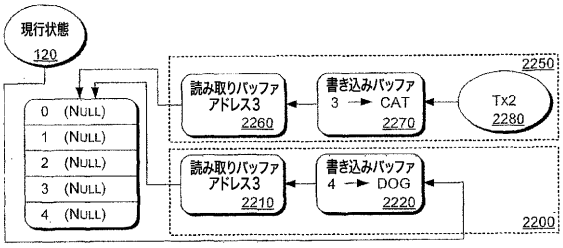
【図２１】



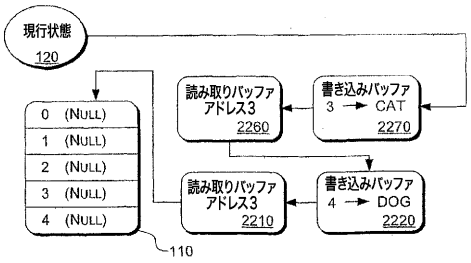
【図２２】



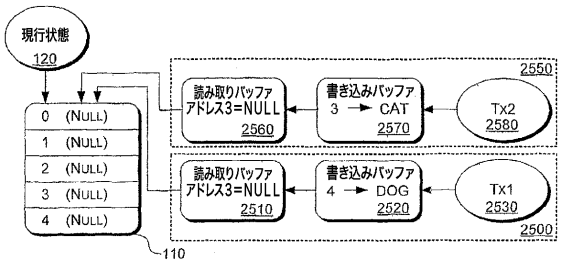
【図２３】



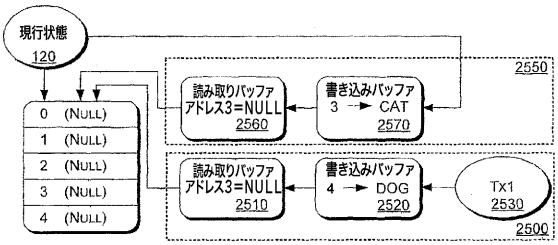
【図２４】



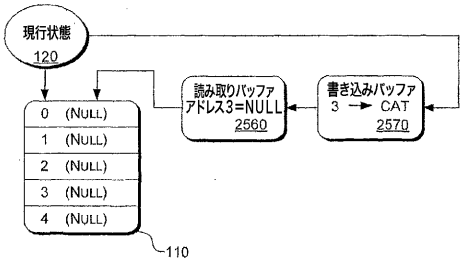
【図２５】



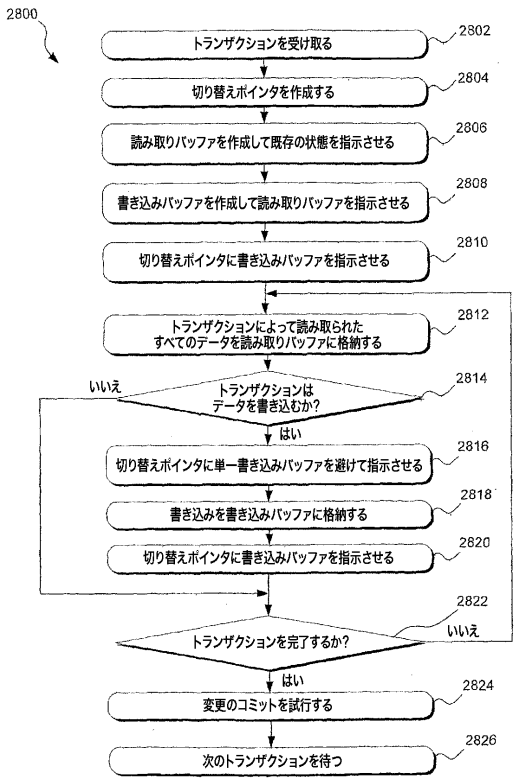
【図２６】



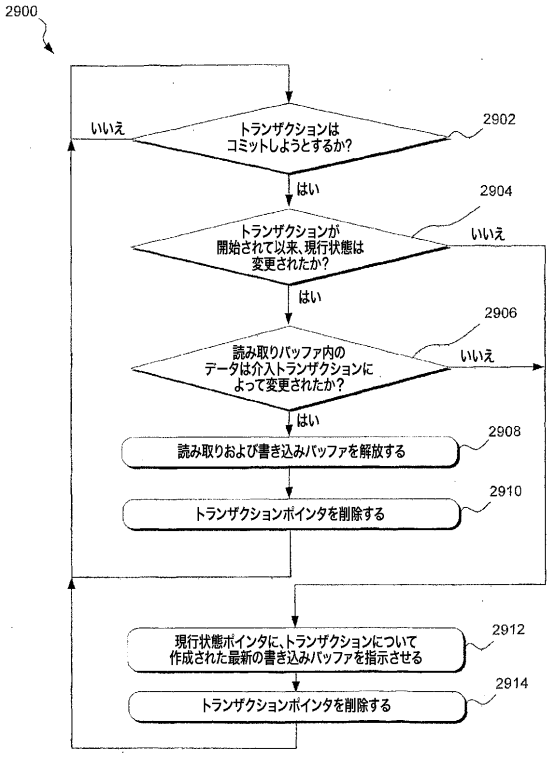
【図２７】



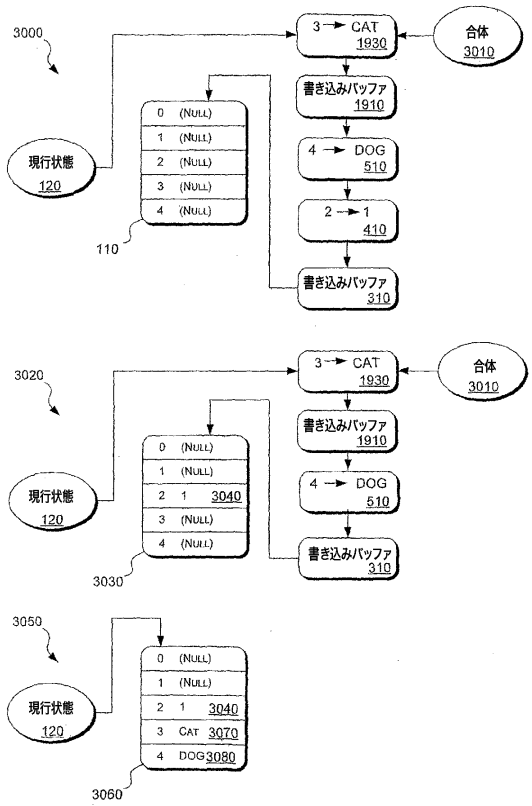
【図２８】



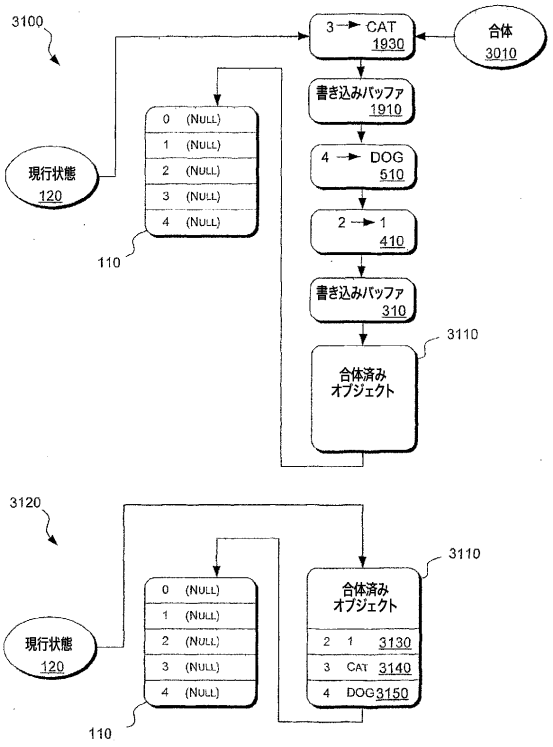
【図２９】



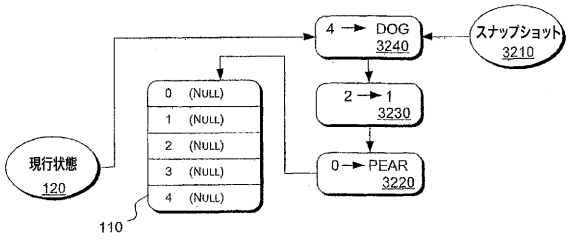
【図３０】



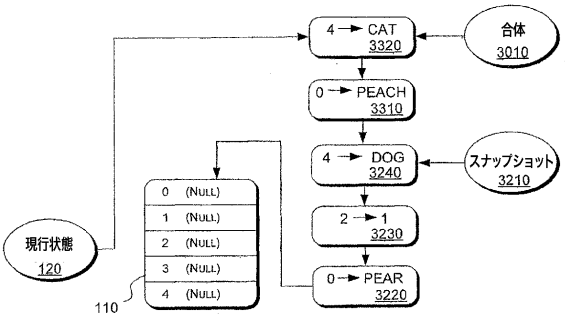
【図３１】



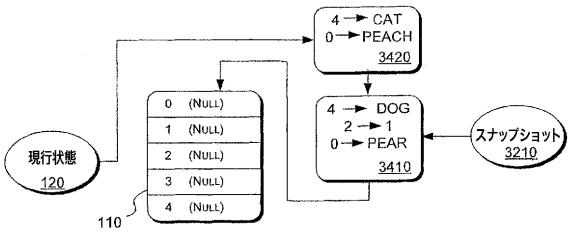
【図３２】



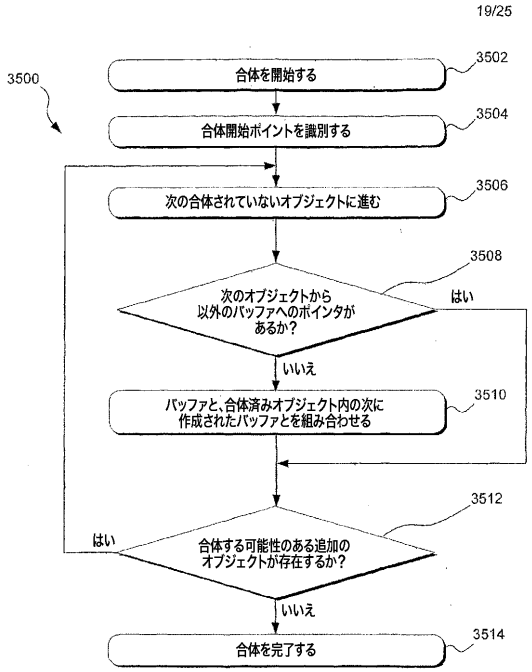
【図３３】



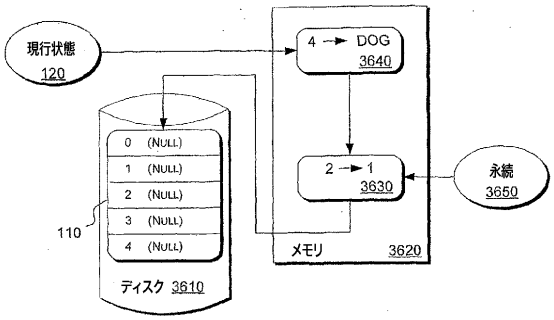
【図３４】



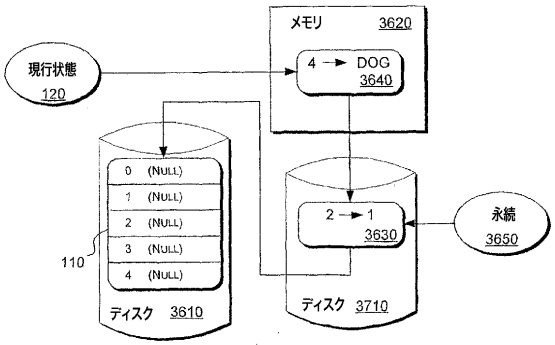
【図３５】



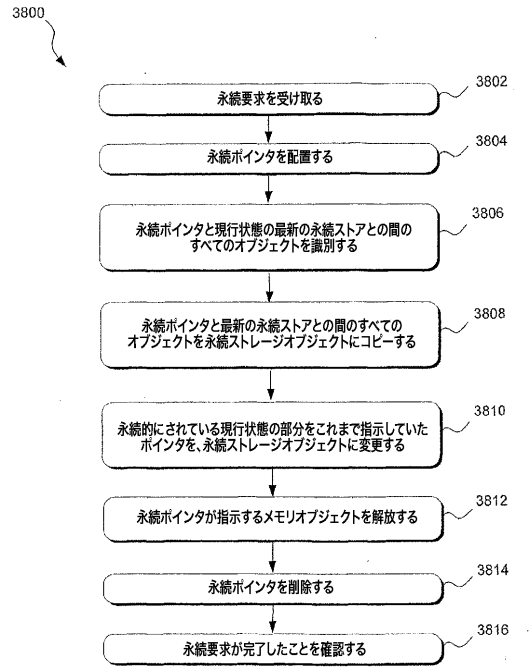
【図３６】



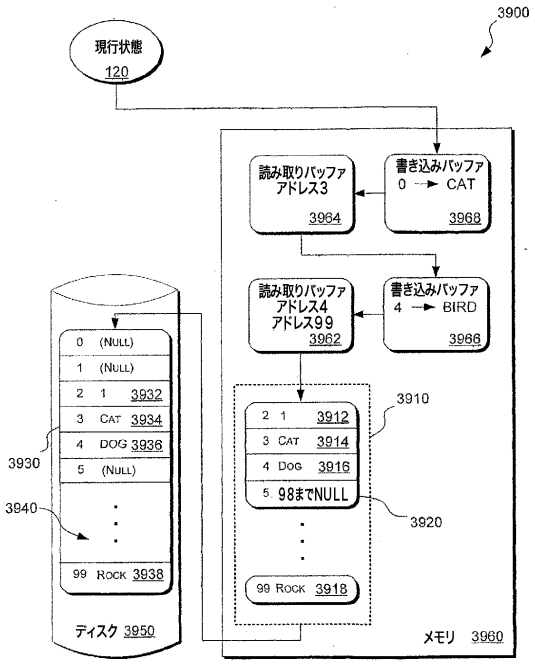
【図３７】



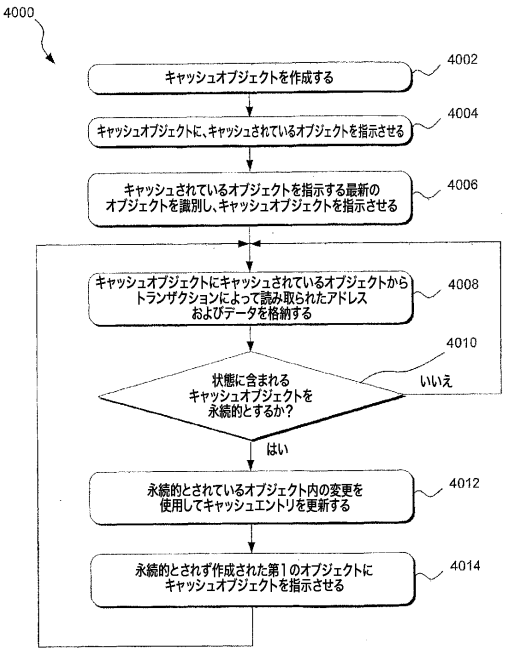
【図３８】



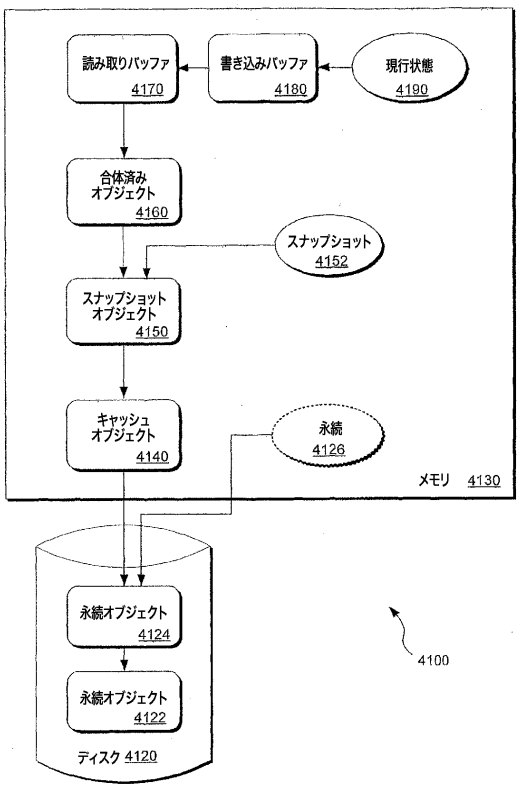
【図３９】



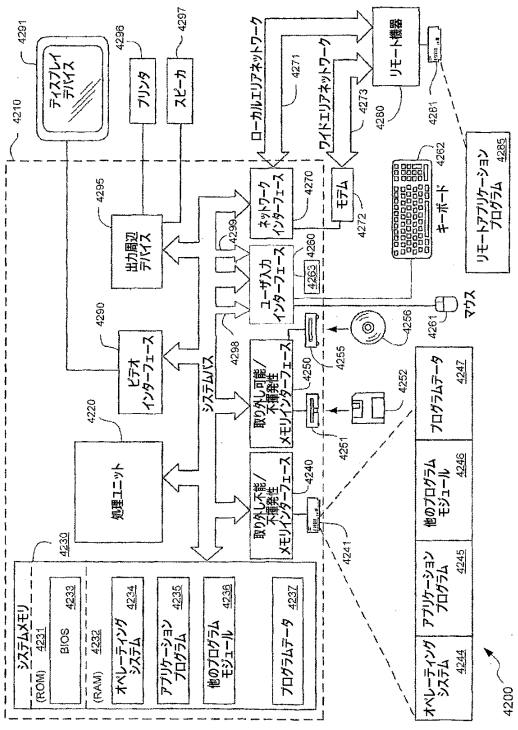
【図４０】


4000
キャッシュオブジェクトを作成する
4002
キャッシュオブジェクトに、キャッシュされているオブジェクトを指示させる
4004
キャッシュされているオブジェクトを指示する最新のオブジェクトを識別し、キャッシュオブジェクトを指示させる
4006
キャッシュオブジェクトにキャッシュされているオブジェクトからトランザクションによって読み取られたアドレスおよびデータを格納する
4008
状態に含まれるキャッシュオブジェクトを永続的とするか?
4010
いいえ
はい
永続的とされているオブジェクト内の変更を使用してキャッシュエントリを更新する
4012
永続的とされず作成された第1のオブジェクトにキャッシュオブジェクトを指示させる
4014

【図４１】


読み取りバッファ 4170
書き込みバッファ 4180
現行状態 4190
合体済みオブジェクト 4160
スナップショット 4152
スナップショットオブジェクト 4150
キャッシュオブジェクト 4140
永続 4126
メモリ 4130
永続オブジェクト 4124
永続オブジェクト 4122
ディスク 4120
4100

【図４２】


ディスプレイデバイス
プリンタ
スピーカ
ローカルエリアネットワーク
ワイドエリアネットワーク
リモート機器
ネットワークインターフェース
モデム
キーボード
リモートアプリケーションプログラム
出力周辺デバイス
ユーザ入力インターフェース
マウス
ビデオインターフェース
システムバス
取り外し可能/不揮発性メモリインターフェース
処理ユニット
システムメモリ
(ROM)
BIOS
(RAM)
オペレーティングシステム
アプリケーションプログラム
他のプログラムモジュール
プログラムデータ
4200

フロントページの続き

(72)発明者　ジョン　アール．ドゥサー
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　インターナショナル　パテンツ内

審査官　桜井　茂行

(56)参考文献　特開平０９－０６２５５４（ＪＰ，Ａ）
国際公開第９３／００３４３６（WO，Ａ１）
米国特許第６５９４７５１（ＵＳ，Ｂ１）
国際公開第９６／０９７３０（WO，Ａ１）
米国特許第６１９９１４１（ＵＳ，Ｂ１）
米国特許第０５９８３２２５（ＵＳ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F　12/00
G06F　17/30
G06F　17/40
JSTPlus(JDreamII)