(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4306023号
(P4306023)

(45)発行日　平成21年7月29日(2009.7.29)　(24)登録日　平成21年5月15日(2009.5.15)

(51)Int.Cl.		F I		
G06F 12/00	(2006.01)	G06F	12/00	535C
G06F 17/30	(2006.01)	G06F	12/00	518A
		G06F	12/00	545A
		G06F	17/30	240A

請求項の数 1　(全 20 頁)

(21)出願番号　特願平11-159008
(22)出願日　平成11年6月7日(1999.6.7)
(65)公開番号　特開2000-347909(P2000-347909A)
(43)公開日　平成12年12月15日(2000.12.15)
審査請求日　平成17年11月30日(2005.11.30)

(73)特許権者　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　100100310
弁理士　井上　学
(72)発明者　猪原　茂和
東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内
(72)発明者　西澤　格
東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内

最終頁に続く

(54)【発明の名称】トランザクション処理向けストレージ方法および装置、トランザクショナルストレージ

(57)【特許請求の範囲】
【請求項１】
計算機システムであって、
計算機と、
前記計算機に接続されるストレージシステムを備え、
前記ストレージシステムは、外部装置に接続される制御装置と、
前記制御装置に接続されるディスク装置と、を有し、
前記ディスク装置は、コンピュータのトランザクション処理の対象データで、一つ以上のカラムからなるレコードを複数有するテーブルを複数備えたデータベースを、固定のブロックを単位として、前記レコードのうち少なくとも一以上を各ブロックに格納し
前記制御装置は、
コミット前のデータを保持する第一のキャッシュ領域とコミット後のデータを保持する第二のキャッシュ領域と、
前記ブロックを一意に識別するブロックＩＤと、前記ブロック内でレコードを一意に識別するレコードＩＤとを含むレコードの指定を、前記計算機から受け、レコードを選択する第１のインタフェースと、
テーブルを一意に識別するテーブルＩＤと、レコード絞り込みのための条件とを含むレコードの指定とを前記計算機から受け、該当するレコードを選択する第２のインタフェースと、
インデックスを一意に識別するインデックスＩＤと該インデックスを検索するためのレコ

ード絞り込みのための条件とを含むレコードの指定を前記計算機から受け、レコードを選択する第３のインタフェースと、
インデックスの中間ノードを一意に識別するブロックＩＤと、該インデックスを検索するためのレコード絞り込みのための条件とを含むレコードの指定を受けてレコードを選択する第４のインタフェースと、により構成される入出力制御部と、
前記第１、第２、第３及び第４のインタフェースいずれかで選択されたレコードが前記第１のキャッシュ領域に存在するか否かを判断し、存在する場合は、前記第１の記憶領域に存在するレコードに対する制御権を取得し、存在しない場合は、前記第２の記憶領域に存在するか否かを判断し、存在する場合は、前記第２の記憶領域に存在するレコードに対する制御権を取得し、前記選択されたレコードを前記第１の記憶領域に格納し、前記第２の記憶領域に存在しない場合は、前記ディスク装置から該当するレコードを取得するトランザクション処理部と、を備え、
前記計算機は、前記データベースの少なくとも一部に関連するインデクス情報を保持し、前記インデクス情報を用いてデータベースを構成するレコードのいずれかにアクセスする場合、前記インデクス情報が前記データベースの全てに関連する場合は、前記インデクス情報を検索し、該当するブロックＩＤとレコードＩＤとを含むレコードの指定を含む前記ストレージシステムに読み出しリクエストを発行し、
前記インデクス情報が前記データベースの一部に関連する場合は、前記インデクス情報を検索し、スキャン要求とともに、検索で得られたレコードＩＤを含むレコードの指定を含む読み出しリクエストを前記ストレージシステムに発行し、
前記インデクス情報を前記計算機が有さない場合は、スキャン要求とともに前記ストレージシステムに読み出しリクエストを発行する入出力管理部を有する、ことを特徴とする計算機システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明はコンピュータシステムに関し、特に応用プログラムに適した新たな機能を追加・拡張可能な二次記憶装置及び方法に関する。

【０００２】

【従来の技術】

今日のコンピュータシステムの主要な構成要素は、コンピュータ（プロセッサとメモリ、周辺機器からなる）とネットワークと二次記憶装置（ストレージ）である。これまでストレージは、コンピュータに付属する装置として存在することが多かった。しかし近年、このような状況が変化しつつある。

【０００３】

第１に、ネットワークの普及により、複数のコンピュータがストレージを共有する機会が増えた。他のコンピュータからのネットワーク経由のストレージ入出力が、ストレージを接続したコンピュータのプロセッサ処理能力がボトルネックとなって滞る事態を招いている。第２に、ストレージ容量およびストレージに対する要求スループットは、年々増大している。「データウェアハウス用途のストレージ容量の要求は、９ヶ月で２倍になる」という予測（Ｇｒｅｇ’ｓ　Ｌａｗ）もある。このため、１つのコンピュータに接続するストレージ数が増大し、やはりコンピュータがストレージ入出力のボトルネックとなる恐れが出てきている。第３に、ハードディスク制御用ＬＳＩの高集積化の進行によって、ストレージの高機能化の可能性の増大している。

【０００４】

これらの背景から、ストレージの制御用ＬＳＩに新たな機能を付加することが考えられている。新たな機能の候補は、ネットワークインタフェースと、応用プログラム向けの高度機能である。

【０００５】

ネットワークインタフェースをストレージが備えることにより、ストレージをネットワー

クに直結することができる。これによりストレージは、複数のコンピュータからの入出力要求を、１つのコンピュータを介することなく、直接受けることができる。

【０００６】

また、現在、ストレージとコンピュータとの間の最も代表的なインタフェースはブロック入出力であるが、これよりも応用毎の高度機能（例えばソーティング、画像処理、データベースシステムの基本演算、例えば選択処理、写像処理、結合処理、集計処理等）をストレージが備えることにより、コンピュータのプロセッサ処理の一部をストレージが受け持つことが可能となる。

【０００７】

一方、ストレージの利用分野の中で、トランザクション処理（データベースのトランザクション処理やオンライントランザクション処理システムのトランザクション処理）は特に高い性能が要請される分野である。データベース処理でもトランザクション処理でも、一連の処理はトランザクションという単位で実行される。トランザクションは、トランザクション開始からトランザクション終了までの間に、データベースに対して１つ以上の参照や更新を行う。トランザクションの終了には２種類の方法があり、１つはコミット（正常終了）、もう１つはアボート（異常終了）である。コミットの場合、トランザクション中に行われた参照や更新が実際に行われたことになる。アボートの場合、トランザクション中に行われた参照や更新はすべて取り消される。

【０００８】

トランザクションは一般に、ＡＣＩＤ性（Ａｔｏｍｉｃｉｔｙ、Ｃｏｎｓｉｓｔｅｎｃｙ、Ｉｓｏｌａｔｉｏｎ、Ｄｕｒａｂｉｌｉｔｙ）の４つの性質を実現している。これらの性質により、あるトランザクションは並行して実行中の別のトランザクションの変更を見ることはないし、あるトランザクションがコミットした場合には、該トランザクションが行った変更はすべてデータベース中、特にステーブルストレージ（電源断や、ソフトウェア、ハードウェアの故障等、一定範囲の障害に耐えうる記憶装置）に保存される。

【０００９】

トランザクションを高性能化することは、企業の情報システムの構築にとって非常に重要である。なぜなら、トランザクションはコンピュータが関係する企業活動の、ほとんどすべての場面で用いられているためである。例えば、銀行のＡＴＭを用いた入金、出金はそれぞれ１つのトランザクションであるし、スーパーマーケットで商品を購入する際、レジスターでは商品の計算をすると同時にＰＯＳシステムにどの商品が何個売れたかをトランザクションで記録している。企業活動のほとんどが、トランザクションによって記録、管理されていると言っても過言ではない。

【００１０】

トランザクションが処理の対象とするのは、データベースシステムの１かたまりのデータである。例えばリレーショナルデータベースでは、１つ１つの型のデータ（整数型、文字列型、固定長小数点型等）はカラムと呼ばれ、カラムが１つ以上集まってできる１つの論理的なデータの単位をレコードと呼ぶ。１つ以上のレコードは１つの表に格納される。例えば従業員のデータベースであれば、従業員全員のデータを１つの表「従業員表」に格納し、従業員表の１つのレコードが１人の従業員に対応し、「氏名」、「従業員番号」、「生年月日」、「住所」、「性別」、「役職」等のカラムが該レコードに含まれる、といった様に、データベースが構成される。１つのトランザクションは例えば、「従業員番号が１０００番のレコードの役職が「係長」であれば「課長」に変更せよ」といった一連のデータベース処理を実現できる。

【００１１】

【発明が解決しようとする課題】

トランザクションは高度なソフトウェア処理であるため、コンピュータのプロセッサに高い負荷を生じる。また、データベースを参照、更新する処理であるため、ストレージに対する負荷も高い。

【００１２】

現在多くのコンピュータシステムは、ブロック（例えば２ＫＢや４ＫＢ等の固定長のデータ）を単位としてコンピュータとストレージの間の入出力を行っている。ブロックは多くの場合多数のレコードを含む。いま、コンピュータがあるトランザクションを実行し、あるレコードに対する参照が行われたとする。この際、コンピュータはストレージから、該レコードを格納したブロックの読み出しを行い、該参照を完了する。ここで、該レコードがブロックのサイズの１／１０だったと仮定すると、コンピュータからストレージに転送したデータのうち、１／１０のみが活用され、残りの９／１０は読み出しを行ったにもかかわらずまったく使われない無駄なデータだったことになる。このことは、大規模なデータベースシステムで参照局所性の低いトランザクション群を実行している場合に顕著になる。書き込みの場合も、同様である。コンピュータ、ストレージとも、データの読み出しおよび書き込みには処理装置と入出力信号線を使用するため、無駄なデータの読み出し、書き込みは極力削減されるべきである。

【００１３】

すなわち、ブロック入出力による無駄なデータの入出力を削減することが本発明が解決すべき第１の課題である。

【００１４】

また、従来のストレージでは読み出しと書き込みのインタフェースを提供しているが更新（読み出しと書き込みを同時に行う）のインタフェースはまれである。トランザクションでは更新が非常に多いが、従来は更新対象のブロックをストレージから読み出して、プロセッサが書きかえて、ストレージに書き込むという２回の入出力操作で実現していた。この入出力は、理想的には「更新」という１回の入出力で実現できる。特に、レコード単位の更新の機能をストレージが提供すれば、トランザクションが行う入出力のデータ量は、従来に比べ大幅に削減できる。

【００１５】

すなわち、更新に伴う無駄な入出力を削減することが本発明が解決すべき第２の課題である。

【００１６】

またデータベースシステムでは、レコードの指定には、何通りもの方法が提供されている。あるテーブルのあるレコードを指定するために、例えば、テーブル自身をスキャン（補助データ構造を使わない方法）、インデックスを経由してレコードに到達する方法、ハッシュインデックスを経由してレコードに到達する方法等である。ストレージが仮にレコード単位の入出力を提供したとしても、レコードの指定方法が１通りだけでは、十分に高速な入出力は見込めない。

【００１７】

すなわち、レコードの指定方法として、直接的なレコードの指定のみならず、データベースが利用するテーブルおよびインデックスの探索方法を提供し、効率的なレコードアクセスを実現することが、本発明が解決すべき第３の課題である。

【００１８】

また、データベースでは、一般に複数のストレージ装置を用いるので、表やインデックスの定義情報（ディクショナリ情報）はすべてのストレージ装置に存在するとは限らない。レコード単位の入出力やインデックスの利用には、ディクショナリ情報が必要であるが、従来はこのような高度な情報をやりとりする方法は、ストレージには存在しなかった。

【００１９】

すなわち、任意のストレージでディクショナリ情報を利用可能にすることが、本発明が解決すべき第４の課題である。

【００２０】

また上記のように、トランザクションの実行中に行われる更新は、アボートによって取り消される可能性がある。このため、従来はあるトランザクション中で起こった更新をディスク中のデータベース（以後ディスク中のデータベースのことをステーブルデータベースと呼ぶ）に書き込んでおり、該トランザクションがアボートした場合、データベースシス

テムは、別に作成しておいたログ（トランザクションの活動記録）を元にステーブルデータベースに対して該トランザクションが行った変更を元に戻す一連の入出力を行う。これもストレージがブロック単位の入出力（読み出しと書き込み）のインタフェースのみを提供しているためである。このアボートに伴う入出力も、ストレージがトランザクションのコミットやアボートを意識したインタフェースを提供していれば削減できる無駄な入出力である。

【００２１】

すなわち、トランザクションの実現（コミットやアボート）による無駄なデータの入出力を削減することが本発明が解決すべき第５の課題である。

【００２２】

また同様に、アボートする際の効率をあげるためにはコミット前のデータはできるだけステーブルデータベースに書き込むべきではないことが分かる。従来のストレージはトランザクションの概念を理解していないため、書き込みを指示されたブロックはその時に書き込まれ（物理的にディスクに書き込まれるか、バッテリーバックアップされたキャッシュメモリ領域に格納される）あとから書き込みを取り消す方法は提供されていなかった。

【００２３】

すなわち、トランザクションがストレージに書き込んだデータを取り消す方法を提供しトランザクションがアボートした際の無駄な入出力を削減することが、本発明の解決すべき第６の課題である。

【００２４】

１つのストレージはアベイラビリティ向上の目的で、複数のコンピュータから共有されることがある。また、１つのコンピュータ上のデータベースシステムにおいても、ブロック単位入出力とレコード単位入出力を併用する場合が多いと考えられる。この際、レコード単位で操作しているレコードをブロック単位で別の経路からアクセスすると、データの不整合が起こりうる。複数のコンピュータが同一のデータにアクセスしようとする場合もまったく同様である。

【００２５】

すなわち、レコード単位入出力とブロック単位入出力の整合性を持たせ、同一のデータをレコード単位でもブロック単位でも処理でき、複数のコンピュータからも共有できるようにすることが、本発明の解決すべき第７の課題である。

【００２６】

【課題を解決するための手段】

本発明では、トランザクションを意識した機能およびインタフェースを持つストレージ「トランザクショナルストレージ」とそれを用いたコンピュータシステムによって、上記の課題を解決する。

【００２７】

ブロック入出力による無駄なデータの入出力を削減する第１の課題を解決するため、トランザクショナルストレージはレコード単位の入出力機能およびインタフェースを備える。

【００２８】

更新に伴う無駄な入出力を削減する第２の課題を解決するため、トランザクショナルストレージはレコードを更新する機能およびインタフェースを備える。更新時に、必要に応じて更新前データを返す。更新前データは、トランザクションの活動記録であるログを作成する際等に利用できる。

【００２９】

レコードの指定方法として、直接的なレコードの指定のみならず、データベースが利用するテーブルおよびインデックスの探索方法を実現し、効率的なレコードアクセスを実現する第３の課題を解決するため、トランザクショナルストレージは（ａ）ブロック番号とレコード番号によるレコード指定インタフェース、（ｂ）テーブルと条件によるレコード指定インタフェース、（ｃ）インデックスと条件によるレコード指定インタフェース、（ｄ）インデックス中間ノードと条件によるレコード指定インタフェース、の４種類のレコー

ド指定インタフェースを備える。
【００３０】
任意のストレージでディクショナリ情報を利用可能にする第４の課題を解決するため、トランザクショナルストレージはコンピュータからディクショナリ情報を入力するインタフェースを備える。
【００３１】
トランザクションの実現（コミットやアボート）による無駄なデータの入出力を削減する第５の課題を解決するため、トランザクショナルストレージは、コミットのインタフェースとアボートのインタフェースを備える。
【００３２】
トランザクションがストレージに書き込んだデータを取り消す方法を提供しトランザクションがアボートした際の無駄な入出力を削減する第６の課題を解決するため、トランザクショナルストレージは、コミット前キャッシュとコミット後キャッシュを備える。コミット前キャッシュの内容はストレージには書き込まれないため、アボートの処理はディスクの操作を伴わずに実現可能となる。
【００３３】
レコード単位入出力とブロック単位入出力の整合性を持たせ、同一のデータをレコード単位でもブロック単位でも処理でき、複数のコンピュータからも共有できるようにすること第７の課題を解決するために、トランザクショナルストレージはレコードおよびブロックのロック（排他制御）インタフェースを備える。
【００３４】
以上により、トランザクション処理に伴ってコンピュータのプロセッサおよびストレージに生じる高い負荷を削減し、大規模なデータベースシステムおよびトランザクション処理システムの実現が可能となる。
【００３５】
【発明の実施の形態】
本発明の実施の一形態を，図面を参照しながら説明する。なお簡単のため、以下に述べる発明の実施の形態を単に「実施例」と呼ぶ。
【００３６】
全体構成
図１を用いて、本実施例の全体構成を説明する。
【００３７】
図１の全体１０１は、本実施例が好適に用いられるコンピュータシステムであり、入出力信号線１０３と、入出力信号線１０３によって相互接続された１つ以上のコンピュータ１０２、１０２’、…および１つ以上のトランザクショナルストレージ（ＴＳ）１０４からなる。
【００３８】
入出力信号線１０３は、コンピュータとストレージを専用につなぐケーブル（ＳＣＳＩケーブル等）でも構わないし、ネットワークでも差し支えない。ネットワークは、ある団体（企業や学校や類似の団体）の全体や位置部門でよく使用されるＬＡＮでもよく、また地理的に分散した複数の地点を結合するＷＡＮの一部または全部でもよい。また入出力信号線１０３は、計算機間結合網や並列計算機内部のプロセッサ要素間の結合網でもよい。
【００３９】
コンピュータ１０２、１０２’、…は、いわゆるパーソナル・コンピュータ、ワークステーション、並列計算機、大型計算機、小型携帯型コンピュータ等、任意のコンピュータでよい。
【００４０】
なお、図１に示したコンピュータ１０２、１０２’、…、入出力信号線１０３、ＴＳ１０４の数と構成は、例として示したもので、本発明の範囲を限定するものではない。
【００４１】

ＴＳ１０４は、拡張型の二次記憶装置（ストレージ）である。ＴＳ１０４は１つ以上のディスク１１１とトランザクショナルストレージコントローラ（ＴＳＣ）１０５とからなる。

【００４２】

ディスク１１１は電源断後もデータを保持することが可能な記憶媒体（二次記憶）である。ディスク１１１のデータ格納単位には、セクタ、トラック等さまざまな呼称があるが、本実施例では一括してブロックと記す。ディスク１１１は複数のブロックからなり、ブロック単位の入出力を行うことができる。二次記憶がハードディスクであれば、多くの場合ブロックは固定長で５１２バイトないし４Ｋバイトである。メインフレーム計算機用のハードディスクであれば、ブロックは固定長の場合と可変長の場合がある。また、テープドライブ等の二次記憶もブロックはその装置毎に決まっている。なお、ブロック単位でなくバイト単位の入出力インタフェースを提供する二次記憶もあるが、ブロックを１バイトと考えることによって本発明を適用することができる。

【００４３】

ＴＳＣ１０５はＴＳ１０４の制御を行う部分である。ＴＳＣ１０５はさらにネットワーク制御部１０６、トランザクション処理部１０７、コミット後キャッシュ１０８、コミット前キャッシュ１０９、ディスク制御部１１０からなる。

【００４４】

ネットワーク制御部１０６は、コンピュータ１０２、１０２’、…をはじめとする外部から入出力信号線１０３経由で送られてくる入出力要求やその他の通信を受け、また入出力要求への応答やその他の通信を入出力信号線１０３へ送り出す。トランザクション処理部１０７は、ＴＳ１０４が提供する各種機能を実現する部分である。各種機能の詳細については、後で詳しく述べる。コミット後キャッシュ１０８は、コミットしたトランザクションが行った更新を保持する記憶領域である。コミット前キャッシュ１０９は、まだコミットしていないトランザクションが行った更新を保持する記憶領域である。ディスク制御部１１０は、ディスク１１１にブロックの読み出し・書き込みを行わせる制御を行う。ネットワーク制御部１０６、ディスク制御部１１０については、従来技術としてよく知られているため、ここではこれ以上詳しく説明しない。

【００４５】

入出力処理部１１２は、コンピュータ１０２、１０２’、…に存在し、ＴＳ１０４を利用する。典型的には入出力処理部１１２はデータベース管理システムのソフトウェアの一部である。

【００４６】

トランザクショナルストレージインタフェース１１３は、ＴＳ１０４とコンピュータ１０２、１０２’、…をはじめとする外部とのインタフェースである。トランザクションの処理を行うため、レコード操作、カラム操作、トランザクション操作等を含む。

【００４７】

以上が本実施例の全体構成である。

【００４８】

トランザクショナルストレージのデータ構造

図２を用いて、トランザクショナルストレージの内部データ構造であるロックテーブル、トランザクションテーブル、ＤＢテーブルテーブル、インデックステーブルの構成についてを説明する。

【００４９】

ロックテーブル２００はブロックまたはレコードと、ロックとの対応表である。ロックテーブル２００の１つの行が１つのロックに対応する。ブロックＩＤ２０１はストレージ上のブロックの一意な番号である。レコードＩＤ２０２はブロック中のレコードの一意な番号である。ブロックＩＤとレコードＩＤの対によって、レコードを一意に特定することができる。レコードＩＤ２０２に指定が無い場合は、ブロックを指定したロックであることを意味する。ロック情報２０３はロックの情報である。典型的には、トランザクションＩ

Ｄとロックモードを格納する。
【００５０】
トランザクションテーブル２１０は、活動中のトランザクションの表である。各トランザクションには一意な識別子が割り当てられており、それがトランザクションＩＤ２１１に格納されている。トランザクションがアクセス中のブロックまたはレコードがブロックＩＤ２１２とレコードＩＤ２１３で保持される。ブロックＩＤ２１２とレコードＩＤ２１３は対で１つのレコードを特定する。
【００５１】
ＤＢテーブルテーブル２２０は、データベースの表を管理する。それぞれの行が１つの表に対応する。各表は一意な名前であるテーブルＩＤが与えられており、それがテーブルＩＤ２２１に格納されている。先頭ブロックＩＤ２２２は表が格納されている先頭のブロック番号である。先頭ブロックＩＤで指定される１つ以上のブロックにテーブルＩＤ２２１で識別される表が格納されていることを意味する。本実施例では、トランザクション処理部１０７は先頭ブロックからどのように表が格納されているかを知っており、先頭ブロックＩＤから該表のすべてのブロックを参照できる。
【００５２】
インデックステーブル２３０は、データベースのインデックスを管理する。それぞれの行が１つのインデックスに対応する。各インデックスは一意な名前であるインデックスＩＤが与えられており、それがインデックスＩＤ２３１に格納されている。先頭ブロックＩＤ２２２はインデックスが格納されている先頭のブロック番号である。先頭ブロックＩＤで指定される１つ以上のブロックにインデックスＩＤ２３１で識別されるインデックスが格納されていることを意味する。本実施例では、トランザクション処理部１０７は先頭ブロックからどのようにインデックスが格納されているかを知っており、先頭ブロックＩＤから該インデックスのすべてのブロックを参照できる。
【００５３】
コミットログ２４０は、現在活動中のトランザクションの活動記録である。それぞれの行が、トランザクションが行った１つの操作に対応する。トランザクションＩＤ２４１は操作を行ったトランザクションのＩＤ、ブロックＩＤ２４２とレコードＩＤ２４３は対で操作対象のレコードを指定する。カラムＩＤ２４４は必要に応じてカラム（インデックスの場合は何個目のインデックスエントリか）を指定する。操作２４５は行った操作であり、更新、削除、挿入が典型的な操作である。前データ２４６と後データ２４７はそれぞれ更新前後のデータである。
【００５４】
ロックテーブル２００、トランザクションテーブル２１０、ＤＢテーブルテーブル２２０、インデックステーブル２３０は、ＴＳＣ１０５上のメモリに保持されても、ディスク上に保持されても差し支えない。コミットログ２４０は、ディスクの特定部分に保持されるか、ＴＳＣ１０５のメモリであって、電源バックアップ、二重化等、ステーブルストレージとしての性質を満たしたメモリに保持される。
【００５５】
図３を用いて、トランザクショナルストレージの内部データ構造であるテーブルディクショナリ、インデックスディクショナリ、コミット後キャッシュ、コミット前キャッシュの構成について説明する。
【００５６】
テーブルディクショナリ３００は、データベースの表の定義を格納する。テーブルＩＤ３０１は表の一意な名前、カラムＩＤ３０２は表の中でのカラムの一意な名前、データ型３０３はカラムのデータ型である。
【００５７】
同様に、インデックスディクショナリ３１０は、データベースのインデックスの定義を格納する。インデックスＩＤ３１１はインデックスの一意な名前、カラムＩＤ３１２は表の中でのカラムの一意な名前、データ型３１３はカラムのデータ型である。

【００５８】
コミット前キャッシュ３２０は、まだコミットされていないトランザクションによって変更されたデータベースの行である。トランザクションＩＤ３２１はトランザクションＩＤ、ブロックＩＤ３２２とレコードＩＤ３２３は対で１つのレコードを指定する。データ３２４は更新後のデータである。
【００５９】
同様にコミット後キャッシュ３３０は、コミットされたがステーブルデータベースに反映されていないトランザクションによって変更されたデータベースの行である。ブロックＩＤ３３１とレコードＩＤ３３２は対で１つのレコードを指定する。データ３３３は更新後のデータである。
【００６０】
コミット後キャッシュ３３０はディスクの特定部分に保持されるか、ＴＳＣ１０５のメモリであって、電源バックアップ、二重化等、ステーブルストレージとしての性質を満たしたメモリに保持される。
【００６１】
トランザクショナルストレージの機能
次に、トランザクショナルストレージの機能と対応するトランザクショナルストレージインタフェース１１３について説明する。
【００６２】
トランザクション制御機能：
トランザクションＩＤ　ＢｅｇｉｎＴｒａｎｓａｃｔｉｏｎ（）；
ｂｏｏｌ　ＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎ（トランザクションＩＤ）；　ｂｏｏｌ　ＰｒｅｐａｒｅＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎ（トランザクションＩＤ）；
ｖｏｉｄ　ＡｂｏｒｔＴｒａｎｓａｃｔｉｏｎ（トランザクションＩＤ）；ＢｅｇｉｎＴｒａｎｓａｃｔｉｏｎ（）、ＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎ（）、ＰｒｅｐａｒｅＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎ（）、ＡｂｏｒｔＴｒａｎｓａｃｔｉｏｎ（）は、トランザクションの制御機能である。コンピュータ１０２はＴＳ１０４に対してＢｅｇｉｎＴｒａｎｓａｃｔｉｏｎ（）を発行し、新たなトランザクションの開始を宣言する。返り値はトランザクションＩＤである。ＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎ（）は、トランザクションを正常終了させるよう試みる。返り値は、トランザクションが正常終了したか否かである。ＰｒｅｐａｒｅＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎ（）は、複数のＴＳ１０４が１つのトランザクションをコミットさせようとする際に使う、ツーフェーズコミットの第１フェーズである。ＰｒｅｐａｒｅＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎ（）でツーフェーズコミットを開始した場合、第２フェーズの開始はＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎ（）で行う。ＡｂｏｒｔＴｒａｎｓａｃｔｉｏｎ（）はトランザクションをアボートさせる。
【００６３】
レコードおよびカラム操作機能：
ｒｅｃｏｒｄ　ＲｅａｄＲｅｃｏｒｄ（トランザクションＩＤ、レコード指定・、ロックモード・）；
ｖｏｉｄ　ＷｒｉｔｅＲｅｃｏｒｄ（トランザクションＩＤ、レコード指定、新レコード・、ロックモード・）；
ｒｅｃｏｒｄ　ＵｐｄａｔｅＲｅｃｏｒｄ（トランザクションＩＤ、レコード指定、新レコード・、ロックモード・）；
ｖｏｉｄ　ＩｎｓｅｒｔＲｅｃｏｒｄ（トランザクションＩＤ、レコード指定、新レコード・、ロックモード・）；
ｒｅｃｏｒｄ　ＤｅｌｅｔｅＲｅｃｏｒｄ（トランザクションＩＤ、レコード指定）；
Ｃｏｌｕｍｎ　ＲｅａｄＣｏｌｕｍｎ（トランザクションＩＤ、カラム指定・、ロックモード・）；
ｖｏｉｄ　ＷｒｉｔｅＣｏｌｕｍｎ（トランザクションＩＤ、カラム指定、新カラム・、

ロックモード・）；
Ｃｏｌｕｍｎ ＵｐｄａｔｅＣｏｌｕｍｎ（トランザクションＩＤ、カラム指定、新カラム・、ロックモード・）；
ＲｅａｄＲｅｃｏｒｄ（）、ＷｒｉｔｅＲｅｃｏｒｄ（）、ＵｐｄａｔｅＲｅｃｏｒｄ（）、ＩｎｓｅｒｔＲｅｃｏｒｄ（）、ＤｅｌｅｔｅＲｅｃｏｒｄ（）、ＲｅａｄＣｏｌｕｍｎ（）、ＷｒｉｔｅＣｏｌｕｍｎ（）、ＵｐｄａｔｅＣｏｌｕｍｎ（）は、レコード単位入出力およびカラム単位入出力の機能である。

【００６４】

ＲｅａｄＲｅｃｏｒｄ（）はトランザクションＩＤと後述するレコード指定とを指定し、１つのレコードを返り値として返す。なお、この機能の拡張として、レコードの選択条件を指定して１つ以上のレコードを返す機能を実現することは容易である。後述する他のインタフェースも同様である。ＷｒｉｔｅＲｅｃｏｒｄ（）と ＵｐｄａｔｅＲｅｃｏｒｄ（）はトランザクションＩＤ、レコード指定、新レコードのデータを指定し、レコードの更新を行う。ＵｐｄａｔｅＲｅｃｏｒｄ（）は旧レコードの値を返り値として返す。ＩｎｓｅｒｔＲｅｃｏｒｄ（）は、トランザクションＩＤ、レコード指定、新レコードのデータを指定し、レコードの挿入を行う。ＤｅｌｅｔｅＲｅｃｏｒｄ（）は、トランザクションＩＤとレコード指定を指定し、レコードの削除を行う。いずれの場合も「ロックモード」は各操作の正常終了時に、レコードをどのようなモードでロックするかを指定する。

【００６５】

ＲｅａｄＣｏｌｕｍｎ（）、ＷｒｉｔｅＣｏｌｕｍｎ（）、ＵｐｄａｔｅＣｏｌｕｍｎ（）は、カラム単位の操作であり、ＲｅａｄＲｅｃｏｒｄ（）、ＷｒｉｔｅＲｅｃｏｒｄ（）、ＵｐｄａｔｅＲｅｃｏｒｄ（）にそれぞれ対応する。レコード指定のかわりにカラム指定を用いる。その他の動作は、ＲｅａｄＣｏｌｕｍｎ（）、ＷｒｉｔｅＣｏｌｕｍｎ（）、ＵｐｄａｔｅＣｏｌｕｍｎ（）と同様である。

【００６６】

インデックス操作機能：
ｖｏｉｄ ＩｎｓｅｒｔＩｎｄｅｘ（トランザクションＩＤ、インデックスＩＤ、インデックス指定）；
ｖｏｉｄ ＵｐｄａｔｅＩｎｄｅｘ（トランザクションＩＤ、インデックスＩＤ、インデックス指定、値）；
ｖｏｉｄ ＤｅｌｅｔｅＩｎｄｅｘ（トランザクションＩＤ、インデックスＩＤ、インデックス指定）；
ＩｎｓｅｒｔＩｎｄｅｘ（）、ＵｐｄａｔｅＩｎｄｅｘ（）、ＤｅｌｅｔｅＩｎｄｅｘ（）はインデックスを操作する機能である。トランザクションＩＤ、インデックスＩＤ、および後述するインデックス指定を指定し、インデックスの一部分の挿入、更新、削除をそれぞれ行う。

【００６７】

ロックつきブロック単位入出力機能およびロック機能：
Ｂｌｏｃｋ ＲｅａｄＢｌｏｃｋＷｉｔｈＬｏｃｋ（ブロック指定、ロックモード）；
ｖｏｉｄ ＷｒｉｔｅＢｌｏｃｋＷｉｔｈＬｏｃｋ（ブロック指定、ロックモード、ブロック）；
Ｂｌｏｃｋ ＬｏｃｋＲｅｃｏｒｄ（レコード指定、ロックモード）；
ｖｏｉｄ ＬｏｃｋＢｌｏｃｋ（ブロック指定、ロックモード）；
ＲｅａｄＢｌｏｃｋＷｉｔｈＬｏｃｋ（）と ＷｒｉｔｅＢＬｏｃｋＷｉｔｈＬｏｃｋ（）は、ロックを伴ってブロック単位入出力を行う機能である。後述するブロック指定によってブロックを指定し、ロックモード（ｒｅａｄ、ｗｒｉｔｅ、ｉｎｔｅｎｔｉｏｎ等）でロックを指定する。また、ＬｏｃｋＲｅｃｏｒｄ（）と ＬｏｃｋＢｌｏｃｋ（）は、ロックを単独で操作する機能である。上述のＲｅａｄＲｅｃｏｒｄ（）、ＷｒｉｔｅＲｅｃｏｒｄ（）等で自動的にロックは取得されるが、得にロックの状態を変更したいときにＬｏｃｋＲｅｃｏｒｄ（）や ＬｏｃｋＢｌｏｃｋ（）の機能を用いる。

【００６８】
ディクショナリ情報入出力：
ｖｏｉｄ　ＧｅｔＴａｂｌｅＤｉｃｔｉｏｎａｒｙ（テーブルＩＤ、ディクショナリ）；
　ｖｏｉｄ　ＧｅｔＩｎｄｅｘＤｉｃｔｉｏｎａｒｙ（インデックスＩＤ、ディクショナリ）；
ＧｅｔＴａｂｌｅＤｉｃｔｉｏｎａｒｙ（）は、テーブルＩＤに対応するテーブルのディクショナリ情報（何個目のカラムがどのような型のデータか）をコンピュータから受け取りテーブルディクショナリ３００に格納する。同様にＧｅｔＩｎｄｅｘＤｉｃｔｉｏｎａｒｙ（）は、テーブルＩＤに対応するテーブルのディクショナリ情報（何個目のカラムがどのような型のデータか）をコンピュータから受け取り、インデックスディクショナリ３１０に格納する。
【００６９】
次に、レコード、ブロック、カラム、インデックスの指定方法について説明する。
【００７０】
レコード指定：
レコード指定には、「ブロックＩＤ、レコードＩＤ」によるレコード直接指定、「テーブルＩＤ、カラムＩＤ=値、カラムＩＤ=値、…」によるテーブルスキャン指定、「インデックスＩＤ、カラムＩＤ=値、カラムＩＤ=値、…」によるインデックススキャン指定、「インデックスのブロックＩＤ、カラムＩＤ=値、カラムＩＤ=値、…」によるインデックス部分スキャン指定、の４種類がある。ここで、「カラムＩＤ=値、カラムＩＤ=値、…」の部分はレコード絞り込みのための条件である。本実施例ではカラム毎の等号条件を用いてレコードの絞り込みを行っているが、本発明はこれに限定されるものではなく、不等号条件、ＮＵＬＬ条件、カラムとカラムの二項条件等でも差し支えない。以下の記述も同様である。
【００７１】
レコード直接指定はブロックＩＤとレコードＩＤによって１つのレコードを指定する。テーブルスキャン指定は、テーブルＩＤでテーブルを指定し、１つ以上の「カラムＩＤ=値」によってレコードの絞り込みを行い、１つのレコードを指定する（条件にあう最初のレコードが指定されたものとみなす）。この際、テーブルに含まれるブロックの解釈は、テーブルディクショナリ３００に格納されている情報を用いて行う。
【００７２】
インデックススキャン指定では、インデックスＩＤで検索すべきインデックスを指定し、１つ以上の「カラムＩＤ=値」によって与えられたキー値によってインデックスを検索する。また、インデックス部分スキャン指定では、インデックスの中間部分（例えばインデックスがＢツリーで実現されている場合、ルートノード以外のノードを格納したブロック）をブロックＩＤで指定し、そこから１つ以上の「カラムＩＤ=値」によってインデックスを検索する。インデックスに含まれるブロックの解釈は、インデックスディクショナリ３１０に格納されている情報を用いて行う。
【００７３】
先に述べた通り、レコード直接指定以外の指定法では、複数レコードを指定することが自然にできる。
【００７４】
ブロック指定：
ブロック指定には、「ブロックＩＤ」によるブロック直接指定、「テーブルＩＤ、カラムＩＤ=値、カラムＩＤ=値、…」によるテーブルスキャン指定、「インデックスＩＤ、カラムＩＤ=値、カラムＩＤ=値、…」によるインデックススキャン指定、「インデックスのブロックＩＤ、カラムＩＤ=値、カラムＩＤ=値、…」によるインデックス部分スキャン指定、の４種類がある。ブロック指定はレコード指定と同じであるが、指定に該当するレコードを含むブロックが指定されたものとみなす。
【００７５】

カラム指定：
カラム指定は「レコード指定、カラムＩＤ」で行う。レコード指定は、上述のレコード指定のうち任意の１つを用いる。
【００７６】
インデックス指定：
「インデックスＩＤ、カラムＩＤ=値、カラムＩＤ=値、…」によるインデックススキャン指定、「インデックスのブロックＩＤ、カラムＩＤ=値、カラムＩＤ=値、…」によるインデックス部分スキャン指定、の２種類がある。
【００７７】
以上がレコード、ブロック、カラム、インデックスの指定方法である。なお、これらの指定に対するレコード、ブロック、カラム、インデックスに対するアクセスの実現は、データベース管理システムと同様の方法をトランザクション処理部１０７が行う。この方法は公知の技術であるため、ここでは特に改めて説明しない。
【００７８】
以下、最も典型的な処理であるトランザクショナルストレージ中でのＲｅａｄＲｅｃｏｒｄ（）、ＷｒｉｔｅＲｅｃｏｒｄ（）、ＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎ（）、ＡｂｏｒｔＴｒａｎｓａｃｔｉｏｎ（）の処理の流れ、およびコンピュータ側からトランザクショナルストレージを呼び出す処理について、流れ図を用いて説明する。
【００７９】
図４を用いて、ＲｅａｄＲｅｃｏｒｄ処理の流れについて説明する。
【００８０】
ＴＳ１０４がＲｅａｄＲｅｃｏｒｄ（トランザクションＩＤ、レコード指定）の要求をコンピュータ１０２から受け取ると、該要求はＴＳＣ１０５のネットワーク制御部１０６が受け取り、トランザクション処理部１０７へ渡される。トランザクション処理部１０７では、レコード指定を解釈し、ブロックＩＤとレコードＩＤを得る（ステップ４０１）。次に、トランザクションＩＤとブロックＩＤとレコードＩＤを用いて、該トランザクションＩＤ・ブロックＩＤ・レコードＩＤの組に合致するエントリがコミット前キャッシュ３２０に存在するかを判定する（ステップ４０２）。ステップ４０２の結果が真（Ｙ）の場合、ステップ４０３へ、偽（Ｎ）の場合ステップ４０４へ制御を移す。
【００８１】
ステップ４０３では、「該ブロックＩＤ、該レコードＩＤ」を用いてロックテーブル２００を検索してロック情報２０３を得、ロック情報２０３のロックモードと該要求で指定されたロックモードとを比較し、より強い方（ｒｅａｄよりｗｒｉｔｅが強い。ｉｎｔｅｎｔｉｏｎよりｒｅａｄが強い等、よく知られたロックの強さによる）を該ロック情報２０３に格納する。指定がなければｒｅａｄ　ｌｏｃｋを格納する。そして、データ３２４に格納されているレコードを返答し、正常終了する（場合によってはすでにそのレコードが削除されたことを意味する「削除」が返る）。
【００８２】
ステップ４０４では、「該ブロックＩＤ、該レコードＩＤ」を用いてロックテーブル２００を検索し、（１）対応するロック情報２０３がないか、または（２）ロック情報２０３に保持されているトランザクションＩＤが上記トランザクションＩＤであるか、または（３）他のトランザクションが保持しているロックがｒｅａｄ　ｌｏｃｋであるか、のいずれかの条件を満たすか否かを判定する。判定が真（Ｙ）なら、ステップ４０５に制御を移す。一方判定が偽（Ｎ）なら、他のトランザクションがすでにアクセス対象のレコードのロックを取得していることになるため、該要求は異常終了する。
【００８３】
ステップ４０５では該ブロックＩＤ・レコードＩＤの組に合致するエントリがコミット後キャッシュ３３０に存在するかを判定する。存在すれば（判定Ｙ）ステップ４０８、存在しなければ（判定Ｎ）ステップ４０６へ制御を移す。
【００８４】

ステップ４０６では、ディスク制御部１１０に制御がわたり、ディスク１１１から該ブロックＩＤのブロックが取り出される。続くステップ４０７では、取り出したブロック中を検索し、該レコードＩＤに合致するレコードを得、ステップ４０８に制御を移す。

【００８５】

ステップ４０８では、「該ブロックＩＤ、該レコードＩＤ」を用いてロックテーブル２００を検索してロック情報２０３を得、ロック情報２０３のロックモード（もしあれば）と該要求で指定されたロックモードとを比較し、より強い方（ｒｅａｄよりｗｒｉｔｅが強い。ｉｎｔｅｎｔｉｏｎよりｒｅａｄが強い等、よく知られたロックの強さによる）を該ロック情報２０３に格納する。指定がなければｒｅａｄ　ｌｏｃｋを格納する。すでにｒｅａｄ　ｌｏｃｋが他のトランザクションによって取得されている場合、新たなエントリを作成してロックテーブル２００に格納する。

【００８６】

ステップ４０９では、「該トランザクションＩＤ、該ブロックＩＤ、該レコードＩＤ、結果のレコード」をコミット前キャッシュ３２０に登録する。コミット前キャッシュ３２０があふれた場合には、ステーブルデータベース（すなわちディスク１１１）に一部または全部を書き戻す。そして結果のレコードを返答して、正常終了する。

【００８７】

以上がＲｅａｄＲｅｃｏｒｄ処理の流れである。

【００８８】

図５を用いて、ＷｒｉｔｅＲｅｃｏｒｄ処理の流れについて説明する。

【００８９】

ＴＳ１０４がＷｒｉｔｅＲｅｃｏｒｄ（トランザクションＩＤ、レコード指定、新レコード）の要求をコンピュータ１０２から受け取ると、該要求はＴＳＣ１０５のネットワーク制御部１０６が受け取り、トランザクション処理部１０７へ渡される。トランザクション処理部１０７では、レコード指定を解釈し、ブロックＩＤとレコードＩＤを得る（ステップ５０１）。次に、トランザクションＩＤとブロックＩＤとレコードＩＤを用いて、該トランザクションＩＤ・ブロックＩＤ・レコードＩＤの組に合致するエントリがコミット前キャッシュ３２０に存在するかを判定する（ステップ５０２）。ステップ５０２の結果が真（Ｙ）の場合、ステップ５０３へ、偽（Ｎ）の場合ステップ５０４へ制御を移す。

【００９０】

ステップ５０３では、「該ブロックＩＤ、該レコードＩＤ」を用いてロックテーブル２００を検索してロック情報２０３を得、ロック情報２０３のロックモードと該要求で指定されたロックモードとを比較し、より強い方（ｒｅａｄよりｗｒｉｔｅが強い。ｉｎｔｅｎｔｉｏｎよりｒｅａｄが強い等、よく知られたロックの強さによる）を該ロック情報２０３に格納する。指定がなければｗｒｉｔｅ　ｌｏｃｋを格納する。そして、データ３２４に格納されているレコードを旧レコードとして以降の処理を続ける。

【００９１】

ステップ５０４では、「該ブロックＩＤ、該レコードＩＤ」を用いてロックテーブル２００を検索し、（１）対応するロック情報２０３がないか、または（２）ロック情報２０３に保持されているトランザクションＩＤが上記トランザクションＩＤであるか、を判定する。判定が真（Ｙ）なら、ステップ５０５に制御を移す。一方判定が偽（Ｎ）なら、他のトランザクションがすでにアクセス対象のレコードのロックを取得していることになるため、該要求は異常終了する。

【００９２】

ステップ５０５では該ブロックＩＤ・レコードＩＤの組に合致するエントリがコミット後キャッシュ３３０に存在するかを判定する。存在すれば（判定Ｙ）データ３３３を旧レコードとしてステップ５０８へ制御を移し、存在しなければ（判定Ｎ）ステップ５０６へ制御を移す。

【００９３】

ステップ５０６では、ディスク制御部１１０に制御がわたり、ディスク１１１から該ブロ

ックＩＤのブロックが取り出される。続くステップ５０７では、取り出したブロック中を検索し、該レコードＩＤに合致するレコードを得てこれを旧レコードとし、ステップ５０８に制御を移す。

【００９４】

ステップ５０８では、「該ブロックＩＤ、該レコードＩＤ」を用いてロックテーブル２００を検索してロック情報２０３を得、ロック情報２０３のロックモードと該要求で指定されたロックモードとを比較し、より強い方（ｒｅａｄよりｗｒｉｔｅが強い。ｉｎｔｅｎｔｉｏｎよりｒｅａｄが強い等、よく知られたロックの強さによる）を該ロック情報２０３に格納する。指定がなければｗｒｉｔｅ　ｌｏｃｋを格納する。

【００９５】

ステップ５０９では、「該トランザクションＩＤ、該ブロックＩＤ、該レコードＩＤ、新レコード」をコミット前キャッシュ３２０に登録する。コミット前キャッシュがあふれた場合には、ステーブルデータベースに一部または全部を書き戻す。

【００９６】

ステップ５１０では、コミットログ２４０に本操作での変更を記録する。すなわち、「該トランザクションＩＤ、該ブロックＩＤ、該レコードＩＤ、＿、更新、旧レコード、新レコード」をコミット前キャッシュ３２０に追記する。そして正常終了する。

【００９７】

以上がＷｒｉｔｅＲｅｃｏｒｄ処理の流れである。

【００９８】

図６を用いて、Ｃｏｍｍｉｔ処理の流れについて説明する。

【００９９】

ＴＳ１０４がＣｏｍｍｉｔＴｒａｎｓａｃｔｉｏｎ（トランザクションＩＤ）の要求をコンピュータ１０２から受け取ると、該要求はＴＳＣ１０５のネットワーク制御部１０６が受け取り、トランザクション処理部１０７へ渡される。トランザクション処理部１０７では、以下の処理を行う。

【０１００】

ステップ６０１で、コミット前キャッシュ３２０をスキャンし、トランザクションＩＤ３２１が該要求のトランザクションＩＤに等しいエントリのうち、データ３２４が「削除」でないエントリをコミット後キャッシュ３３０に移動する。この際「ブロックＩＤ３２２、レコードＩＤ３２３、データ３２４」を「ブロックＩＤ３３１、レコードＩＤ３３２、データ３３３」とする。この際コミット後キャッシュ３３０があふれた場合には、コミット後キャッシュ３３０の一部または全部をステーブルデータベースに移動する。

【０１０１】

また、トランザクションＩＤ３２１が該要求のトランザクションＩＤに等しいエントリのうち、データ３２４が「削除」であるエントリはディスク制御部１１０経由で対応するレコードの削除を行う。

【０１０２】

ステップ６０２で、トランザクションテーブル２１０をスキャンしてロックテーブル２００の解放を行う。すなわち、トランザクションテーブル２１０中でトランザクションＩＤ２１１が該要求のトランザクションＩＤに等しいエントリを検索し、条件に合致するエントリのブロックＩＤ２１２とレコードＩＤ２１３の対それぞれについて、ロックテーブル２００を検索する。そして、条件に合致するロックテーブル２００のエントリのそれぞれについて、ロック情報２０３に該要求のトランザクションＩＤが格納されていれば、該エントリを削除する。

【０１０３】

ステップ６０３で、トランザクションテーブル２１０のトランザクションＩＤ２１１が該要求のトランザクションＩＤを用いて検索し、対応するエントリを削除する。

【０１０４】

ステップ６０４で、コミットログ２４０中でトランザクションＩＤ２４１が該要求のトラ

ンザクションＩＤと等しいエントリを検索し、条件に合致するエントリを削除する。
【０１０５】
以上がＣｏｍｍｉｔ処理の流れである。
【０１０６】
図７を用いて、Ｃｏｍｍｉｔ処理の流れについて説明する。
【０１０７】
ＴＳ１０４がＡｂｏｒｔＴｒａｎｓａｃｔｉｏｎ（トランザクションＩＤ）の要求をコンピュータ１０２から受け取ると、該要求はＴＳＣ１０５のネットワーク制御部１０６が受け取り、トランザクション処理部１０７へ渡される。トランザクション処理部１０７では、以下の処理を行う。
【０１０８】
ステップ７０１で、コミットログ２４０中でトランザクションＩＤ２４１が該要求のトランザクションＩＤと等しいエントリを後方から前方へ検索し、条件に合致するエントリのそれぞれについて、コミット後キャッシュ３３０またはステーブルデータベース中で「ブロックＩＤ２４２、レコードＩＤ２４３」・、カラムＩＤ２４４・」で指定されるレコードを得て、そのレコードの値を後データ２４７から前データ２４６へ戻す。
【０１０９】
ステップ７０２で、トランザクションテーブル２１０をスキャンしてロックテーブル２００の解放を行う。すなわち、トランザクションテーブル２１０中でトランザクションＩＤ２１１が該要求のトランザクションＩＤに等しいエントリを検索し、条件に合致するエントリのブロックＩＤ２１２とレコードＩＤ２１３の対それぞれについて、ロックテーブル２００を検索する。そして、条件に合致するロックテーブル２００のエントリのそれぞれについて、ロック情報２０３に該要求のトランザクションＩＤが格納されていれば、該エントリを削除する。
【０１１０】
ステップ７０３で、トランザクションテーブル２１０のトランザクションＩＤ２１１が該要求のトランザクションＩＤを用いて検索し、対応するエントリを削除する。
【０１１１】
ステップ７０４で、コミットログ２４０中でトランザクションＩＤ２４１が該要求のトランザクションＩＤと等しいエントリを検索し、条件に合致するエントリを削除する。
【０１１２】
以上がＡｂｏｒｔ処理の流れである。
【０１１３】
図８を用いて、ＲｅａｄＲｅｃｏｒｄ処理の要求を発行する際のコンピュータ１０２側の処理の流れについて説明する。
【０１１４】
ステップ８０１で、アクセスメソッドを決定する。すなわち、アクセスしようとするレコードをインデックスを用いてアクセスするか、インデックスを用いずにアクセスするか、また、インデックスを用いる場合にはどのインデックスを用いるかを決定する。この処理はデータベース管理システムでよく用いられる処理である。
【０１１５】
ステップ８０２で、インデックスを使用するか否かによって、ステップ８０３（インデックスを用いない場合）またはステップ８０４（インデックスを用いる場合）に制御を移す。
【０１１６】
ステップ８０３では、レコード指定をテーブルスキャンとしてＲｅａｄＲｅｃｏｒｄ（）処理をＴＳ１０４へ要求する。
【０１１７】
ステップ８０４では、コンピュータ１０２のメモリ上にインデックスのすべてが存在するか否かを判定する。存在すれば（Ｙ）ステップ８０５へ、存在しなければ（Ｎ）ステップ

８０６へ制御を移す。
【０１１８】
ステップ８０５では、インデックスをアクセスしてブロックＩＤとレコードＩＤを得、レコード指定をレコード直接指定としてＲｅａｄＲｅｃｏｒｄ（）処理をＴＳ１０４へ要求する。
【０１１９】
ステップ８０６では、コンピュータ１０２のメモリ上にインデックスの一部が存在するか否かを判定する。存在すれば（Ｙ）ステップ８０７へ、存在しなければ（Ｎ）ステップ８０８へ制御を移す。
【０１２０】
ステップ８０７では、コンピュータ１０２のメモリ上に存在するインデックスをアクセスしてインデックスの中間ノードを得、レコード指定をインデックス部分スキャン指定としてＲｅａｄＲｅｃｏｒｄ（）処理をＴＳ１０４へ要求する。
【０１２１】
ステップ８０８では、使用するインデックスのインデックスＩＤを用いて、レコード指定をインデックススキャン指定としてＲｅａｄＲｅｃｏｒｄ（）処理をＴＳ１０４へ要求する。
【０１２２】
最後にステップ８０９で、ＴＳ１０４から結果を得る。以上がＲｅａｄＲｅｃｏｒｄ処理の要求を発行する際のコンピュータ１０２側の処理の流れである。この流れは、ＷｒｉｔｅＲｅｃｏｒｄ、ＩｎｓｅｒｔＲｅｃｏｒｄ、ＤｅｌｅｔｅＲｅｃｏｒｄ、また、カラム単位の操作でも同様である。
【０１２３】
【発明の効果】
以上述べた本発明の、トランザクションを意識した機能およびインタフェースを持つストレージ「トランザクショナルストレージ」とそれを用いたコンピュータシステムによって、従来のトランザクション処理に伴う以下の課題が解決される。
【０１２４】
（１）トランザクショナルストレージのレコード単位の入出力機能およびインタフェースにより、ブロック入出力による無駄なデータの入出力が削減される。
【０１２５】
（２）トランザクショナルストレージの、レコードを更新する機能およびインタフェースにより、更新に伴う無駄な入出力が削減される。更新時には、必要に応じて更新前データを返すことにより、ログを効率的に作成できる。
【０１２６】
（３）トランザクショナルストレージが（ａ）ブロック番号とレコード番号によるレコード指定インタフェース、（ｂ）テーブルと条件によるレコード指定インタフェース、（ｃ）インデックスと条件によるレコード指定インタフェース、（ｄ）インデックス中間ノードと条件によるレコード指定インタフェース、の４種類のレコード指定インタフェースを備えることにより、効率的なレコードアクセスが実現される。
【０１２７】
（４）トランザクショナルストレージがコンピュータからディクショナリ情報を入力するインタフェースを備えることにより、任意のストレージでディクショナリ情報が利用可能となる。
【０１２８】
（５）トランザクショナルストレージがコミットのインタフェースとアボートのインタフェースを備えることにより、トランザクションの実現（コミットやアボート）による無駄なデータの入出力が削減される。
【０１２９】
（６）トランザクショナルストレージが、コミット前キャッシュとコミット後キャッシュ

を備え、書き込んだデータを取り消す方法を提供することにより、トランザクションがアボートした際の無駄な入出力が削減される。

【０１３０】

（７）トランザクショナルストレージがレコードおよびブロックのロック（排他制御）インタフェースを備えることにより、同一のデータをレコード単位でもブロック単位でも処理でき、複数のコンピュータからの共有が可能になる。

【０１３１】

以上により、トランザクション処理に伴ってコンピュータのプロセッサおよびストレージに生じる高い負荷を削減し、大規模なデータベースシステムおよびトランザクション処理システムの実現が可能となる。

【図面の簡単な説明】

【図１】本実施例の全体構成を示すブロック図。

【図２】トランザクショナルストレージの内部データの構成図。

【図３】トランザクショナルストレージのディクショナリとキャッシュの構成図。

【図４】ＲｅａｄＲｅｃｏｒｄ処理の流れ図。

【図５】ＷｒｉｔｅＲｅｃｏｒｄ処理の流れ図。

【図６】Ｃｏｍｍｉｔ処理の流れ図。

【図７】Ａｂｏｒｔ処理の流れ図。

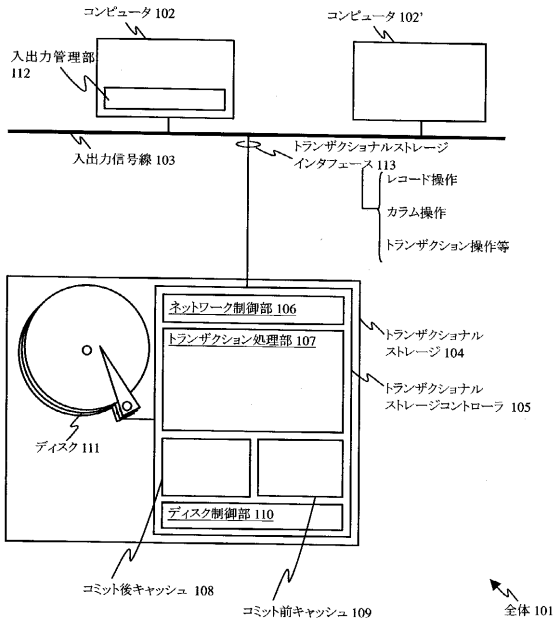
【図８】コンピュータ側処理の流れ図。

【符号の説明】

１０１：全体
１０２：コンピュータ
１０３：入出力信号線
１０４：ＴＳ
１０５：ＴＳＣ
１０６：ネットワーク制御部
１０７：トランザクション処理部
１０８：コミット後キャッシュ
１０９：コミット前キャッシュ
１１０：ディスク制御部
１１１：ディスク
１１２：入出力処理部
１１３：トランザクショナルストレージインタフェース。

【図１】

図１



【図２】

図２

ロックテーブル 200

ブロック ID 201	レコード ID 202	ロック情報 203
101	30	Tr1024, read
101	-	Tr1024, read-intention
101	31	Tr1024, read
102	45	Tr1022, write
…	…	…

トランザクションテーブル 210

トランザクション ID 211	ブロック ID 212	レコード ID 212
Tr1024	101	30
Tr1024	101	31
Tr1022	102	45
	…	…
…	…	…

DB テーブルテーブル 220

テーブル ID 221	先頭ブロック ID 222
t01	101
t02	151
…	…

インデックステーブル 230

インデックス ID 231	先頭ブロック ID 232
i01	201
i02	202
…	…

コミットログ 240

トランザクション ID 241	ブロック ID 242	レコード ID 243	カラム ID 244	操作 245	前データ 246	後データ 247
Tr1024	101	30	c2	更新	'hello'	'hello'
	101	31	---	削除	"20, 'hello', 10.5"	---
Tr1022	102	45	---	挿入	---	"433, 'suzuki'"
…	…	…	…	…	…	…

【図３】

図３

テーブルディクショナリ 300

テーブル ID 301	カラム ID 302	データ型 303
t01	c1	int
	c2	char(20)
	c3	decimal(8,3)
t02	c1	shortint
	c2	varchar(20)
…	…	…

インデックスディクショナリ 310

インデックス ID 311	カラム ID 312	データ型 313
i01	c1	int
	c2	char(20)
t02	c1	shortint
…	…	…

コミット前キャッシュ 320

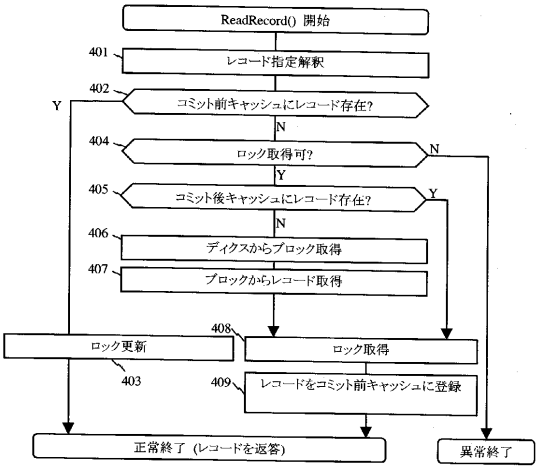
トランザクション ID 321	ブロック ID 322	レコード ID 323	データ 324
101	101	30	"20, 'hello', 10.5"
101	101	31	削除
102	102	45	"433, 'suzuki'"
…	…	…	…

コミット後キャッシュ 330

ブロック ID 331	レコード ID 332	データ 333
101	30	"20, 'hello', 10.5"
101	32	"21, 'world', 11.5"
102	45	"433, 'suzuki'"
…	…	…

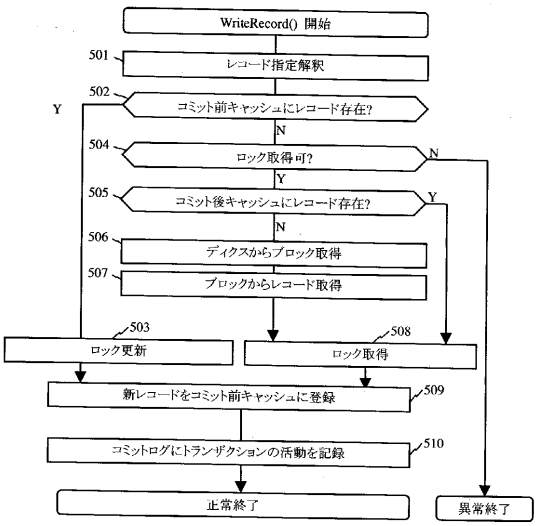
【図４】

図４



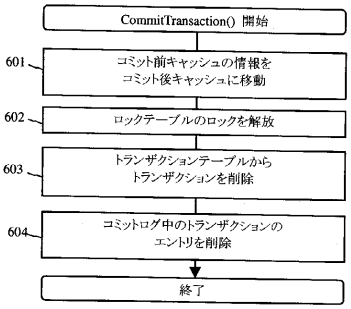
【図５】

図５



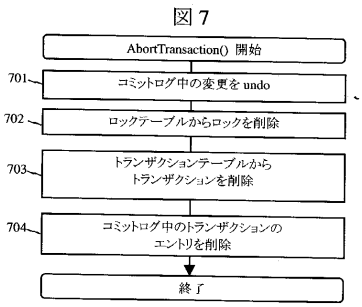
【図６】

図６



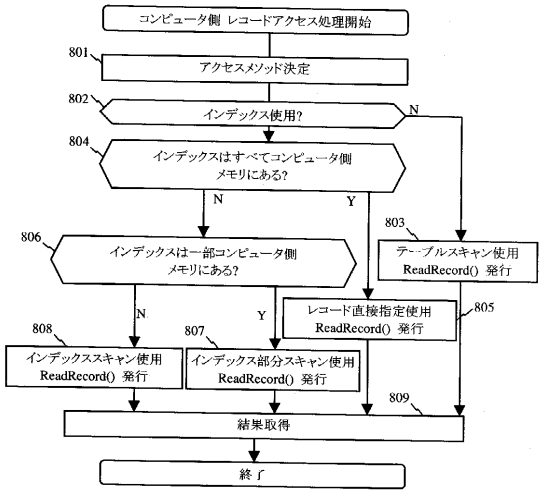
【図７】

図７



【図８】

図８



フロントページの続き

(72)発明者　牛嶋　一智
東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所中央研究所内
(72)発明者　森永　智之
東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所中央研究所内
(72)発明者　佐川　暢俊
東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所中央研究所内

審査官　田川　泰宏

(56)参考文献　特開平０２－２３８５６８（ＪＰ，Ａ）
特開平０６－１６１６６９（ＪＰ，Ａ）
特開平１０－０６９３５７（ＪＰ，Ａ）
機能ディスクシステム：関係データベース処理とその性能評価，電子情報通信学会論文誌，日本，１９９１年　8月２５日，Vol.J74-D-I,No.8，p.496-507
機能ディスク・システム第2版，コンピュータアーキテクチャ　シンポジウム論文集　昭和63年5月17日、18日，日本，１９８８年　5月１８日，p.37-46

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