(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7101566号
(P7101566)

(45)発行日　令和4年7月15日(2022.7.15)

(24)登録日　令和4年7月7日(2022.7.7)

(51)国際特許分類

Ｇ０６Ｆ　16/23　(2019.01)	Ｇ０６Ｆ	16/23	
Ｇ０６Ｆ　16/17　(2019.01)	Ｇ０６Ｆ	16/17	２００
Ｇ０６Ｆ　16/18　(2019.01)	Ｇ０６Ｆ	16/18	１００

ＦＩ

請求項の数　20　外国語出願　（全27頁）

(21)出願番号　特願2018-161332(P2018-161332)
(22)出願日　平成30年8月30日(2018.8.30)
(65)公開番号　特開2019-102059(P2019-102059A)
(43)公開日　令和1年6月24日(2019.6.24)
審査請求日　令和3年7月2日(2021.7.2)
(31)優先権主張番号　62/594,270
(32)優先日　平成29年12月4日(2017.12.4)
(33)優先権主張国・地域又は機関　米国(US)
(31)優先権主張番号　15/900,150
(32)優先日　平成30年2月20日(2018.2.20)
(33)優先権主張国・地域又は機関　米国(US)

(73)特許権者　300015447
エスアーペー エスエー
ドイツ連邦共和国，６９１９０ バルドルフ，ディートマル－ホップ－アレー１６
Ｄｉｅｔｍａｒ－Ｈｏｐｐ－Ａｌｌｅｅ １６，６９１９０ Ｗａｌｌｄｏｒｆ，Ｇｅｒｍａｎｙ
(74)代理人　100108453
弁理士　村山 靖彦
(74)代理人　100110364
弁理士　実広 信哉
(74)代理人　100133400
弁理士　阿部 達彦
(72)発明者　イズマイル・オウキッド

最終頁に続く

(54)【発明の名称】　不揮発性メモリにおけるマルチバージョン同時実行制御（ＭＶＣＣ）

(57)【特許請求の範囲】

【請求項１】

コンピュータ実装方法であって、

イベントが発生したと判定するステップであり、前記イベントより前に保留中であったマルチバージョンデータベースの1つまたは複数のレコードに対する1つまたは複数の書込みトランザクションがコミットしておらず、前記マルチバージョンデータベースが不揮発性メモリに記憶される、ステップと、

前記イベントより前に前記不揮発性メモリに記憶されたコミット値に基づいて前記1つまたは複数の書込みトランザクションを識別するステップであり、前記1つまたは複数の書込みトランザクションの各々がコミット値を含む、ステップと、

前記識別されたコミットされていない書込みトランザクションのうちの特定の1つを選択するステップと、

前記選択されたコミットされていない書込みトランザクションに対応するレコードの第1のバージョンを前記マルチバージョンデータベースから識別するステップであり、前記第1のバージョンがコミットされていない、ステップと、

前記イベントより前にコミットされた前記レコードの以前のバージョンを識別するステップと、

前記レコードの前記以前のバージョンが可視であり、前記レコードの前記第1のバージョンが可視でないことを示すように、前記レコードの可視性を設定するステップと

を含むコンピュータ実装方法。

【請求項２】
前記可視性を設定する前記ステップが、
前記レコードの前記以前のバージョンが可視であることを示すために、前記レコードの前記以前のバージョンに対応する削除タイムスタンプを設定するステップであり、トランザクションに対するレコードの前記可視性が前記削除タイムスタンプに基づく、ステップ
を含む、請求項1に記載の方法。
【請求項３】
前記可視性を設定する前記ステップが、
前記レコードの前記第1のバージョンに対応する削除タイムスタンプをガベージコレクションしきい値未満に設定するステップ
を含む、請求項1に記載の方法。
【請求項４】
前記ガベージコレクションしきい値が、最も古い実行中のトランザクションが開始された時間に対応する最小開始タイムスタンプに基づく、請求項3に記載の方法。
【請求項５】
前記イベントが、コンピュータシステムのクラッシュまたはリブートに対応する、請求項1に記載の方法。
【請求項６】
選択する前記ステップが、
前記識別されたトランザクションが複数のステートメントを含むことを決定するステップであり、前記識別されたトランザクションが前記複数のステートメントの各々のステートメント識別子を含む、ステップと、
前記複数のステートメントの第1のステートメントを識別するステップと
を含む、請求項1に記載の方法。
【請求項７】
前記ステートメント識別子がステートメントカウンタに基づき、前記選択されたコミットされていない書込みトランザクションに対応するトランザクション識別子が、前記ステートメントカウンタとは異なるトランザクションカウンタに対応する、請求項6に記載の方法。
【請求項８】
メモリと、
前記メモリに結合された少なくとも1つのプロセッサを含み、前記少なくとも1つのプロセッサが、
イベントが発生したと判定することであり、前記イベントより前に保留中であったマルチバージョンデータベースの1つまたは複数のレコードに対する1つまたは複数の書込みトランザクションがコミットしておらず、前記マルチバージョンデータベースが不揮発性メモリに記憶される、判定することと、
前記イベントより前に前記不揮発性メモリに記憶されたコミット値に基づいて前記1つまたは複数の書込みトランザクションを識別することであり、前記1つまたは複数の書込みトランザクションの各々がコミット値を含む、識別することと、
前記識別されたコミットされていない書込みトランザクションのうちの特定の1つを選択することと、
前記選択されたコミットされていない書込みトランザクションに対応するレコードの第1のバージョンを前記マルチバージョンデータベースから識別することであり、前記第1のバージョンがコミットされていない、識別することと、
前記イベントより前にコミットされた前記レコードの以前のバージョンを識別することと、
前記レコードの前記以前のバージョンが可視であり、前記レコードの前記第1のバージョンが可視でないことを示すように、前記レコードの可視性を設定することと
を行うように構成されている、システム。
【請求項９】

前記可視性を設定する前記プロセッサが、
前記レコードの前記以前のバージョンが可視であることを示すために、前記レコードの前記以前のバージョンに対応する削除タイムスタンプを設定することであり、トランザクションに対するレコードの前記可視性が前記削除タイムスタンプに基づく、設定すること
を行うように構成されている、請求項8に記載のシステム。
【請求項１０】
前記可視性を設定する前記プロセッサが、
前記レコードの前記第1のバージョンに対応する削除タイムスタンプをガベージコレクションしきい値未満に設定する
ように構成されている、請求項8に記載のシステム。
【請求項１１】
前記ガベージコレクションしきい値が、最も古い実行中のトランザクションが開始された時間に対応する最小開始タイムスタンプに基づく、請求項10に記載のシステム。
【請求項１２】
前記イベントが、コンピュータシステムのクラッシュまたはリブートに対応する、請求項8に記載のシステム。
【請求項１３】
選択する前記プロセッサが、
前記識別されたトランザクションが複数のステートメントを含むことを決定し、前記識別されたトランザクションが前記複数のステートメントの各々のステートメント識別子を含み、
前記複数のステートメントの第1のステートメントを識別する
ように構成されている、請求項8に記載のシステム。
【請求項１４】
前記ステートメント識別子がステートメントカウンタに基づき、前記選択されたコミットされていない書込みトランザクションに対応するトランザクション識別子が、前記ステートメントカウンタとは異なるトランザクションカウンタに対応する、請求項13に記載のシステム。
【請求項１５】
少なくとも1つのコンピューティングデバイスによって実行されると、前記少なくとも1つのコンピューティングデバイスに、
イベントが発生したと判定することであり、前記イベントより前に保留中であったマルチバージョンデータベースの1つまたは複数のレコードに対する1つまたは複数の書込みトランザクションがコミットしておらず、前記マルチバージョンデータベースが不揮発性メモリに記憶される、判定することと、
前記イベントより前に前記不揮発性メモリに記憶されたコミット値に基づいて前記1つまたは複数の書込みトランザクションを識別することであり、前記1つまたは複数の書込みトランザクションの各々がコミット値を含む、識別することと、
前記識別されたコミットされていない書込みトランザクションのうちの特定の1つを選択することと、
前記選択されたコミットされていない書込みトランザクションに対応するレコードの第1のバージョンを前記マルチバージョンデータベースから識別することであり、前記第1のバージョンがコミットされていない、識別することと、
前記イベントより前にコミットされた前記レコードの以前のバージョンを識別することと、
前記レコードの前記以前のバージョンが可視であり、前記レコードの前記第1のバージョンが可視でないことを示すように、前記レコードの可視性を設定することと
を含む動作を実行させる命令が記憶された非一時的コンピュータ可読デバイス。
【請求項１６】
前記可視性を設定することが、
前記レコードの前記以前のバージョンが可視であることを示すために、前記レコードの前

記以前のバージョンに対応する削除タイムスタンプを設定することであり、トランザクションに対するレコードの前記可視性が前記削除タイムスタンプに基づく、設定すること
を含む、請求項15に記載の非一時的コンピュータ可読デバイス。
【請求項１７】
前記可視性を設定するように構成された前記1つのコンピューティングデバイスが、
前記レコードの前記第1のバージョンに対応する削除タイムスタンプをガベージコレクションしきい値未満に設定すること
を含む動作を実行するように構成されている、請求項15に記載の非一時的コンピュータ可読デバイス。
【請求項１８】
前記ガベージコレクションしきい値が、最も古い実行中のトランザクションが開始された時間に対応する最小開始タイムスタンプに基づく、請求項17に記載の非一時的コンピュータ可読デバイス。
【請求項１９】
前記イベントが、コンピュータシステムのクラッシュまたはリブートに対応する、請求項15に記載の非一時的コンピュータ可読デバイス。
【請求項２０】
選択するように構成された前記1つのコンピューティングデバイスが、
前記識別されたトランザクションが複数のステートメントを含むことを決定することであり、前記識別されたトランザクションが前記複数のステートメントの各々のステートメント識別子を含む、決定することと、
前記複数のステートメントのうちの第1のステートメントを識別することであり、前記ステートメント識別子がステートメントカウンタに基づき、前記選択されたコミットされていない書込みトランザクションに対応するトランザクション識別子が、前記ステートメントカウンタとは異なるトランザクションカウンタに対応する、識別することと
を含む動作を実行するように構成されている、請求項15に記載の非一時的コンピュータ可読デバイス。
【発明の詳細な説明】
【技術分野】
【０００１】
関連出願の相互参照
本出願は、Oukidらによる、2017年12月14日に出願された「Multi-Versioning Concurrency Control (MVCC) In Non-Volatile Memory」の米国仮特許出願第62/594,270号の利益を主張するものであり、Oukidらによる、2017年6月13日に出願された「Big Block Allocation of Persistent Main Memory」の同時係属米国特許出願第15/621,640号、Oukidらによる、2017年6月13日に出願された「Defragmentation of Persistent Main Memory」の米国特許出願第15/621,736号、およびBoossらによる「Hybrid SCM_DRAM Transactional Storage Engine for Fast Recovery」の米国特許出願第2015/0355981号に関係し、それらのすべては、参照によりその全体が本明細書に組み込まれる。
【背景技術】
【０００２】
一般に、ストレージクラスメモリ(SCM)は、ダイナミックリードアクセスメモリ(DRAM)の低レイテンシおよびバイトアドレス指定能力を、従来の記憶媒体の不揮発性、面密度、および経済的特性と組み合わせる。さらに、SCM技術のバイトアドレス指定能力と低レイテンシが与えられると、中央処理装置(CPU)は、DRAMにデータをバッファリングすることなく、SCMに記憶されたデータにアクセスすることができる。したがって、SCM技術は、コンピュータメモリと従来の記憶媒体との区別を曖昧にする。しかしながら、SCMは、DRAMとともに使用されることがある。
【発明の概要】

【課題を解決するための手段】
【０００３】
添付の図面は、本明細書に組み込まれ、本明細書の一部を形成する。
【図面の簡単な説明】
【０００４】
【図１】いくつかの実施形態による、不揮発性メモリにおけるマルチバージョン同時実行制御(ＭＶＣＣ)の例を示すブロック図である。
【図２】例示的な一実施形態による、マルチバージョン同時実行制御(ＭＶＣＣ)システムに関連付けられた回復プロセスを実行するためのフローチャートである。
【図３】様々な実施形態を実施するのに有用な例示的なコンピュータシステムである。
【図４】別の例示的な実施形態による、マルチバージョン同時実行制御(ＭＶＣＣ)システムに関連付けられた回復プロセスを実行するためのフローチャートである。
【発明を実施するための形態】
【０００５】
図面において、同様の参照番号は、一般に、同一または類似の要素を示す。さらに、一般に、参照番号の一番左の桁は、参照番号が最初に現れる図面を識別する。
【０００６】
不揮発性メモリにおけるマルチバージョン同時実行制御(ＭＶＣＣ)のためのシステム、方法、および/またはコンピュータプログラム製品の実施形態、および/またはそれらの組合せおよび部分的な組合せが本明細書で提供される。
【０００７】
図1は、いくつかの実施形態による、不揮発性メモリにおけるマルチバージョン同時実行制御(ＭＶＣＣ)の例を示すブロック図100である。トランザクションストレージシステム(ＴＳＳ)102は、マルチバージョンデータベース(ＭＤＢ)104に対して実行されているトランザクション106を管理し得る。
【０００８】
トランザクションがデータベースに変更を加えると、その変更は、データが記憶または保持される(persist)ディスクに書き込まれる前に、先書きログ(write-ahead log)(以下、「ログ」)に最初に書き込まれ得る。一般に、ログは、トランザクションが失敗した場合にトランザクションを元に戻すのに十分な情報を含む。ログは、データがディスクに保持される前にクラッシュした場合のトランザクションの再生を可能にするやり直しまたはリプレイ情報も含み得る。ログ自体は、ディスクストレージに記憶され得る。
【０００９】
ログは、しばしば、データベースのデータの複数のコピーを含む。たとえば、トランザクションによって行が更新される場合、更新によって1つの属性のみが変更された場合でも、ログは、更新前のすべての行データのコピーと、更新後のすべての行データのコピーの両方を含み得る。したがって、ログは、記憶のために余分なメモリを消費する冗長な情報と、ディスクストレージに書き込むための余分なコンピューティングサイクルとを含み得る。
【００１０】
いくつかの実施形態では、あるデータ(たとえば、日付またはログ情報)をディスクストレージに書き込むのではなく、ＴＳＳ102は、不揮発性メモリ(ＮＶＭ)108にアクセスする。ＮＶＭ108は、プロセッサまたは他のコンピューティングデバイスによって直接アクセス可能なバイトアドレス指定可能なメモリを含み得る。一実施形態では、ＮＶＭ108は、ダイナミックリードアクセスメモリ(ＤＲＡＭ)の低レイテンシおよびバイトアドレス指定能力を、従来の記憶媒体の不揮発性、面密度、および経済的特性と組み合わせるストレージクラスメモリ(ＳＣＭ)を含み得る。いくつかの実施形態に関して本明細書で使用されるように、ストレージクラスメモリ(ＳＣＭ)、バイトアドレス指定可能不揮発性メモリ(ＮＶＭ)、およびＮＶＲＡＭ(ランダムアクセスメモリ)は互換的に使用される。
【００１１】

一般的なディスクストレージシステムでは、データは、DRAMからのプロセッサにのみアクセス可能であり得る。たとえば、ディスクに書き込まれるデータは、最初にDRAMに書き込まれる。DRAMからのデータは、次いで、ディスク上に書き込まれるか、さもなければ、ディスク上に保持され得る。同様に、ディスクストレージから読み取られているデータは、最初にDRAMに転送され、次いで、プロセッサによってアクセスされ得る。

【００１２】

対照的に、NVM108は、DRAMを仲介として使用しない。揮発性メモリ(VM)110おとNVM108の両方は、(一般的なディスクストレージと同様に)(ページ粒度で)ページ(ブロック)アドレス指定可能ではなく、(バイト粒度で)バイトアドレス指定可能であり、したがって、最初にDRAMとの間でデータを移動させることなく直接アクセス可能であり得る。しかしながら、電力が失われた後にデータを失う可能性があるVM110とは異なり、NVM108は、電力サイクル、クラッシュ、またはシステムのリブートにわたってその状態を維持し得る。

【００１３】

一実施形態では、NVM108は、ディスクストレージよりも、読取り/書込みアクセスのために、より高速であるか、またはより少ない計算サイクル、時間、または他のリソースを必要とするが、DRAMまたは他のVM110アクセスよりも遅い可能性がある。たとえば、DRAMアクセスは、ディスクアクセスよりも最高100万倍速い、または安価であり得るが、VM110にアクセスすることは、NVM108にアクセスするよりも5倍速い(または、5倍少ない計算サイクルまたは処理コストを消費する)可能性がある。

【００１４】

TSS102は、メモリと計算サイクルの両方を無駄にする可能性のある重複するトランザクション情報をログに書き込む代わりに、NVM108とVM110の両方に直接アクセスし、ログを維持するのに必要なリソースを無駄にすることなく情報を記憶するために、それらを互いに組み合わせて使用し得る。TSS102は、MDB104のデータ上で、またはそれに対して動作するトランザクション106を管理するために、一般的なディスクストレージよりもNVM108ストレージの効率性を利用し得る。

【００１５】

MDB104は、複数のバージョンのデータを記憶または維持するマルチバージョンデータベースであり得る。MDB104は、データおよびトランザクションのタイムスタンプに基づいて、異なるバージョンのデータが異なるトランザクション106によってアクセスおよび/または変更されることを可能にし得る。

【００１６】

一実施形態では、MDB104のデータは、上書きまたは削除されない可能性がある。たとえば、(たとえば、トランザクション106によって)新しいデータがMDB104に追加されると、新しい行116が挿入され得る。データがMDB104から削除されるとき、削除タイムスタンプ(DTS112)は、その行がもはや可視でなくなったことを示すように更新され得る(MDB104は、ガベージコレクション時まで元の「削除済み」データを保持し得る)。データがMDB104において更新されるべきとき、行の元のバージョンのDTS112は、元の行データがもはや可視でないことを示すために更新され、更新されたデータを有する新しい行が挿入され、削除された行および新しく追加された行が同じレコードまたはデータの異なるバージョンであることを示すために、指示(ポインタ122など)が提供され得る。

【００１７】

MDB104は、レコードまたは行116に関する情報を行メタデータ114として維持し得る。一実施形態では、行メタデータ114は、作成またはコミットタイムスタンプ(CTS120)、削除タイムスタンプ(DTS112)、および読取りタイムスタンプ(RTS)134などのタイムスタンプ情報を含み得る。

【００１８】

MDB104のマルチバージョン動作のために、トランザクション106によって実行される削除動作は、削除された行がガベージコレクションされるまで行の削除されたバージョンが

メモリ内にまたはMDB104の一部として残り得るように、本質的に論理的であり得る。元の(削除された)行データを維持することによって、クラッシュまたは他の障害の場合にトランザクション106を元に戻すまたはロールバックする必要がある場合に必要な情報にアクセスすることができ得る。上記のように、行がもはやトランザクション106に可視でなくなった時を示すようにDTS112を更新することによって、削除動作が実行され得る。次いで、たとえば、DTS112の後に開始する(たとえば、開始タイムスタンプ(STS)118を有する)任意のトランザクション106は、削除されたデータを見たりアクセスしたりすることができない可能性がある。

【００１９】

CTS120は、行(または行のバージョン)が作成されたとき、コミットされたとき、またはさもなければトランザクション106に可視になったときを示し得る。たとえば、行116のCTS120より前のSTS118を有するトランザクション106は、(トランザクションが開始されたときに行116が作成されなかったので)行データを見たりアクセスしたりすることができない。一実施形態では、レコードまたは行116は、その行がいつ可視であったかを示すCTS120とDTS112の両方を含み得る。TSS102は、CTS120、DTS112、およびSTS118を使用して、それらの実行または動作中にどのデータまたはレコード116がどのトランザクション106に可視であったかを判定し得る。

【００２０】

一実施形態では、DTS112を使用して、行の現在のバージョンがレコードまたは行の最新バージョンであるかどうかを示すことができる。たとえば、DTS112は、無限大の値、負の数、または(行の複数のバージョンが存在する場合)行、列、もしくはレコードの最新バージョンであることを示し得る他の指定された値に設定され得る。行がトランザクション106によって削除された場合、DTS112は、削除が生じたまたはコミットされたときに関するタイムスタンプを示し得る。本明細書で使用されるように、レコード、行、タプル、および列は、すべて互換的に使用され得る。

【００２１】

一実施形態では、MDB104は、行の1つの属性のみが更新されたとしても、完全な行のタプルまたは行の値を付加または追加することによって行の更新が生成され得る付加専用ストレージ(append only storage)として動作する。たとえば、トランザクション106は、行の新しいコピー(行の新しいバージョン)を作成し、またはMDB104のテーブルに付加し、行の新しいバージョンの属性を更新し、新しいバージョンを以前のバージョンのレコードまたはタプルにリンクまたはポイントし得る。行の新しいまたは最新バージョンは、行のバージョンチェーンの先頭または末尾のいずれかであり得る。一実施形態では、付加専用を使用して、TSSまたはMDB104は、メモリ(たとえば、NVM108)に属性を連続的に記憶することができ、これは、CPUキャッシュミスを最小限に抑え、性能を向上させ得る。

【００２２】

削除動作は、完全停止削除(full stop deletion)、または更新動作の一部のいずれかであり得る。行の完全停止削除の場合、DTS112は、行を削除するトランザクション106のSTS118またはCTS120Aに更新され、削除が完了し得る。一実施形態では、STS118は、トランザクション106がいつ開始するかを示し、CTS120Aは、トランザクションがコミットまたは持続される時間を示し得る。

【００２３】

更新動作の場合、データの新しいバージョンは、(削除された)レコードの前のバージョンのDTS112と同じCTS120を含み得る。一実施形態では、データの古いバージョンおよび/または新しいバージョンは、それらが関連していることを示すポインタ122を含み得る。

【００２４】

一実施形態では、TSS102は、行の異なるバージョンを追跡するために、CTS120およびDTS112ならびに/または逆インデックスを使用し得る。一実施形態では、逆インデックスは、タプルへの参照として行識別子を使用し得る。次いで、逆インデックスから、どの行が互いに異なるバージョンであるかが決定され得る。

【００２５】
MDB104は、単一のタプルまたはデータベースレコードの複数のバージョンを維持し得、各バージョンは、特定の時間間隔におけるタプルの状態を表し得る。図1に示す例では、MDB104は、行1の2つのバージョン、行2の単一のバージョン、および行3の3つのバージョンを含む。示された例ではデータの行が示されているが、他の実施形態では、行に加えて、または行の代わりに、データを記憶または表すために列または他のレコードが使用され得る。一実施形態では、本明細書で使用される用語「行」は、論理的な行またはタプルを指し得る。
【００２６】
MDB104がタプルまたはレコードの古いバージョンを維持している限り、データまたはデータベースの状態は、以前の時間期間中に決定され(および必要であればロールバックされ)得る。上述のように、MDB104は、(CTS120およびDTS112によって示されるように)様々な時間期間におけるデータの状態を示すメタデータを維持し得る。したがって、任意の時点において、TSS102は、STS118に基づいて、どのデータ(バージョン)が特定のトランザクション106に可視であったかを判定し得る。
【００２７】
したがって、MDB104は、データが更新されたかどうかにかかわらず、読取りクエリがデータの一貫したバージョンを読み取ることを可能にし得る。たとえば、トランザクション106が受信または開始されると、TSS102は、トランザクション106に関する情報またはメタデータを含むトランザクションオブジェクト124を作成(または以前に作成されたものを再利用)し得る。トランザクションオブジェクト124の情報は、特に、1つまたは複数の進行中または作動中の(inflight)トランザクションが完了しなかったクラッシュまたは障害の場合に、異なる時点におけるトランザクション106の状態を決定するために使用され得る。
【００２８】
トランザクションオブジェクト124は、読取りセット126を含み得る。読取りセット126は、データのどの行(バージョン)がトランザクション106によってアクセスまたは読み取られたかを示し得る。読取りセット126の各行またはレコードは、トランザクション106のSTS118以下のCTS120を含み得る。次いで、たとえば、トランザクション106の存続期間中、トランザクションは、トランザクションが依然として進行中である間に(後のトランザクション106によってその後削除されても)同じデータにアクセスし得る。
【００２９】
一実施形態では、TSS102は、グローバルカウンタ128を含み得る。グローバルカウンタ128は、本明細書で説明される行/レコードおよびトランザクションの様々なタイムスタンプにどの値が使用されるべきかを追跡または示し得る。一実施形態では、トランザクション106が受信または開始されると、そのSTS118は、グローバルカウンタ128の現在の値に等しくなるように設定され得る(またはグローバルカウンタ128の時間が増分され、STS118として設定され得る)。
【００３０】
一実施形態では、グローバルカウンタ128は、開始および/またはコミットされる書込みトランザクションごとに増分される論理時間であり得る。たとえば、第1の書込みトランザクションは、タイムスタンプ1で行われ、第2の後続の書込みトランザクションはタイムスタンプ2が割り当てられ得る。別の実施形態では、タイムスタンプは、整数または数値カウントではなく、クロック時間に関連し得る。一実施形態では、タイムスタンプは8バイトの長さであり、コミットまたは新しい書込みトランザクションごとに自動的に増分される、グローバルタイムスタンプカウンタ128(現在の時間)によって生成された符号なし整数を含み得る。
【００３１】
トランザクション106が開始すると、一意のTXID130、およびグローバルカウンタ128から現在の論理時間に等しいSTS118が割り当てられ得る。一実施形態では、特定のトラン

ザクションに割り当てられたタイムスタンプが、そのトランザクション106のトランザクション識別子(TXID)130として使用され得る。別の実施形態では、TXID130およびSTS118に異なる値が使用されてもよい。たとえば、TXID130は、1～2^63の範囲内の値を含み、タイムスタンプは、2^63～2^64の範囲内にあり得る。他の実施形態では、異なる値が使用されてもよい。

【００３２】

以下でより詳細に説明するように、CTS120またはDTS112が特定のトランザクション106のTXID130に設定されている場合、TXID103Aの値は、データがロックされている(および示されたトランザクションによって潜在的に変更されている)ことを、データにアクセスしようとする他のトランザクション106に対して示し得る。次いで、たとえば、要求中のトランザクションは、ロック中のトランザクションにデータの可視性ステータスを要求し得る。

【００３３】

行を読み取るとき、トランザクション106は、その行が別のトランザクションによってロックされておらず、トランザクションのSTS118がその行のCTS120とDTS112との間にある場合、行の最新バージョンを読み取ることができ得る。一実施形態では、TSS102は、ロックされたタプルに遭遇するときはいつでもトランザクションを中止し得る。これらの条件が満たされた場合、トランザクションは、既存のRTS134が新しいトランザクションのTXID130よりも低い場合、行の読取りタイムスタンプ(RTS)134をそのTXID130に設定し得る。RTS134は、最も古いトランザクション106が特定の行116にアクセスしていることを示し得る。これらの条件が満たされない場合、トランザクションは、行の古いバージョンを読み取り、行のRTS134は更新されない。別の実施形態では、TSS102は、RTS134を使用しない、または維持しない場合がある。

【００３４】

TSS102は、トランザクション106がロックを取得することなく行116を読み取ることを可能にする。TSS102は、読取りまたは開始時間118にどの行のどのバージョンがクエリまたはトランザクション106によって読み取られたかを示す読取りセット126を維持する。一実施形態では、コミット時(書込みセット132のデータがNVM108に書き込まれるかまたはNVM108に維持されようとしているとき)に、読取りセット126が再度読み取られ、元の読取りセットと交差され得る。いずれのデータも変更されていない場合、トランザクションは、コミットタイムスタンプ(CTS120A)を取得し、次いで、変更された行またはタプルのメタデータを更新する(作成されたタプルのCTS120および削除されたタプルのDTS112をCTS120Aに設定する)ことによって、開始される書込み段階に入る。このプロセスについて、以下でより詳細に説明する。そうでなく、データがSTS118とCTS120Aとの間で変更された場合、トランザクションは中止され得る。

【００３５】

作動中のトランザクション106は、何らかのデータにアクセスし、修正したが、まだ完了していない、進行中のトランザクションであり得る。上記で参照したように、ディスクシステムは、システムがクラッシュした場合、または他の何らかの障害がある場合に、完了していない可能性のある作動中のトランザクションを元に戻すために、先書きログを使用し得る。先書きログを維持するのではなく、TSS102は、MDB104によって記憶または維持された以前バージョンのデータを使用して、そうでなければログに書き込まれることになるアンドゥー情報を決定し得る(したがって、ログに書き込んだり維持したりするためにコンピューティングリソースを使用する必要がなくなる)。

【００３６】

一実施形態では、重複する情報をログに書き込むことを回避することによって、TSS102は、(そうでなければ冗長なログ情報によって消費されることになる)より少ないメモリリソースを使用し、ログ書込みトランザクションの実行を回避することによってスループットを向上させる。その結果、システムがより速くなり、ログ書込みシステムよりもコンピュータリソースの消費が少なくなり得る。一実施形態では、TSS102は、複製または他の

目的のために使用され得るトランザクションデータを別のシステムに送り出すまたは提供するためにログを作成または使用し得る。たとえば、複製は、ログを第1のシステムから第2のシステムに送り出す必要があり得る。

【００３７】

TSS102は、MDB104によって記憶された行の以前のバージョンと(NVM108に記憶され得る)トランザクションオブジェクト124の情報の両方から、「ログ情報」を取得し得る。上述のように、トランザクションオブジェクト124は、トランザクション106によって読み取られた識別子または行を含む読取りセット126を含み得る。トランザクションオブジェクト124は、トランザクション106によって挿入、更新(削除および挿入)、または削除された行を識別する書込みセット132も含み得る。一実施形態では、読取りセット126および書込みセット132は、(一般にログに書き込まれる行の元の情報のすべてを含まずに)どの行がアクセスまたは変更されたかを示す行識別子または参照のアレイを含み得る。

【００３８】

一般的なディスクベースのストレージシステムで実行され得るように、電源サイクルで失われ、ディスク上の別個のログを維持し得るDRAMまたは他の揮発性メモリに、トランザクションオブジェクト124を記憶するのではなく、TSS102は、トランザクションオブジェクト124の一部をVM110内に、トランザクションオブジェクト124の別の部分を永続的であり得るNVM108内に記憶し得る。一実施形態では、読取りセット126はVM110に記憶され、書込みセット132はNVM108に記憶され得る。いくつかの実施形態では、NVM108とVM110の両方に、一部分(たとえば、読取りセット126など)が記憶され得る。

【００３９】

書込みセット132は、トランザクション106によってどの行116が追加、更新、または削除されたかを含み得る。一実施形態では、書込みセット132に新しいエントリがある(たとえば、データがMDB104に挿入される、またはそこから削除される)たびに、エントリまたは書込みセット132がNVM108に保持され得る。次いで、たとえば、TSS102は、一般的なディスクベースのストレージシステムのログに記憶された情報を考慮するために、NVM108からの保持された書込みセット132とMDB104によって記憶されたバージョン情報との組合せを使用し得る。クラッシュ、電力損失、または他のシステム障害の場合、(NVM108内の)トランザクションオブジェクト124の保持された情報を使用して、TSS102は、完了していない作動中のトランザクション106を識別し、ロールバックし得る。たとえば、書込みセット132および他の保持されたトランザクションオブジェクト124の情報は、変更された作動中のトランザクション106の行を識別するために使用され、これらの行の以前のバージョンは、MDB104の記憶された情報から決定され得る。

【００４０】

一実施形態では、トランザクションオブジェクト124は、トランザクションがコミットされたかどうかを示すコミットフラグ136を含み得る。コミットフラグ136は、NVM108に記憶され得る。次いで、たとえば、システムクラッシュおよびリブートまたは再起動時に、TSS102は、コミットフラグ136がロールバックされるべき作動中のトランザクションを識別するように設定されていない任意のトランザクション106を識別し得る。一実施形態では、書込みセット132は、トランザクションによって作成された行を参照する作成サブセット、およびトランザクションによって削除された行を参照する削除サブセットなど、2つのサブセットを含み得る。

【００４１】

一実施形態では、VM110に記憶されたトランザクションオブジェクト124の一時的な部分は、未完了のトランザクション106をロールバックするのに必要ではない情報を含み得る。VM110に記憶され得る例示的なトランザクションオブジェクト124の情報は、トランザクションが無効であるか、アクティブであるか、コミットされたか、中止されたかを示し得るステータス変数、TXID130、STS118、スキャンセット、作成セットおよび削除セットにおいて参照される行を除く、その存続期間中にトランザクションによって読み取られた可視の行を参照する読取りセット126、同じトランザクションオブジェクトを使用(

トランザクションオブジェクト124の場合には再利用)した以前のトランザクションによって登録されたガベージエントリのリスト、および別のトランザクションとの競合が検出されたのでトランザクションを中止する必要があることを示す中止フラグを含み得る。一実施形態では、スキャンセットは、データを検索するために使用されたプレディケート(predicate)を含み得る。たとえば、スキャンセットは、すべての検索をやり直し、古いものと交差される新しい読取りセットを構築するために、コミット時に使用され得る。

【００４２】

他の実施形態では、この情報の一部または全部がNVM108に保持されてもよく、またはVM110とNVM108の両方に記憶されてもよい。たとえば、CTS120Aは、NVM108内のトランザクションオブジェクト124の永続的な部分と、VM110内のトランザクションオブジェクト124の一時的な部分の両方で維持され得る。

【００４３】

一実施形態では、耐久性および/または原子性(atomicity)(動作またはトランザクションが完全に動作する、またはまったく動作しない)を提供するために必要とされ得る、またはそうでなければ使用され得るどんなトランザクションオブジェクト124の情報でもNVM108に記憶され得る。たとえば、トランザクション106が部分的に実行され、MDB104または別の関連するシステムまたはコンピューティングデバイスがクラッシュした後、ロールバックされる、または変わる、または元に戻される場合、NVM108からの情報が使用され得る。そのような耐久性または原子性のいずれかを提供するために情報が必要とされ、または有用である場合、それはNVM108に記憶され、他の情報がVM110に記憶されてもよい。

【００４４】

【数１】

```
1: function IsVisible(mvcc&, atTime, txid)
2:     if mvcc.CTS == 0 or mvcc.DTS == 0 then                ▷CTSおよび/またはDTSが初期化されていない
3:         return false;
4:     do
5:         cts = mvcc.CTS; dts = mvcc.DTS;                   ▷CTSおよびDTSの現在の状態を読み取る
6:         if dts >= kMinTS then                             ▷DTSはタイムスタンプ
7:             dtsVisible = atTime < dts;
8:         else                                              ▷DTSはトランザクションID
9:             rslt = GetTxInfo(dts).IsTsVisible( mvcc.DTS, atTime, txid);
10:            if rslt == -2 then
11:                continue;                                 ▷ 新しいトランザクション、やり直しチェック
12:            if rslt == -1 then
13:                GetTxInfo(txid).willAbort = true;
14:                return false;
15:            dtsVisible = !rslt;                           ▷DTSが可視であるとき、txInfo. Is. Visibleは0を戻す
16:        if cts >= kMinTS then                             ▷CTSはタイムスタンプ
17:            ctsVisible = atTime >= cts;
18:        else                                              ▷CTSはトランザクションID
19:            rslt = GetTxInfo(cts).IsTsVisible(mvcc.CTS, atTime, txid);
20:            if rslt == -2 then
21:                continue;                                 ▷新しいトランザクション、やり直しチェック
22:            if rslt == -1 then
23:                GetTxInfo(txid).willAbort = true;
24:                return false;
25:            ctsVisible = rslt;                            ▷CTSが可視であるとき、txInfo. Is. Visibleは1を戻す
26:        return ctsVisible and dtsVisible;
27:    while true
```

アルゴリズム1

【００４５】

アルゴリズム1は、一実施形態によれば、特定の行116が特定のトランザクション106に可視であるかどうかをTSS102がどのように決定し得るかの例示的な動作を提供する。TSS102は、トランザクション106のSTS118として設定され得るグローバルカウンタ128から現在の時間を読み取り得る。STS118に基づいて、TSS102は、トランザクション106にどの行が可視であるかを判定し得る。行CTS120がSTS118よりも大きい、もしくはSTS118の後である場合、または行DTS112がSTS118より前である場合、その行はトランザクション106には可視でないことになる。しかしながら、STS118がCTS120およびDTS112内である場合、その行はトランザクション106に可視であり得る。

【００４６】

一実施形態では、CTS120およびDTS112の値は、行またはデータにアクセスしている可能性のある特定のトランザクション106のTXID130に対応するタイムスタンプ値またはトランザクション識別子(TXID)130Aのいずれかであり得る。CTS120またはDTS112がTXID130A値である場合、これは、その行が識別されたトランザクションによってロックされていることを示し得る。

【００４７】

一実施形態では、ある行が別のトランザクションによってロックされているとき、TSS102は、データの状態を判定し、その行が可視であるかどうかを判定するために、トランザクションに対する問合せを行い得る。たとえば、トランザクションがまだコミットされておらず、これが新しく挿入された行である場合、その行は読み取られない、またはアクセスされない可能性がある。あるいは、たとえば、トランザクションがコミットした場合、その行が読み取られ得る。

【００４８】

【数２】

```
1: function TxInfo::IsTsVisible(ts, atTime, callerTxid)
2:     locked_scope(mutex);
3:     if ts != txid or status == kInvalid then          ▷ID 'ts' を有するTXが終了した場合
4:         return -2;                                    ▷最初からやり直しチェックを行なわなければならない
5:     if txid == callerTxid then                        ▷呼び出し側トランザクションによって行がロックされる
6:         return 1;                                     ▷トランザクションによって作成(削除)された行がそれ自体に可視(不可視)である
7:     if status != kCommitted then                      ▷TXがアクティブであるか中止されている場合
8:         return -1;                                    ▷競合。トランザクションは中止されるものとする
9:     return atTime >= commitTS;                        ▷式が真である場合、1を戻し、そうでない場合、0を戻す
```

アルゴリズム2

【００４９】

アルゴリズム2は、アルゴリズム1を実行する際に使用され得る例示的な関数であり、一実施形態に従って、TSS102が、別のトランザクションによってロックされた行の行可視性をどのようにチェックするかを示す。TSS102は、行の状態を判定するために、ロック中のトランザクションに対する問合せを行い得る。

【００５０】

一実施形態では、TSS102は、ガベージコレクションの目的で有益であり得る最小のSTS118Aを維持し得る。たとえば、TSS102は、最も古い実行中のトランザクションのSTS118(min STS118A)を識別し、このトランザクションのSTS118(min STS118A)が特定のまたは識別された行のDTS112よりも大きい場合、行は、ガベージコレクションされ得る。たとえば、min STS118Aが5である場合、4以下のDTS112を有する任意の行がガベージコレクションされ得る。次いで、ガベージコレクションされた行のメモリが再利用される、または新しいデータもしくは情報のために利用可能にされ得る。

【００５１】

一実施形態では、TSS102は、MDB104が列記憶データベースであるか行記憶データベースであるかにかかわらず動作し得る。同様の方法で、テーブルの値の論理表現が動作され得る。たとえば、特定の行または列の値がメモリに連続的に記憶され得る。一実施形態では、TSS102が特定の数またはインスタンスのプレディケートについてレコードIDを取り出すとき、TSS102は、データがどのように構成され、メモリに記憶されるかを抽象化する。

【００５２】

【数３】

```
1: function INSERT(tableName, rowValues, txid)
2:     txInfo = GetTxInfo(txid);
3:     (tabID, pTab) = GetTablePointer(tableName);
4:     rowID = pTab->Insert(rowValues);          ▷行を挿入し、転置インデックスを更新する
5:     GetPersTxInfo(txid).createSet.PersistentAppend({tabID, rowID});   ▷作成セットを更新する
6:     pTab->MVCC[rowID].CTS = txid;             ▷そのCTSをtxidに設定することによって、行をロックする
7:     pTab->MVCC[rowID].DelTS = kInfinityTS;
8:     return true;
```

アルゴリズム3

【００５３】

アルゴリズム3は、一実施形態による、TSS102がMDB104に新しい行を挿入または付加し得る方法の一例である。たとえば、挿入したい値を新しい行に付加し得る。クラッシュが発生した場合、行はまだ可視でない(CTS120またはDTS112は依然として0に設定されている可能性がある)ので、4行目まで、TSS102はデータを保持しない可能性がある。一実施形態では、MDB104に追加された新しい行は、CTS120およびDTS112を含み得、これらは両方とも最初はゼロに設定されている。変更は、CTS120および/またはDTS112が更新されるまで(6～7行目に示すように)有効(可視)ではない可能性がある。6行目では、CTS120をTXID130Aに設定することができるので、その行はロックされる。7行目では、DTS112は、行が可視であることを意味する無限大に設定され得る。

【００５４】

しかしながら、CTS120またはDTS112が更新される前に、TSS102は、5行目に示されるようにNVM108内の書込みセット132の作成セットを最初に更新し得る。一実施形態では、テーブル識別子および行識別子、または他の値が作成セットに追加され得る。他の実施形態は、どの行またはレコードが付加、作成、または追加されているかを示すための他の参照を含み得る。書込みセット132は、追加されるレコードのコピーではなく、単にそれへの参照を含み得る。書込みセット132の作成セットの更新は、CTS120またはDTS112のいずれかが更新される前に実行される。

【００５５】

トランザクションをコミットする前に、(特定の行またはレコード116のCTS120またはDTS112を更新する前に)書込みセット132の新しく付加されたレコードがNVM108に保持され得る。トランザクション106がコミットされると、CTS120の値は、コミット時にTXID130Aからグローバルカウンタ128のコミットタイムスタンプに更新され得、コミットフラグ136が設定される。一実施形態では、コミットフラグ136が設定されると、グローバルカウンタ128は増分され、トランザクション106のコミットタイムスタンプ120Aとして使用され、これは、トランザクション106によって更新される行のCTS120またはDTS112として使用され得る。

【００５６】

【数４】

```
1:  function DELETE(tableName, predicates, txid)
2:      txInfo = GetTxInfo(txid);
3:      (tabID, pTab) = GetTablePointer(tableName);
4:      rowIDs = GetVisibleRows(pTab, predicates, txInfo.STS, txid);
5:      if txInfo.willAbort then
6:          abort(txid);
7:          return false;
8:      txInfo.scanSet.Append({tabID, predicates});                    ▷スキャンセットを更新する
9:      for each rowID in rowIDs do
10:         GetPersTxInfo(txid).deleteSet.PersistentAppend({tabID, rowID});   ▷削除セットを更新する
11:         if !TryToDelete(rowID, txid) then   ▷そのDTSをtxidに自動的に設定することによって、行のロックを試みる
12:             abort(txid);                                           ▷行の削除に失敗した場合、中止する
13:             return false;
14:     return true;
```

アルゴリズム4

【００５７】

アルゴリズム4は、一実施形態による、トランザクションが行を削除するときにTSS102によって実行され得る削除機能の一例である。上記のアルゴリズムでは、TSSは、ルックアップを実行して、どの行が可視であるかを判定し、7行目までのプレディケートを満たし得る。9～13行目では、TSS102は、最初に、NVM108に保持されている、任意の追加の変更を行う前の書込みセット132の削除セットにその行へのエントリまたは参照を付加し得る。次いで、たとえば、書込みセット132が更新され、保持された後、TSS102は、TXID130AをDTS112に書き込み得、レコード上のロックとして信号を送り得る。任意のその後のトランザクションは、次いで、DTS112がタイムスタンプの代わりにTXID130Aであるので、行がロックされていることを知ることになる。

【００５８】

【数５】

```
1:  function UPDATE(tableName, predicates, colFilter, newValues, txid)
2:      txInfo = GetTxInfo(txid);
3:      (tabID, pTab) = GetTablePointer(tableName);
4:      rowIDs = GetVisibleRows(pTab, predicates, txInfo.STS, txid);
5:      if txInfo.willAbort then
6:          abort(txid);
7:          return false;
8:      txInfo.scanSet.Append({tabID, predicates});                    ▷スキャンセットを更新する
9:      for each rowID in rowIDs do
10:         GetPersTxInfo(txid).deleteSet.PersistentAppend({tabID, rowID});   ▷削除セットを更新する
11:         if !TryToDelete(rowID, txid) then   ▷そのDTSをtxidに自動的に設定することによって、行のロックを試みる
12:             abort(txid);                                           ▷行の削除に失敗した場合、中止する
13:             return false;
14:     rows = pTab->GetValIDs(rowIDs);                    ▷削除された行のValueIDを取得する
15:     UpdateRows(rows, colFilter, newValues);                        ▷行に更新を適用する
16:     for each row in rows do                                        ▷更新された行を挿入し直す
17:         rowID = pTab->Insert(row);
18:         GetPersTxInfo(txid).createSet.PersistentAppend({tabID, rowID});
19:         pTab->MVCC[rowID].CTS = txid;                  ▷そのCTSをtxidに設定することによって、行をロックする
20:         pTab->MVCC[rowID].DTS = kInfinityTS;
21:     return true;
```

アルゴリズム5

【００５９】
アルゴリズム5は、一実施形態による、TSS102によって実行され得る更新動作の一例である。上述のように、更新動作は、行の削除動作に続いて行の挿入動作を伴い得る(削除された行のDTS112と挿入された行のCTS120とが同じである)。9～13行目は、(アルゴリズム4に関して上述したように)削除動作を実行する一例を示す。ひとたび削除が完了すると、行の新しいバージョンが作成され得る(16～20行目に示されているように、および、アルゴリズム3に関して上記で説明したように)。別の実施形態では、挿入動作は、更新動作における削除動作より前に実行されてもよい。
【００６０】
【数６】

```
1:  function Commit(txid)
2:      txInfo = GetTxInfo(txid);
3:      pTxInfo = GetPersTxInfo(txid);
4:      if pTxInfo.createSet.Empty() and pTxInfo.deleteSet.Empty() then        ▷読み取り専用TX
5:          txInfo.SetInvalid();                                               ▷TXを終了する(startTSがcommitTSとして使用される)
6:          return true;
7:      txInfo.commitTS = AtomicIncrement(m_currentTime);                      ▷commitTSを取得する
8:      for each scan in txInfo.scanSet do                                     ▷commitTS時にすべてのスキャンをやり直す
9:          rowIDs = GetVisibleRows(m_tables[scan.tabID], scan.predicates, txInfo.commitTS, txid);
10:         for each rowID in rowIDs do
11:             newReadSet.Append({scan.tabID, rowID});
12:     readSetDiff = Difference(txInfo.readSet, newReadSet)
13:     for each row in readSetDiff do                                         ▷TXによって変更された行のみをreadSetDiffに入れることができる
14:         mvcc = m_tables[row.tabID]->MVCC[row.rowID];
15:         if mvcc.CTS != txid and mvcc.DTS != txid then
16:             abort(txid);
17:             return false;
18:     pTxInfo.commitTS = info.commitTS;                                      ▷コミットタイムスタンプをpTxinfoに保持する
19:     MemBarrier();                                                          ▷commitTSおよびIsCommitedへの書込みの順序を確実にする
20:     pTxInfo.IsCommitted = TRUE;                                            ▷このTXがコミットされた旨の情報を保持する
21:     Persist(&(pTxInfo.IsCommitted));                                       ▷commitTsおよびIsCommittedが同じキャッシュラインを共有する
22:     txInfo.SetCommitted();                                                 ▷TXステータスをコミット済みに設定する
23:     for each row in pTxInfo.deleteSet do                                   ▷削除された行のDTSを更新する
24:         m_tables[row.tabID]->MVCC[row.rowID].DTS = txInfo.commitTS;
25:     for each row in pTxInfo.createSet do                                   ▷作成された行のCTSを更新する
26:         m_tables[row.tabID]->MVCC[row.rowID].CTS = txInfo.commitTS;
27:     MemBarrier();                                                          ▷mvccエントリに対するすべての更新が確実に実行を終了するようにする
28:     for each row in pTxInfo.createSet ∪ pTxInfo.deleteSet do               ▷作成された行および削除された行のmvccをフラッシュする
29:         Flush(&m_tables[row.tabID]->MVCC[row.rowID]);                      ▷非同期のフラッシュ
30:     MemBarrier();                                                          ▷すべてのフラッシュが確実に実行を終了するようにする
31:     txInfo.addGarbage(info.commitTS, pTxInfo.deleteSet);                   ▷削除された行をガーベッジとして登録する
32:     pTxInfo.createSet.Clear();                                             ▷作成セットをクリアする
33:     pTxInfo.deleteSet.Clear();                                             ▷削除セットをクリアする
34:     txInfo.SetInvalid();                                                   ▷TXを終了する
35:     return true;
```

アルゴリズム6

【００６１】
アルゴリズム6は、一実施形態による、TSS102によって実行され得るトランザクションコミットの一例である。コミット中、TSS102は、コミットされるべきトランザクション106と別のトランザクションとの間に競合が生じたかどうかを確認し得る。たとえば、8～11行目では、読取りセット126からの行が再度読み取られ、以前読み取られた値と交差され、または比較されて、データが仲介において(別のトランザクションによって)更新され

たかどうかを判定し得る。データが更新されていない(別のトランザクションとの競合がない)場合、TSS102は、CTS120Aの新しい時間値を戻し得るグローバル時間を増分することによって、グローバルカウンタ128からCTS120A(7行目)を取得し得る。

【００６２】

競合が検出された場合、トランザクション106は中止され得る。一実施形態では、トランザクションオブジェクト124の中止フラグが設定され得る。トランザクション106をコミットするか、中止するかを決定する際に、TSS102は、STS118で開始されたトランザクションが新しく取得されたCTS120Aと、可視である同じデータであることを確認し得る。TSS102は、トランザクションによって使用される入力データがこのトランザクションまたは別のトランザクションによって変更されていないことを保証する。読取りセット126の読取り行のいずれかが変更された場合、トランザクションは中止される。一実施形態では、異なる読取りセットに対する唯一の例外は、トランザクション自体がデータを挿入、削除、または変更した13～17行目に示されている。

【００６３】

競合が検出されない場合、トランザクションオブジェクト124の状態はコミットされた状態に更新され得る。18～21行目では、CTS120がCTS120Aに更新され、書込みがこの順序で行われることを確実にするために、メモリバリアが発行され、コミットフラグ136が設定され、コミットフラグ136とCTS120Aの両方がNVM108に保持され得る。コミットフラグ136およびCTS120Aは、障害時にまだ完了していない作動中のトランザクションによって行われた変更をTSS102が元に戻すことを可能にし得る。

【００６４】

データがひとたびNVM108に保持されると、クラッシュがあっても、トランザクションを再度実行する代わりに、TSSは、コミットプロトコルを完了または終了し得る。コミットプロトコルを終了するために、TSS102は、TXID130Aの値を有する削除セット内の行のDTS112を読み取り、DTS112の値をCTS120Aに設定し得る。TSS102は、各行のCTS120がTXID130AからCTS120Aに更新されることを除いて、書込みセット132の作成セット内の行と同様の動作を実行し得る。ある行のタイムスタンプがTXID130AからCTS120Aに更新されると、もはや行の可視性についてトランザクションに対する問合せを行う必要はない。

【００６５】

28～29行目において、データは、NVM108の揮発性CPUキャッシュ部分から不揮発性部分に明示的にフラッシュされる。一実施形態では、フラッシュされるデータは、書込みセット132内の挿入/削除された行のCTS120および/またはDTS112を含み得、付加されたレコードは、CTS120Aを取得する(およびCTS120またはDTS112を更新するためにそれを使用する)前にNVM108に保持され得る。変更は、トランザクションを終了する前に保持される。削除される行は、ガベージコレクションされる候補である。一実施形態では、TSS102は、これらの行のうちのどれをすぐにガベージコレクションできるかを判定し、別のトランザクションまたはデータによる使用のためにそのスペースを解放し得る。削除された行がガベージコレクションのために利用可能でない場合、行のバージョンはMDB104の一部のままである。トランザクションは、トランザクションオブジェクト124内のステータスを無効または完了に設定することによって終了し得る。

【００６６】

【数７】

```
1: function ABORT(txid)
2:     txInfo = GetTxInfo(txid);
3:     pTxInfo = GetPersTxInfo(txid);
4:     txInfo.SetAborted();                                          ▷ステータスを中止に設定する
5:     for each row in pTxInfo.deleteSet do                          ▷削除された行を再アクティブ化する
6:         if m_tables[row.tabID]->MVCC[row.rowID].DTS == txid then  ▷このTXによってDTSがロックされている場合
7:             m_tables[row.tabID]->MVCC[row.rowID].DTS = kInfinityTS;
8:     for each row in pTxInfo.createSet do                          ▷作成された行を無効として設定する
9:         m_tables[row.tabID]->MVCC[row.rowID].CTS = 0;             ▷行を無効としてマークする
10:        m_tables[row.tabID]->MVCC[row.rowID].DTS = CurMinSTS;     ▷DTSをCurMinSTSに設定する
11:    MemBarrier();                ▷mvccエントリに対するすべての更新が確実に実行を終了するようにする
12:    for each row in pTxInfo.createSet ∪ pTxInfo.deleteSet do      ▷変更された行のmvccをフラッシュする
13:        Flush(&m_tables[row.tabID]->MVCC[row.rowID]);             ▷非同期のフラッシュ
14:    MemBarrier();                ▷すべてのフラッシュが確実に実行を終了するようにする
15:    CollectGarbage({kMinTS, pTxInfo.createSet});                  ▷ガーベッジ(作成された行)を収集する
16:    pTxInfo.deleteSet.Clear();                                    ▷削除セットをクリアする
17:    pTxInfo.createSet.Clear();                                    ▷作成セットをクリアする
18:    txInfo.SetInvalid();                                          ▷TXを終了する
```

アルゴリズム7

【００６７】

アルゴリズム7は、一実施形態による、TSS102によって実行され得る例示的な中止動作である。上述のように、(たとえば、トランザクションの実行中にデータの読取りセット126が変更されたなど)競合が発生した場合、トランザクションが中止され得る。TSS102は、削除セットを識別し、削除のためにロックされた行をロック解除し得る。たとえば、5～7行目において、DTS112がTXID130Aに設定された行は、無限大に再設定されてもよく、したがって、行が再度可視であることを示す。

【００６８】

コミットされていない中止されたトランザクションによって追加された作成セットのアンドゥーの場合、行は可視ではなかった(CTS120はTXID130Aに設定されていた)。したがって、TSS102は、これらの行のCTS120の値を0に更新し、対応するDTS112値をmin STS118A以下に設定し、行をガベージコレクションにすぐに利用可能にし、したがって、メモリおよびリソースを節約し、それらを、より多くのデータにすぐに利用できるようにする。12～13行目では、変更がフラッシュされ得る。

【００６９】

【数８】

```
1: function RecoverTransaction(pTxInfo, ts)
2:     pTxInfo.createSet.Recover();                                  ▷ 作成セットを修復する
3:     pTxInfo.deleteSet.Recover();                                  ▷ 削除セットを修復する
4:     for each row in pTxInfo.deleteSet do
5:         if m_tables[row.tabID]->MVCC[row.rowID].DTS < kMinTS then   ▷ DTSがtxidである場合
6:             m_tables[row.tabID]->MVCC[row.rowID].DTS = ts;        ▷ コミットを終了する、または行の作成を元に戻す
7:     for each row in pTxInfo.createSet do
8:         m_tables[row.tabID]->MVCC[row.rowID].CTS = ts;            ▷ コミットを終了する、または行の作成を元に戻す
9:         if pTxInfo.IsCommitted == TRUE then
10:            m_tables[row.tabID]->MVCC[row.rowID].DTS = kInfinityTS;   ▷ TXがコミットされた場合、DTSを設定する
11:        else
12:            m_tables[row.tabID]->MVCC[row.rowID].DTS = CurMinSTS;     ▷ DTSをCurMinSTSに設定する
13:    MemBarrier();                                                 ▷ mvccエントリに対するすべての更新が確実に実行を終了するようにする
14:    for each row in pTxInfo.createSet ∪ pTxInfo.deleteSet do      ▷ 変更された行のmvccをフラッシュする
15:        Flush(&m_tables[row.tabID]->MVCC[row.rowID]);
16:    MemBarrier();                                                 ▷ すべてのフラッシュが確実に実行を終了するようにする
17:    pTxInfo.createSet.Clear();
18:    pTxInfo.deleteSet.Clear();
```

アルゴリズム8A

【００７０】
アルゴリズム8Aは、一実施形態による、TSS102によって実行され得る例示的なトランザクション回復機能である。アルゴリズム8Aは、TSS102が、システム障害、電力損失、またはリブート中に発生している作動中のトランザクションの処理および回復を可能にし得る。

【００７１】
システムがリブートまたは再起動すると、TSS102は、コミットフラグ136が特定のトランザクションに対して設定されていないと判定し得る。2～3行目において、TSS102は、NVM108から書込みセット132(作成セットおよび削除セット)を取り出し得る。削除された行のDTS112および作成された行のCTS120について、値がタイムスタンプまたはトランザクションID130Aに設定されるかどうかが判定される。

【００７２】
上述のように、一実施形態では、TSS102は、整数のドメインを2つの部分に分割することによって、タイムスタンプとTXID130Aとを区別し得る。たとえば、1～2^63の値はトランザクション識別子であり、2^63～2^64の値はタイムスタンプ値からの値であり得る。次いで、TS120およびDTS112の値は、その値がタイムスタンプであるかTXID130Aであるかを判定するために、それらが中間値よりも小さいか大きいかが判定され得る。これは、書込みセット132の各行について実行され得る。別の実施形態では、CTS120およびDTS112の値は、それらがNVM108に保持され得るTXID130に等しいかどうかがチェックされ得る。

【００７３】
DTS112が最小タイムスタンプよりも小さい場合、それはタイムスタンプ値ではなく、TXID130Aである。これは、クラッシュ時の間に、行がこのトランザクションによってロックされたことを示す。値がタイムスタンプでない場合、TSS102は、コミットフラグ136をチェックすることによってコミットされたトランザクションであるかどうかを判定し得る。トランザクションは、コミットされていない場合、中止され、ロールバックされ得る。

【００７４】
一実施形態では、トランザクションがコミットされる場合、TSS102は、コミットをロールフォワード/完了し、トランザクションをそこで終了または完了し得る。あるいは、たとえば、トランザクションがコミットされていない場合、TSS102は、(1)書込みセット132

、CTS120、およびコミットフラグ136が保持されている場合、トランザクションをロールバックし、または(2)障害時に実行していたステートメントをロールバックし、ユーザがトランザクションを続行したいか、(ステートメントまたはトランザクションを)中止したいかを決定できるようにし得る。

【００７５】

一実施形態では、ステートメントID140が読取りセット126と書込みセット132の両方の一部として記憶されると、ステートメントのロールバックが行われ得る。これらのセット126および132、ならびに対応するステートメントID140の両方は、トランザクションオブジェクト124の一部としてNVM108に保持され得る。ステートメントロールバックは、ステートメントID140が記憶されていない他の実施形態では含まれない追加の機能である。

【００７６】

【数９】

```
1: function RecoverDB
2:     RecoverWorkingPointers();                    ▷相対的なPPtrを変換することによって作業ポインタを設定する
3:     for each pTab in m_tables do                 ▷テーブルコンテンツを修復する
4:         pTab->Recover();
5:     for each pTxInfo in m_persTxInfoArray do     ▷コミットされたTXを終了する
6:         if pTxInfo.IsCommitted == TRUE then
7:             RecoverTransaction(pTxInfo, pTxInfo.commitTS);
8:     for each pTxInfo in m_persTxInfoArray do     ▷コミットされていないTXを元に戻す
9:         if pTxInfo.IsCommitted == FALSE then
10:            RecoverTransaction(pTxInfo, kInfinityTS);
11:    m_txInfoArray = new TxTransientInfo[m_MaxParallelTX];   ▷新しいTxInfoArrayを構築する
12:    for (txid = 0; txid < m_MaxParallelTX; ++txid) do       ▷各TxInfoのtxidを初期化する
13:        m_txInfoArray[i].txid = txid;
14:    m_currentTime = MaxCommitTS();               ▷現在の時間をpersTxInforArray内のmax commitTSに設定する
15:    m_curMinSTS = m_currentTime;                 ▷CurMinSTSを現在の時間に設定する
16:    thread recoverGarbage(m_curMinSTS, tableSizes);   ▷背景におけるガーベッジを修復する
17:    RecoverSecondaryDataStructures();            ▷一時インデックスまたはハイブリッドインデックスを修復する
```

アルゴリズム8B

【００７７】

アルゴリズム8Bは、一実施形態による、TSS102によって実行される一般的な回復機能であり得る。6行目は、TSS102がコミットフラグ136の値が真に等しいかどうかをチェックし得、その場合、アルゴリズム8Aがタイムスタンプ値CTS120Aで実行され得ることを示す。9～10行目では、コミットフラグ136が偽に設定されている場合、回復トランザクション(アルゴリズム8A)は、TXID130Aの値を置き換えるために使用され得る無限大のタイムスタンプとともに呼び出される。一実施形態では、CTS120が無限大に設定されている場合、トランザクションは、この行を見ることさえできない。

【００７８】

どのタイムスタンプ(CTS120Aまたは無限大)が提供されているかに応じて、アルゴリズム8Aは、それぞれトランザクションを続行する、またはロールバックする。ひとたびコミットフラグ136が真に設定されると、他のトランザクションは、トランザクションがタイムスタンプを永続的に更新していたコミットの最後の部分を終了しなかったとしても、データが保持または更新される前に、特定のトランザクションに対する問合せを行ってその値が可視であるかどうかを調べ、それに基づいて他の決定を行い得る。ひとたびトランザクションオブジェクト124の状態がコミットに変更されると、TSS102は、コミットされた値が確実に保持されるようにし得る。一実施形態では、コミット段階をロールフォワードすることによって、そのような一貫性およびデータの完全性が維持されることを保証し得

る。

【００７９】

TSS102は、削除セット内の行のDTS112を読み取り、値がタイムスタンプ(CTS120A)であるかTXID130Aであるかをテストし得る。値がTXID130Aである場合、ロールフォワードが実行されている場合、値はCTS120Aに更新される。ロールバックが実行されている場合、DTS112は無限大に設定され得る。削除動作はDTS112のみを更新するが、作成動作は特定の行についてCTS120とDTS112の両方を更新する。トランザクションがコミットされた場合、DTS112は、(可視の場合)無限大に設定され、そうでなく、中止し、ロールバックする場合、DTS112は、ガベージコレクションのためにmin STS118A以下に設定され得る。

【００８０】

典型的には、DBがクラッシュすると、その時間中に実行されていた任意のトランザクションが中止され、実行中であった任意の作動中のトランザクションはロールバックされ、中止される。TSS102は、これらの作動中のトランザクションの継続を可能にし、完了しなかったトランザクション内のステートメント(トランザクションは複数のステートメントを有する)に対して、アンドゥーが実行されることを可能にする。一実施形態では、管理者または他のユーザに、システム障害またはロールバック中に実行中であったトランザクションを継続するオプションが提供され得る。

【００８１】

一実施形態では、ステートメントロールフォワード動作を可能にするために、トランザクションオブジェクト124は、ステートメント識別子140を含み得る。上述したように、書込みセット132は、特定の行の行IDおよびテーブルIDへの参照を含み得るが、これは、実行されているステートメントを識別しない可能性がある。したがって、TSS102は、書込みセット132内の行参照ごとのステートメントID140を追跡または維持し得る。

【００８２】

たとえば、トランザクション106は、トランザクションの一部として実行される複数のステートメントまたはアクションを含み得る。TSS102は、各ステートメントに識別子を割り当てるステートメントカウンタ(各トランザクション106のプライベートカウンタであり得る)を含み得る。たとえば、第1のステートメントはステートメント1であり、第2のステートメントはステートメント2であり、以下同様であり得る。次いで、たとえばTSS102は、トランザクション106のどのステートメントによってどの行が変更されたかを決定し得る。この追加の情報を維持することによって、管理者またはユーザは、トランザクションを中止するか、完了したステートメントを保持してトランザクションを続けるかを選択し得る。

【００８３】

一実施形態では、トランザクションオブジェクト124はNVM108に完全に保持され得る。たとえば、その一部のみではなく、トランザクションオブジェクト124の情報のすべてがNVM108に保持されてもよい。この完全な永続性によって、追加の永続性オーバーヘッドが作られる可能性があるが、ロールフォワードが実行される可能性があるので、障害時にリソースを節約し得る。一実施形態では、トランザクションオブジェクト124は、NVM108に記憶されてもよく、トランザクションオブジェクト124の一部(NVM108の保持された部分と重複し得る)は、VM110から参照され得る。

【００８４】

図2は、例示的な一実施形態による、マルチバージョン同時実行制御(MVCC)システムに関連付けられた回復プロセスを実行するためのフローチャートである。方法200は、ハードウェア(たとえば、回路、専用ロジック、プログラマブルロジック、マイクロコードなど)、ソフトウェア(たとえば、処理デバイス上で実行される命令)、またはそれらの組合せを含むことができる処理論理によって実行することができる。本明細書で提供される本開示を実行するためにすべてのステップが必要とされるとは限らないことを諒解されたい。さらに、当業者には理解されるように、いくつかのステップは、同時に実行されてもよく、

または図2に示されるものとは異なる順序で実行されてもよい。方法200について、図1を参照しながら説明する。しかしながら、方法200は例示的な実施形態に限定されない。

【００８５】

210において、イベントが発生したと判定され、この場合、イベントより前に保留中であったマルチバージョンデータベースの1つまたは複数のレコードに対する1つまたは複数の書込みトランザクションがコミットしなかった。たとえば、MDB104のデータに対して(並行して)実行されている複数のトランザクション106があり得る。コンピュータシステムがクラッシュしたり、電源が切れたり、何らかの他のリブートイベントが発生したときに、1つまたは複数のトランザクション106が進行中、作動中、またはさもなければコミットされていない場合がある。

【００８６】

220において、1つまたは複数の書込みトランザクションは、イベントより前に不揮発性メモリに記憶されたコミット値に基づいて識別される。たとえば、システムのリブート時に、書込みセット132が読み取られ得る。各トランザクション106は、それ自体の書込みセット132を有し得る。書込みセット132は、NVM108に記憶されてもよく、システムがクラッシュしたとき、どのレコードまたはタプルがトランザクションに書き込まれているかまたはトランザクションによって変更されたかを識別し得る。NVM108に記憶されたトランザクションオブジェクト124は、トランザクション106がレコードに対するその変更をコミットしたかどうかを示すコミットフラグ136を含み得る。NVM108に記憶され得るコミットフラグ136に基づいて、進行中であったがコミットしていないトランザクション106が識別され得る。

【００８７】

230において、識別されたコミットされていない書込みトランザクションのうちの特定の1つが選択される。たとえば、識別されたコミットされていないトランザクション106のうちの1つがロールバックのために選択され得る。一実施形態では、複数のコミットされていないトランザクションが選択され、(たとえば、異なるプロセッサまたはコンピューティングスレッドによって)並列に処理され得る。

【００８８】

240において、選択されたコミットされていない書込みトランザクションに対応するレコードの第1のバージョンが、書込みセットから識別される。たとえば、選択されたトランザクションは、行1の値を更新することに対応し得る。行1Bは、コミットしなかったトランザクション106によって更新されていた第1のバージョンのデータに対応し得る。

【００８９】

250において、イベントより前にコミットされたレコードの以前のバージョンが識別される。たとえば、行1Aは、クラッシュまたはシステムのリブートより前にコミットされた前のバージョンのレコードである可能性がある。

【００９０】

260において、レコードの可視性は、レコードの以前のバージョンが可視であり、レコードの第1のバージョンが可視でないことを示すように設定される。たとえば、行1Aおよび1BのDTS112は、行1Aが可視として示され、行1Bが不可視として示されるように更新されてもよい。一実施形態では、DTS112は、トランザクション106がロールバックされている場合に行1BがガベージコレクションにすぐにMin STS118A以下に設定されてもよく、したがって、以前に消費されたリソースは、再利用のためにすぐに利用可能になり得る。一実施形態では、無限大のDTS112の値または別の設計値は、行のバージョンが行の最新バージョンであることを示し得る。

【００９１】

図3は、様々な実施形態を実装するのに有用な例示的なコンピュータシステム300である。様々な実施形態は、たとえば、図3に示すコンピュータシステム300など、1つまたは複数のよく知られているコンピュータシステムを使用して実装することができる。コンピュータシステム300は、本明細書に記載された機能を実行することができる任意のよく知ら

れているコンピュータとすることができる。
【００９２】
コンピュータシステム300は、プロセッサ304などの1つまたは複数のプロセッサ(中央処理装置またはCPUとも呼ばれる)を含む。プロセッサ304は、通信インフラストラクチャまたはバス306に接続される。
【００９３】
1つまたは複数のプロセッサ304は各々、グラフィックス処理ユニット(GPU)であり得る。一実施形態では、GPUは、数学的に集中的なアプリケーションを処理するように設計された特殊な電子回路であるプロセッサである。GPUは、コンピュータグラフィックスアプリケーション、画像、ビデオなどに共通する数学的に集中的なデータなど、大きいデータブロックの並列処理に効率的な並列構造を有し得る。あるいは、たとえば、1つまたは複数のプロセッサは、製造後にユーザまたは設計者によって構成され得るフィールドプログラマブルゲートアレイ(FPGA)であり得る。
【００９４】
コンピュータシステム300は、ユーザ入力/出力インターフェース302を介して通信インフラストラクチャ306と通信する、モニタ、キーボード、ポインティングデバイスなどのユーザ入力/出力デバイス303も含む。
【００９５】
コンピュータシステム300は、ランダムアクセスメモリ(RAM)などのメインメモリまたは1次メモリ308も含む。メインメモリ308は、1つまたは複数のレベルのキャッシュを含み得る。メインメモリ308には、制御ロジック(すなわち、コンピュータソフトウェア)および/またはデータが記憶されている。一実施形態では、メインメモリ308は、揮発性メモリ307と不揮発性メモリ309の両方を含み得る。不揮発性メモリ309は、本明細書で説明する永続的メモリ110に対応し得る。揮発性メモリ307は、コンピュータシステム300の電源サイクルでリセットされるかまたは保持しない任意のメモリまたはストレージを含み得る。
【００９６】
コンピュータシステム300は、1つまたは複数の2次記憶デバイスまたはメモリ310も含み得る。2次メモリ310は、たとえば、ハードディスクドライブ312および/またはリムーバブルストレージデバイスもしくはドライブ314を含み得る。リムーバブルストレージドライブ314は、フロッピーディスクドライブ、磁気テープドライブ、コンパクトディスクドライブ、光ストレージデバイス、テープバックアップデバイス、および/または任意の他のストレージデバイス/ドライブであり得る。
【００９７】
リムーバブルストレージドライブ314は、リムーバブルストレージユニット318と対話し得る。リムーバブルストレージユニット318は、コンピュータソフトウェア(制御論理)および/またはデータが記憶されたコンピュータ使用可能または可読記憶デバイスを含む。リムーバブルストレージユニット318は、フロッピーディスク、磁気テープ、コンパクトディスク、DVD、光ストレージディスク、および/または任意の他のコンピュータデータ記憶デバイスであってもよい。リムーバブルストレージドライブ314は、よく知られている方法でリムーバブルストレージユニット318から読み取り、および/またはリムーバブルストレージユニット318に書き込む。
【００９８】
例示的な実施形態によれば、2次メモリ310は、コンピュータプログラムおよび/または他の命令および/またはデータがコンピュータシステム300によってアクセスされることを可能にするための他の手段(means)、手段(instrumentalities)、または他の手法を含み得る。そのような手段(means)、手段(instrumentalities)、または他の手法は、たとえば、リムーバブルストレージユニット322およびインターフェース320を含み得る。リムーバブルストレージユニット322およびインターフェース320の例は、プログラムカートリッジおよびカートリッジインターフェース(ビデオゲームデバイスに見られるものなど)、

リムーバブルメモリチップ(EPROMまたはPROMなど)および関連するソケット、メモリスティックおよびUSBポート、メモリカードおよび関連するメモリカードスロット、ならびに/または任意の他のリムーバブルストレージユニットおよび関連するインターフェースを含み得る。

【００９９】

コンピュータシステム300は、通信またはネットワークインターフェース324をさらに含み得る。通信インターフェース324によって、コンピュータシステム300は、リモートデバイス、リモートネットワーク、リモートエンティティ(個々に、および集合的に参照番号328によって参照される)などの任意の組合せと通信し、対話することが可能になる。たとえば、通信インターフェース324は、コンピュータシステム300が、ワイヤードおよび/またはワイヤレスであり、LAN、WAN、インターネットなどの任意の組合せを含み得る通信経路326を介してリモートデバイス328と通信することを可能にし得る。コンピュータシステム300との間で通信経路326を介して制御論理および/またはデータが送信されてもよい。

【００１００】

一実施形態では、制御ロジック(ソフトウェア)が記憶された有形のコンピュータ使用可能または可読媒体を含む有形の装置または製造品は、本明細書ではコンピュータプログラム製品またはプログラム記憶デバイスとも呼ばれる。これは、限定はしないが、コンピュータシステム300、メインメモリ308、2次メモリ310、およびリムーバブルストレージユニット318および322、ならびに上記の任意の組合せを組み込む有形の製造品を含む。そのような制御論理は、(コンピュータシステム300など)1つまたは複数のデータ処理デバイスによって実行されると、そのようなデータ処理デバイスに、本明細書で説明するように動作させる。

【００１０１】

本開示に含まれる教示に基づいて、データ処理デバイス、コンピュータシステム、および/または図3に示したもの以外のコンピュータアーキテクチャを使用して、本開示の実施形態を作成および使用する方法が当業者には明らかであろう。特に、実施形態は、本明細書に記載されているもの以外のソフトウェア、ハードウェア、および/またはオペレーティングシステムの実装で動作することができる。

【００１０２】

図4は、別の例示的な実施形態による、マルチバージョン同時実行制御(MVCC)システムに関連付けられた回復プロセスを実行するためのフローチャート400である。方法200は、ハードウェア(たとえば、回路、専用ロジック、プログラマブルロジック、マイクロコードなど)、ソフトウェア(たとえば、処理デバイス上で実行する命令)、またはそれらの組合せを含むことができる処理論理によって実行することができる。本明細書で提供される本開示を実行するためにすべてのステップが必要とされるとは限らないことを諒解されたい。さらに、当業者には理解されるように、いくつかのステップは、同時に実行されてもよく、または図4に示されるものとは異なる順序で実行されてもよい。方法400について、図1を参照しながら説明する。しかしながら、方法400は例示的な実施形態に限定されない。

【００１０３】

要素410～440は、図2に関して上述した要素210～240と実質的に同様である。

【００１０４】

450において、識別されたトランザクションが複数のステートメントを含むと判定される。たとえば、トランザクション106は、トランザクションの一部として実行される複数の異なるステートメントを含み得る。一実施形態では、各ステートメントは、それ自体のステートメントID140を有し得る。

【００１０５】

460において、ステートメントの最初のものをロールバックするかロールフォワードするかに関する指示が受信される。たとえば、TSS102は、管理者に、特定のトランザクション106の各ステートメントを処理し続ける(ロールフォワードする)か、中止する(およびロ

ールバックする)かの選択肢を提供し得る。これによって、すでに実行されたステートメントの不要な重複する実行を防止することによって、処理リソースを節約し得る。

【０１０６】

470において、識別されたトランザクションは、指示に基づいて実行される。たとえば、ロールバック指示が受信された場合、(図2に関して上述したように)データの以前の状態が可視であるように設定される。あるいは、たとえば、ロールフォワード指示が受信された場合、トランザクションの最新のステートメントが実行を完了したところで実行を継続し得る。

【０１０７】

詳細な説明のセクションは、任意の他のセクションではなく、特許請求の範囲を解釈するために使用されることが意図されていることを諒解されたい。他のセクションは、発明者によって企図されるように、全部ではなく1つまたは複数の例示的な実施形態を示すことができ、したがって、本開示または添付の特許請求の範囲をいかなる形でも限定しないものとする。

【０１０８】

本開示は、例示的な分野および用途についての例示的な実施形態を記載しているが、本開示はそれに限定されないことを理解されたい。他の実施形態およびそれへの修正が可能であり、本開示の範囲および意図の範囲内である。たとえば、本段落の一般性を制限することなく、実施形態は、図に示され、および/または本明細書で説明されるソフトウェア、ハードウェア、ファームウェア、および/またはエンティティに限定されない。さらに、実施形態(本明細書に明示的に記載されているかどうかにかかわらず)は、本明細書に記載されている例を超えて、分野および用途に対する重要な有用性を有する。

【０１０９】

特定の機能の実装およびその関係を示す機能的構成単位の助けを借りて、本明細書で実施形態を説明してきた。これらの機能的構成単位の境界は、説明の便宜のために、本明細書において任意に定義されている。指定された機能および関係(またはその均等物)が適切に実行される限り、代替の境界を定義することができる。また、代替の実施形態は、本明細書に記載された順序とは異なる順序を使用して、機能ブロック、ステップ、動作、方法などを実行することができる。

【０１１０】

「一実施形態」、「実施形態」、「例示的な実施形態」、または同様の句への本明細書での言及は、記載された実施形態が特定の特徴、構造、または特性を含むことができることを示しているが、あらゆる実施形態が必ずしも特定の特徴、構造、または特性を含むとは限らない可能性がある。さらに、そのような句は、必ずしも同じ実施形態を指しているとは限らない。さらに、特定の特徴、構造、または特性が実施形態に関連して記載されているとき、そのような特徴、構造、または特性を他の実施形態に組み込むことは、本明細書において明示的に言及または記載されているかどうかにかかわらず、当業者の知識の範囲内であろう。さらに、いくつかの実施形態は、それらの派生語とともに「結合された」および「接続された」という表現を使用して説明することができる。これらの用語が必ずしも互いに同義語として意図されているとは限らない。たとえば、いくつかの実施形態は、2つ以上の要素が互いに物理的または電気的に直接接触していることを示すために「接続された」および/または「結合された」という用語を使用して説明することができる。しかしながら、「結合された」という用語は、2つ以上の要素が互いに直接接触していないが、依然として互いに協働しまたは相互作用していることも意味し得る。

【０１１１】

本開示の幅および範囲は、上記の例示的な実施形態のいずれによっても制限されないものとするが、以下の特許請求の範囲およびそれらの均等物に従ってのみ定義されるものとする。

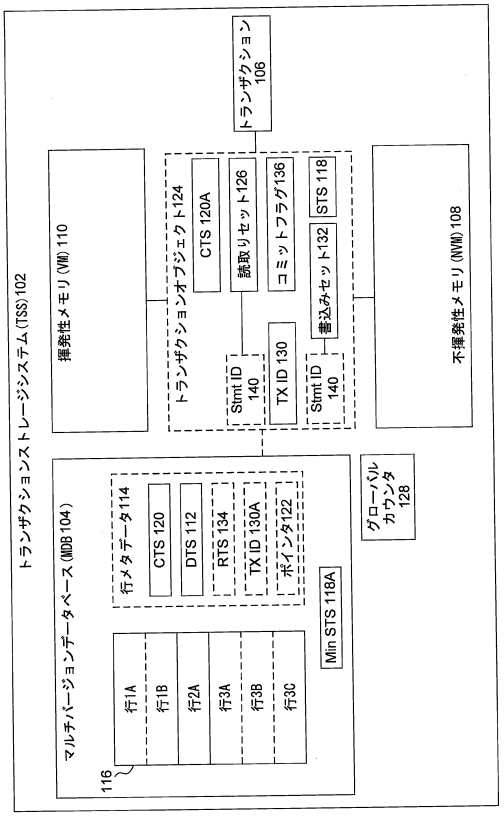
【符号の説明】

【０１１２】

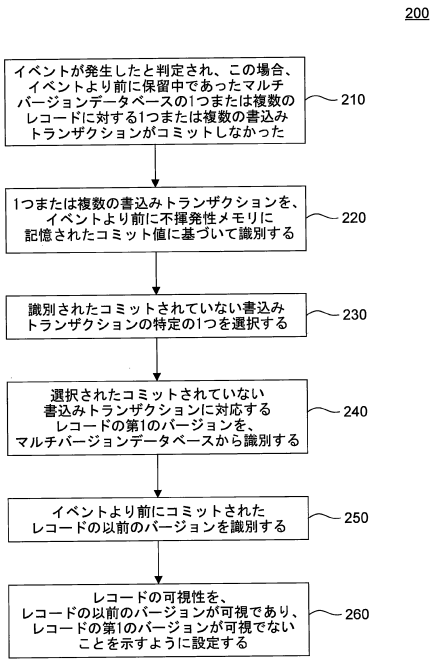
102 トランザクションストレージシステム(TSS)
104 マルチバージョンデータベース(MDB)
106 トランザクション
108 不揮発性メモリ(NVM)
110 揮発性メモリ(VM)
112 削除タイムスタンプ(DTS)
114 行メタデータ
116 行/レコード
118 開始タイムスタンプ(STS)
120 作成またはコミットタイムスタンプ(CTS)
122 ポインタ
124 トランザクションオブジェクト
126 読取りセット
128 グローバルカウンタ/グローバルタイムスタンプカウン
130 トランザクション識別子(TXID)
132 書込みセット
134 読取りタイムスタンプ(RTS)
136 コミットフラグ
140 ステートメント識別子(ID)
200 方法
210～240 要素
300 コンピュータシステム
302 ユーザ入力/出力インターフェース
303 ユーザ入力/出力デバイス
304 プロセッサ
306 通信インフラストラクチャまたはバス
307 揮発性メモリ
308 メインメモリまたは1次メモリ
309 不揮発性メモリ
310 2次記憶デバイスまたはメモリ
312 ハードディスクドライブ
314 リムーバブルストレージデバイスまたはドライブ
318 リムーバブルストレージユニット
320 インターフェース
322 リムーバブルストレージユニット
324 ネットワークインターフェース
326 通信経路
328 リモートデバイス
410～440 要素

【図面】

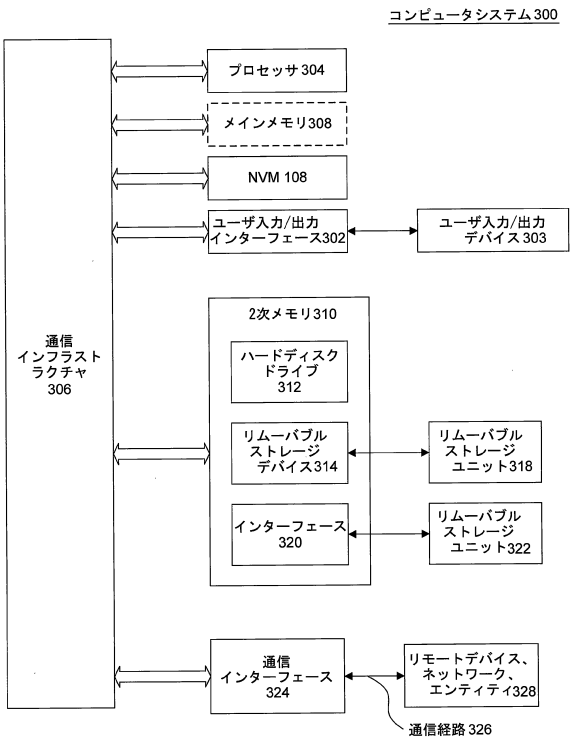
【図１】


トランザクションストレージシステム(TSS)102
揮発性メモリ(VM)110
トランザクションオブジェクト124
CTS 120A
読取りセット126
コミットフラグ136
STS 118
書込みセット132
Stmt ID 140
TX ID 130
Stmt ID 140
トランザクション 106
不揮発性メモリ(NVM)108
マルチバージョンデータベース(MDB 104)
行メタデータ114
CTS 120
DTS 112
RTS 134
TX ID 130A
ポインタ122
グローバルカウンタ 128
Min STS 118A
行1A
行1B
行2A
行3A
行3B
行3C
116

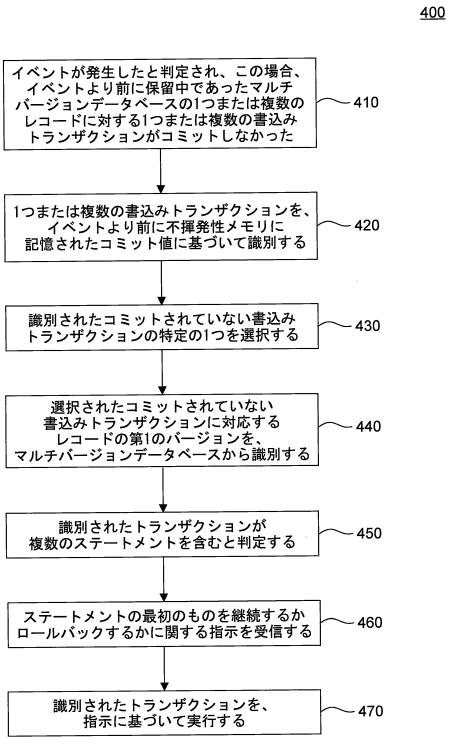
【図２】


200
イベントが発生したと判定され、この場合、イベントより前に保留中であったマルチバージョンデータベースの1つまたは複数のレコードに対する1つまたは複数の書込みトランザクションがコミットしなかった
210
1つまたは複数の書込みトランザクションを、イベントより前に不揮発性メモリに記憶されたコミット値に基づいて識別する
220
識別されたコミットされていない書込みトランザクションの特定の1つを選択する
230
選択されたコミットされていない書込みトランザクションに対応するレコードの第1のバージョンを、マルチバージョンデータベースから識別する
240
イベントより前にコミットされたレコードの以前のバージョンを識別する
250
レコードの可視性を、レコードの以前のバージョンが可視であり、レコードの第1のバージョンが可視でないことを示すように設定する
260

【図３】


コンピュータシステム300
通信インフラストラクチャ306
プロセッサ304
メインメモリ308
NVM 108
ユーザ入力/出力インターフェース302
ユーザ入力/出力デバイス303
2次メモリ310
ハードディスクドライブ312
リムーバブルストレージデバイス314
インターフェース320
リムーバブルストレージユニット318
リムーバブルストレージユニット322
通信インターフェース324
リモートデバイス、ネットワーク、エンティティ328
通信経路326

【図４】


400
イベントが発生したと判定され、この場合、イベントより前に保留中であったマルチバージョンデータベースの1つまたは複数のレコードに対する1つまたは複数の書込みトランザクションがコミットしなかった
410
1つまたは複数の書込みトランザクションを、イベントより前に不揮発性メモリに記憶されたコミット値に基づいて識別する
420
識別されたコミットされていない書込みトランザクションの特定の1つを選択する
430
選択されたコミットされていない書込みトランザクションに対応するレコードの第1のバージョンを、マルチバージョンデータベースから識別する
440
識別されたトランザクションが複数のステートメントを含むと判定する
450
ステートメントの最初のものを継続するかロールバックするかに関する指示を受信する
460
識別されたトランザクションを、指示に基づいて実行する
470

フロントページの続き

ドイツ・６９１９０・ヴァルドルフ・ディートマール－ホップ－アレー・１６・エスアーペー・エスエー
(72)発明者　ヴォルフガング・レーナー
ドイツ・６９１９０・ヴァルドルフ・ディートマール－ホップ－アレー・１６・エスアーペー・エスエー
(72)発明者　ダニエル・ドス・サントス・ボッスル
ドイツ・６９１９０・ヴァルドルフ・ディートマール－ホップ－アレー・１６・エスアーペー・エスエー

審査官　早川 学

(56)参考文献　米国特許出願公開第２０１５／０３５５９８１（ＵＳ，Ａ１）
特開２０１７－０５４２０７（ＪＰ，Ａ）
特開２０１７－０２７３２６（ＪＰ，Ａ）
再公表特許第２０１６／１６２９８１（ＪＰ，Ａ１）
特開２０１４－２１５９１４（ＪＰ，Ａ）
米国特許出願公開第２０１０／０１０６７５３（ＵＳ，Ａ１）
特開平６－３１８１６５（ＪＰ，Ａ）

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１６／００－１６／９５８