(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5343399号
(P5343399)
(45)発行日　平成25年11月13日(2013.11.13)　(24)登録日　平成25年8月23日(2013.8.23)

(51)Int.Cl.　F I
G06F 12/00 (2006.01)　G06F 12/00 535R
G06F 12/00 518A
G06F 12/00 535C

請求項の数 7　(全 27 頁)

(21)出願番号　特願2008-134031 (P2008-134031)
(22)出願日　平成20年5月22日 (2008.5.22)
(65)公開番号　特開2009-282746 (P2009-282746A)
(43)公開日　平成21年12月3日 (2009.12.3)
審査請求日　平成23年1月18日 (2011.1.18)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100070150
弁理士　伊東　忠彦
(72)発明者　石本　大明
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　味沢　丞
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　伊藤　尚洋
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
最終頁に続く

(54)【発明の名称】管理プログラム、管理方法、及び管理装置

(57)【特許請求の範囲】
【請求項１】
データベースにおける第一のデータの参照の開始時に時間の前後関係を識別可能な第一の時系列データを前記第一のデータを識別する識別情報と対応付けて記憶部に記憶して、該第一のデータを参照する処理を開始し、
前記第一のデータの更新要求に応じ、前記第一のデータに対応する第二のデータを前記データベースに生成し、前記更新要求に基づいて該第二のデータの更新を開始し、
前記第二のデータの更新の完了時に第二の時系列データを前記第二のデータを識別する識別情報と対応付けて前記記憶部に記憶し、
前記参照する処理の終了時に前記第一の時系列データと、当該第一の時系列データに対応付けて前記記憶部に記憶された前記第一のデータの識別情報とを前記記憶部から削除し、
前記参照する処理の終了より後の前記更新の完了に応じて、前記第二の時系列データより前の時間を示す前記第一の時系列データが前記記憶部に記憶されているかを判定し、
前記第二の時系列データより前の時間を示す第一の時系列データが前記記憶部に記憶されていない場合は前記第一のデータを削除する
ことをコンピュータに実行させることを特徴とする管理プログラム。
【請求項２】
前記削除する処理は、前記第二の時系列データより前の時間を示す前記第一の時系列データであって、当該第二の時系列データと同一の前記識別情報に対応付けられている前記

第一の時系列データが前記記憶部に記憶されていない場合は前記第一のデータを削除する請求項１記載の管理プログラム。
【請求項３】
前記更新を開始する処理は、前記第一のデータ及び前記第二のデータに対してロックを適用する処理を含み、
前記管理プログラムは、
前記データの更新の完了に応じて、前記第二の時系列データより前の時間を示す前記第一の時系列データが前記記憶部に記憶されているかを判定した結果、当該第一の時系列データが前記記憶部に記憶されている場合は、前記ロックの内容を示す占有情報を前記記憶部に保持させ、
前記記憶部に保持されている占有情報に基づいて前記データベースの排他制御を行い、
前記参照する処理の終了に応じて前記第二の時系列データより前の時間を示す前記第一の時系列データが前記記憶部に記憶されていない場合は前記占有情報を削除する
ことを前記コンピュータに実行させる請求項１又は２記載の管理プログラム。
【請求項４】
前記第二の時系列データより前の時間を示す前記第一の時系列データに係る参照については前記第一のデータを参照対象とし、前記第二の時系列データより後の時間を示す前記第一の時系列データに係る参照については前記第二のデータを参照対象とする処理を前記コンピュータに実行させる請求項１乃至３いずれか一項記載の管理プログラム。
【請求項５】
前記第一及び第二の時系列データには、複数ノード環境における各ノードを識別するノード識別子が関連付けられ、
前記参照する処理を開始する処理は、他ノードに対して前記第一の時系列データの生成を要求し該要求に応じて該他ノードで生成された前記第一の時系列データを取得し、
前記更新を開始する処理は、前記他ノードからの更新要求に応じ、前記第一のデータに対応する第二のデータを前記データベースに生成し、該第二のデータを更新し、
前記他ノードにおける前記更新の完了に応じて、該他ノードにおいて生成された前記第二の時系列データを取得し、
前記第二の時系列データの取得に応じて、前記他ノードより取得された前記第二の時系列データより前の時間を示す前記他ノードより取得された前記第一の時系列データが前記記憶部に記憶されていない場合は前記第一のデータを削除する
処理を前記コンピュータに実行させる請求項１乃至４いずれか一項記載の管理プログラム。
【請求項６】
データベースにおける第一のデータの参照の開始時に時間の前後関係を識別可能な第一の時系列データを前記第一のデータを識別する識別情報と対応付けて記憶部に記憶して、該第一のデータを参照する処理を開始し、
前記第一のデータの更新要求に応じ、前記第一のデータに対応する第二のデータを前記データベースに生成し、前記更新要求に基づいて該第二のデータの更新を開始し、
前記第二のデータの更新の完了時に第二の時系列データを前記第二のデータを識別する識別情報と対応付けて前記記憶部に記憶し、
前記参照する処理の終了時に前記第一の時系列データと、当該第一の時系列データに対応付けて前記記憶部に記憶された前記第一のデータの識別情報とを前記記憶部から削除し、
前記参照する処理の終了より後の前記更新の完了に応じて、前記第二の時系列データより前の時間を示す前記第一の時系列データが前記記憶部に記憶されているかを判定し、
前記第二の時系列データより前の時間を示す第一の時系列データが前記記憶部に記憶されていない場合は前記第一のデータを削除する
ことをコンピュータが実行することを特徴とするデータ管理方法。
【請求項７】

データベースにおける第一のデータの参照の開始時に時間の前後関係を識別可能な第一の時系列データを前記第一のデータを識別する識別情報と対応付けて記憶部に記憶して、該第一のデータを参照する処理を開始する参照開始手段と、
前記第一のデータの更新要求に応じ、前記第一のデータに対応する第二のデータを前記データベースに生成し、前記更新要求に基づいて該第二のデータの更新を開始する更新手段と、
前記第二のデータの更新の完了時に第二の時系列データを前記第二のデータを識別する識別情報と対応付けて前記記憶部に記憶する完了手段と、
前記参照する処理の終了時に前記第一の時系列データと、当該第一の時系列データに対応付けて前記記憶部に記憶された前記第一のデータの識別情報とを前記記憶部から削除する参照終了手段と、
前記参照する処理の終了より後の前記更新の完了に応じて、前記第二の時系列データより前の時間を示す前記第一の時系列データが前記記憶部に記憶されているかを判定する判定手段と、
前記第二の時系列データより前の時間を示す第一の時系列データが前記記憶部に記憶されていない場合は前記第一のデータを削除する削除手段とを有する管理装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、管理プログラム、管理方法、及び管理装置に関する。
【背景技術】
【０００２】
従来、データベースでは、トランザクション（参照処理・更新処理）の多重実行において、同時実行性を確保しつつ、データの整合性や一貫性を保証するために、下記のような技術が利用されている。
（１）参照処理・更新処理をデータベース資源(行など)のロックにより制御する技術
（２）行をマルチバージョンで管理し、更新中にも更新前のデータを参照させる技術
（３）行を２バージョン以上で管理し、不要レコードの回収を行う技術
（１）では参照処理の際に対象資源に参照ロックを、更新処理の際に更新ロックをとることでデータの一貫性を保持している。参照ロックがとられている資源に対しては、参照ロックを取ろうとする処理のみ許可し、それ以外を排他する。更新ロックが取られている資源に対しては、どのロックの処理も排他する。このようにすることで、参照中や更新中の資源が更新されることを防ぎ、データの一貫性が保たれる。但し、参照処理と更新処理とをシリアライズ（排他待ち）することによるコンカレンシ（並列性）の向上に課題がある。また、データベース資源のロックによる結果として排他待ちが発生するため、業務構築において排他設計が困難となる。
【０００３】
そこで、（２）では各レコードに複数のバージョンを作成できるようにしている。すなわち、更新の際に新しいバージョンが作成される。参照処理は参照可能なバージョンを参照する（すなわち、参照処理と更新処理において対象資源のバージョンを変える）ことで、参照処理と更新処理とにおいて排他待ちが発生しないようにしている。しかし、過去バージョンのデータの破棄タイミングが不明であるため、過去バージョンを破棄できず、データ領域が肥大化してしまうという問題がある。
【０００４】
そこで、（３）ではレコードを２バージョン＋αで管理することで、更新と参照とを同時に実行できるようにすると共に、各バージョンのレコードに参照カウンタを付加している。具体的には、参照開始後の対象物理レコードのアクセス時に当該物理レコードの参照カウンタに１を加算し、参照の終了時に１を減算する。参照カウンタに基づいて参照されなくなった（参照カウンタが０になった）削除可能な過去バージョンのレコード（過去レコード）を判別及び削除し、データ領域の肥大化を抑止している。但し、バージョン数を

固定した場合、当該バージョン数以上のバージョンを作成しようとすると最古のバージョンが削除されてしまう。そうすると、最古のバージョンを用いる処理が実行できなくなるという不具合が発生する。そこで、更新トランザクションの完了時に、参照されている過去レコード（参照カウンタが１以上である過去レコード）がある場合は、遅延削除と呼ばれる偽の削除を行う。遅延削除とは、物理レコードに付加された削除ビットと呼ばれる偽の削除が行われたことを表すフラグを立てる処理をいう。遅延削除により、その過去レコードを参照する参照処理以外は当該過去レコードにアクセスできなくなる。全ての参照が完了すると（参照カウンタが０になると）、遅延削除が行われたレコードは削除される。

【特許文献１】特表２００７－５０１４６８号公報

【非特許文献１】“Postgresql Documentation Manuals PostgreSQL 7.1 Multi-VersionConcurrency Control”、[online]、[平成２０年４月２５日検索]、< http://www.postgresql.org/docs/7.1/static/mvcc.html>

【発明の開示】

【発明が解決しようとする課題】

【０００５】

しかしながら、（３）では、参照カウンタを物理レコードに付加し、参照の際に更新している。したがって、参照カウンタの更新のたびにデータベースへの書き戻しのためのＩ／Ｏが発生する可能性があり、参照処理が低速になるという問題がある。

【０００６】

また、ＤＢＭＳ（DataBase Management System）がダウンした場合、そのリカバリ時において物理レコードに書き込まれた参照カウンタなどに何らかの補正（例えば、全レコードの参照カウンタを０クリアする等）が必要とされる。補正が行われないと、リカバリ後に参照カウンタが０にならず、いつまでも不要なレコードが存在してしまう可能性があるからである。そのため、リカバリ時において、ＣＰＵ、メモリ、２次記憶装置等へのＩ／Ｏの発生といったコストがかかるという問題がある。

【０００７】

また、（３）では、更新完了後に参照される古いバージョンのレコードが不要となった際のレコードの回収、クリーンアップ等を、参照カウンタを０にした参照処理に行わせている。その結果、当該参照処理が低速となるという問題もある。

【０００８】

更に、（３）では、通常の２バージョンに加え、遅延削除により参照中の参照処理以外にアクセスできないバージョンが存在する。そのため、データベース資源の容量を設計する際に、この遅延削除のための容量についても考慮しなければならない。例えば、参照中の或るレコードに対し２回更新を行うと、２回目の更新中の際に、遅延削除による最も古いバージョンのレコード、参照可能なレコード、及び更新中のレコードの３バージョン分の容量が必要となる。遅延削除はいつ、どの程度行われるのかを事前に把握するのは難しいため、データベース資源の容量を見積もるのが難しいという問題がある。

【０００９】

本発明は、上記の点に鑑みてなされたものであって、データベースのアクセスの効率性とデータの一貫性とを適切に確保することのできる管理プログラム、管理方法、及び管理装置の提供を目的とする。

【課題を解決するための手段】

【００１０】

そこで上記課題を解決するため、管理プログラムは、データベースにおける第一のデータの参照の開始時に時間の前後関係を識別可能な第一の時系列データを前記第一のデータを識別する識別情報と対応付けて記憶部に記憶して、該第一のデータを参照する処理を開始し、前記第一のデータの更新要求に応じ、前記第一のデータに対応する第二のデータを前記データベースに生成し、前記更新要求に基づいて該第二のデータの更新を開始し、前記第二のデータの更新の完了時に第二の時系列データを前記第二のデータを識別する識別情報と対応付けて前記記憶部に記憶し、前記参照する処理の終了時に前記第一の時系列デ

ータを前記記憶部から削除し、前記更新の完了に応じて、前記第二の時系列データより前の時間を示す前記第一の時系列データが前記記憶部に記憶されているかを判定し、前記第二の時系列データより前の時間を示す第一の時系列データが前記記憶部に記憶されていない場合は前記第一のデータを削除することをコンピュータに実行させることを特徴とする。

【００１１】

このような管理プログラムでは、データベースのアクセスの効率性とデータの一貫性とを適切に確保することができる。

【発明の効果】

【００１２】

データベースのアクセスの効率性とデータの一貫性とを適切に確保することのできる管理プログラム、管理方法、及び管理装置を提供することができる。

【発明を実施するための最良の形態】

【００１３】

以下、図面に基づいて本発明の実施の形態を説明する。図１は、第一の実施の形態におけるデータ管理システムの構成例を示す図である。同図において、データ管理システムは、データベースサーバ１０及びアプリケーションサーバ２０を含む。データベースサーバ１０及びアプリケーションサーバ２０は、ＬＡＮ（Local Area Network）又はインターネット等のネットワーク（有線又は無線の別は問わない。）を介して接続されている。

【００１４】

データベースサーバ１０は、データベース及びデータベースに対するアクセス手段（操作手段）等が実装されたコンピュータであり、データベース１１、ＳＱＬ実行制御部１２、アクセス部１３、排他制御部１４、及びゴーストデーモン１５等を有する。

【００１５】

データベース１１は、記憶装置に体系的に記録されたデータ（情報）の集合である。本実施の形態では、データを表形式によって管理するＲＤＢ（Relational Database）を例とする。ＳＱＬ実行制御部１２は、ＳＱＬ（Structured Query Language）によるデータベース１１の操作指示を受信し、当該操作指示に応じた処理を制御する。アクセス制御部１３は、ＳＱＬ実行制御部１２からの指示に応じ、データベース１１に対する操作（更新、参照、削除等）を実行する。排他制御部１４は、データベースに対する操作時における排他制御を行うことによりデータの一貫性を保証する。ゴーストデーモン１５は、ゴーストトランザクションの管理等を行うデーモンプロセスである。ゴーストトランザクションとは、更新トランザクション又は削除トランザクションが所定の条件が満たされる場合に移行する（又は開始される）トランザクションをいう。ゴーストトランザクションは、トランザクション自体は実行せずに、移行元のトランザクションの占有情報（ロック情報）を引き継ぐ（保持する。）。したがって、移行元のトランザクションがゴーストトランザクションに移行した場合、当該移行元のトランザクションが終了後も操作対象のデータに対するロックは維持される。なお、所定の条件とは、該当するトランザクションのコミット時において操作対象とされていたデータと同一のデータを参照している他のトランザクションが存在することである。

【００１６】

アプリケーションサーバ２０は、例えば、ユーザによる指示入力に応じてデータベースサーバ１０に対してＳＱＬによるデータベース１１の操作指示を送信する。

【００１７】

図２は、本発明の実施の形態におけるデータベースサーバのハードウェア構成例を示す図である。図２のデータベースサーバ１０は、それぞれバスBで相互に接続されているドライブ装置１００と、補助記憶装置１０２と、メモリ装置１０３と、ＣＰＵ１０４と、インタフェース装置１０５とを有する。

【００１８】

データベースサーバ１０での処理を実現するプログラムは、ＣＤ－ＲＯＭ等の記録媒体

１０１によって提供される。プログラムを記録した記録媒体１０１がドライブ装置１００にセットされると、プログラムが記録媒体１０１からドライブ装置１００を介して補助記憶装置１０２にインストールされる。補助記憶装置１０２は、インストールされたプログラムを格納すると共に、データベース１１等を格納する。

【００１９】

メモリ装置１０３は、プログラムの起動指示があった場合に、補助記憶装置１０２からプログラムを読み出して格納する。ＣＰＵ１０４は、メモリ装置１０３に格納されたプログラムに従ってデータベースサーバ１０に係る機能を実行する。インタフェース装置１０５は、ネットワークに接続するためのインタフェースとして用いられる。

【００２０】

続いて、データベース１１におけるデータ構造について説明する。図３は、本発明の実施の形態におけるデータベースのデータ構造の例を示す図である。なお、本実施の形態において「データ」とは、論理的なテーブルにおける一つの行（論理行）に相当するデータベース資源をいう。

【００２１】

同図に示されるように、一つのデータは、物理的なレコードとして管理レコード、削除レコード、及び標準レコードの３つのレコードを含みうる。削除レコードとは、更新処理又は削除処理の際に一時的に生成されるレコードであり、対応するデータが更新中の際に参照対象とされる。削除レコードが生成されることにより、更新処理又は削除処理と参照処理との同時実行性が確保される。標準レコードは、対応するデータが削除されない限り常に存在するレコードであり、更新処理の際は更新対象とされる。また、削除レコードが有る場合、標準レコードは削除レコードの新バージョンのレコードであるともいえる。管理レコードは、二つの物理レコードアドレスとカレントレコードアドレスを含む。二つの物理レコードアドレスのうち一方は、削除レコードに対するアドレス情報（データベース１１における物理的な位置情報）である。他方は、標準レコードに対するアドレス情報である。カレントレコードアドレスは、二つの物理レコードアドレスのうち、最新バージョンのレコード（カレントレコード）のアドレス情報が格納されている領域に対する位置情報である。なお、管理レコードはデータが存在する限り常に存在する。すなわち、管理レコード及び標準レコードは常に存在し、削除レコードは、更新処理が行われる際に生成される。

【００２２】

以下、データベースサーバ１０の処理手順について説明する。図４は、挿入トランザクションの処理手順を説明するためのフローチャートである。

【００２３】

アプリケーションサーバ２０よりデータの挿入指示を受信したＳＱＬ実行制御部１２からの指示に応じ、アクセス部１３は、挿入対象のデータ（以下、「挿入データ」という。）を挿入するための空きレコード（管理レコード及び標準レコードを生成可能な空き領域）をデータベース１１より検索する（Ｓ１０１）。続いて、排他制御部１４は、検索された空きレコードに対して更新ロックを適用する（更新ロックをかける）（Ｓ１０２）。本実施の形態において、更新ロックには２種類存在する。一つはＥＸロックといい、もう一つはＳＸロックという。管理レコードに対するＥＸロックは、対応するデータに対して変更（挿入、更新、又は削除）が行われている最中であることを示すロックである。また、管理レコードに対するＳＸロックは、対応するデータに対して変更は行われている最中ではないないことを示す。一方、標準レコード又は削除レコードに対するＥＸロックは、ある特定の場合（ゴーストトランザクション後に開始した参照の場合）を除き参照対象とすることができないことを示す。また、標準レコード又は削除レコードに対するＳＸロックは参照対象とすることができることを示す。なお、ＥＸロック、ＳＸロックのそれぞれは更新ロックであることを除いてそれ自体に特段の絶対的な意味はない。すなわち、一方を他方と区別するための相対的な意味を有するに過ぎない。したがって、二つのロックは更新ロックとしての仕組みは同じであり、例えば、ＥＸロック又はＳＸロックを識別するた

めの識別子が付加されていると考えてもよい。ここでは、対応するデータがこれから挿入されるため管理レコードに対してＥＸロックが適用される。また、挿入中のデータを参照させることはできないため、標準レコードに対してＥＸロックが適用される。

【００２４】

続いて、アクセス部１３は、空きレコードに対して管理レコード及び標準レコードを生成する（Ｓ１０３）。標準レコードの生成において、挿入データが標準レコードに登録される。続いて、アクセス部１３は、生成された管理レコードに対して標準レコードのアドレス情報を物理レコードアドレスとして登録する（Ｓ１０４）。なお、ここでは、削除レコードは生成されないため、もう一つの物理レコードアドレスは空のままとされる。続いて、アクセス部１３は、管理レコードのカレントレコードアドレスに、標準レコードに係る物理レコードアドレスの位置情報を登録する（Ｓ１０５）。続いて、コミット処理が実行される（Ｓ１０６）。なお、コミット処理の詳細については後述する。

【００２５】

続いて、更新トランザクションについて説明する。図５は、更新トランザクションの処理手順を説明するためのフローチャートである。

【００２６】

アプリケーションサーバ２０よりデータの更新指示を受信したＳＱＬ実行制御部１２からの指示に応じ、アクセス部１３は、更新対象のデータ（以下、「更新データ」という。）に係るレコード（管理レコード及び標準レコード）をデータベース１１より検索する（Ｓ１２１）。続いて、排他制御部１４は、検索された管理レコード及び標準レコードに対してＳＸロックを適用する（Ｓ１２２）。続いて、アクセス部１３は、標準レコードについて、更新後の状態を登録するための空きレコード（標準レコードに対応する１レコード分）をデータベースより検索する（Ｓ１２３）。続いて、排他制御部１４は、更新処理を開始するため、検索された空きレコードと更新データに係る管理レコードとにＥＸロックを適用する（Ｓ１２４）。

【００２７】

続いて、アクセス部１４は、更新前の標準レコードに対して更新した結果を空きレコードに登録する（Ｓ１２５）。したがって、この時点において標準レコードには更新前の状態が、空レコードには更新後の状態が登録されていることになる。なお、図５における以下の説明において、更新前の状態が登録されたレコード（これまでの標準レコード）を「旧標準レコード」と呼ぶ。また、更新後の状態が登録されたレコードを（これまでの空きレコード）を「標準レコード」と呼ぶ。

【００２８】

続いて、アクセス部１３は、管理レコードに対して標準レコードのアドレス情報を物理レコードアドレスとして登録する（Ｓ１２６）。続いて、アクセス部１３は、管理レコードのカレントレコードアドレスに、標準レコードに係る物理レコードアドレスの位置情報を登録する（Ｓ１２７）。続いて、アクセス部１３は、旧標準レコードを削除レコードに変更する（Ｓ１２８）。削除レコードへの変更とは、該当するレコードが削除レコードであることを示す情報（削除フラグ）を記録することをいう。削除フラグは、該当するレコード（物理レコード）に記録してもよいし、該当するレコードに関連付けてメモリ装置１０３上に記録してもよい。続いて、コミット処理が実行される（Ｓ１２９）。

【００２９】

続いて、削除トランザクションについて説明する。図６は、削除トランザクションの処理手順を説明するためのフローチャートである。

【００３０】

アプリケーションサーバ２０よりデータの削除指示を受信したＳＱＬ実行制御部１２からの指示に応じ、アクセス部１３は、削除対象のデータ（以下、「削除データ」という。）に係るレコード（管理レコード及び標準レコード）をデータベース１１より検索する（Ｓ１４１）。続いて、排他制御部１４は、検索された管理レコード及び標準レコードに対してＳＸロックを適用する（Ｓ１４２）。続いて、排他制御部１４は、削除処理を開始す

るため、管理レコードにＥＸロックを適用する（Ｓ１４３）。続いて、アクセス部１４は、標準レコードを削除レコードに変更する（Ｓ１４４）。標準レコードから削除レコードへの変更については、図５のＳ１２８において説明した通りである（すなわち、該当レコードについて削除フラグを記録する。）。続いて、コミット処理が実行される（Ｓ１４５）。

【００３１】

図４、図５、又は図６において説明した処理手順より明らかなように、挿入処理中、更新処理中、又は削除処理中における、各レコードのロック状態は次の通りである。なお、「処理中」とは、コミット前の状態をいう。

【００３２】

図７は、操作対象とされたレコードのロック状態を示す図である。同図において、（Ａ）は挿入処理中のロック状態を示す。挿入処理中には、管理レコード及び挿入レコードのそれぞれにＥＸロックが適用される（図４参照）。（Ｂ）は更新処理中のロック状態を示す。更新処理中には、管理レコードと標準レコードに対してＥＸロックが適用され、削除レコード（又は削除レコードに変更される前の旧標準レコード）にはＳＸロックが適用される（図５参照）。したがって、更新処理中において、削除レコードは参照可能とされる。（Ｃ）は削除処理中のロック状態を示す。削除処理中には、管理レコードに対してＥＸロックが適用され、削除レコード（又は削除レコードに変更される前の標準レコード）にはＳＸロックが適用される。したがって、削除処理中において、削除レコードは参照可能とされる。なお、各レコードのロックの内容（状態）を示す情報（占有情報）は、トランザクション情報の一部としてトランザクションごとにメモリ装置１０３に記録（保持）される。

【００３３】

続いて、ステップＳ１０６、Ｓ１２９、及びＳ１４５におけるコミット処理について説明する。図８は、コミット処理の処理手順を説明するためのフローチャートである。同図の説明におけるレコードは、コミットの対象とされたトランザクション（以下、「カレントトランザクション」という。）において操作対象とされたデータに係るレコードをいう。

【００３４】

コミットの開始に際し、アクセス部１３は、カレントトランザクションに対して発行データを発行し、当該発行データをカレントトランザクションに関連付けて発行データ格納領域に登録する（Ｓ２０１）。発行データ格納領域は、メモリ装置１０３内における所定のメモリ領域である。

【００３５】

図９は、発行データの構成例を示す図である。同図において、発行データ１５０は、ノード番号、対象データＩＤ、時系列番号、及び発行時期種別等の情報を含む。ノード番号は、ロードシェア（負荷分散）環境（複数ノード環境）のようにデータベースサーバ１０が複数存在する構成において各データベースサーバ１０（ノード）を識別するための番号（識別子）である。したがって、図１に示されるように単一ノードの環境の場合は必須ではない。対象データＩＤは、発行データ１５０の発行対象とされたトランザクションにおいて操作対象とされたデータの識別子である。複数のデータが操作対象とされた場合は複数の対象データＩＤが登録される。時系列番号は、発行データ１５０の発行順序（時間の前後関係）を識別するための番号である。時系列番号は、例えば、通番で各発行データ１５０に割り当てられる。但し、時系列番号は、時間の前後関係を識別できるデータであれば所定のものに限定されない。例えば、時刻であってもよいし、他の符号であってもよい。発行時期種別は、発行データ１５０が発行された時期（タイミング）の種別（参照の開始又はコミットの開始時）を示す情報である。すなわち、発行データ１５０は、ステップＳ２０１のタイミング（コミットの開始時）のみならず、参照の開始時にも発行される。なお、参照の開始時に発行された発行データ１５０は、当該参照の終了に応じて削除される。

【００３６】
続いて、アクセス部１３は、当該コミットより前に開始され、カレントトランザクションにおいて操作対象とされたデータを参照する参照処理であって、現時点においても実行中のもの（以下、「並列参照処理」という。）の存否を判定する（Ｓ２０２）。この判定は、発行データ１５０に基づいて行われる。すなわち、ステップＳ２０１において発行された発行データ１５０の対象データＩＤと少なくとも一部が重複する対象データＩＤを含み、かつ、発行時期種別が参照の開始を示し、かつステップS２０１にて発行された発行データの時系列番号より小さい発行データ１５０の存否が判定される。斯かる発行データ１５０が存在する場合、並列参照処理は存在すると判定する。斯かる発行データ１５０が存在しない場合、並列参照処理は存在しないと判定する。
【００３７】
例えば、図１０は、発行データに基づく並列参照処理の存否の判定内容を説明するための図である。同図では、発行データ格納領域１６０に記録されている発行データ１５０が発行順に配列されている。
【００３８】
同図において、発行データ１５０ａは、参照の開始時に発行され、時系列番号が「１００」の発行データ１５０である。一方、発行データ１５０ｂは、更新のコミット時に発行され、時系列番号が「１０１」の発行データある。ここで、発行データ１５０ａの時系列番号（１００）は、発行データ１５０ｂの時系列番号（１０１）より小さい。したがって、発行データ１５０ａに係る参照は発行データ１５０ｂに係る更新のコミットよりも先に実行されたものであることが分かる。また、発行データ１５０ａが存在していることにより、当該参照はまだ終了していないことが分かる。よって、この場合、並列参照処理は存在すると判定される。
【００３９】
並列参照処理が存在しない場合（Ｓ２０２でＮＯ）、アクセス部１３は、当該通知に応じ、カレントトランザクションをコミットする（Ｓ２０３）。続いてアクセス部１３は、削除レコードが有る場合は削除レコードを削除する（Ｓ２０４）。図７の（Ｂ）、（Ｃ）より、更新又は削除が行われた場合は削除レコードが存在する。続いて、アクセス部１３は、管理レコードに登録されている物理データアドレスのうち、削除レコードに対応する物理データアドレスをクリアする（Ｓ２０５）。続いて、当該コミットが削除処理を含むトランザクションのコミットである場合（Ｓ２０６でＹＥＳ）、アクセス部１３は、削除されたデータに係る管理レコードを削除する（Ｓ２０７）。続いて、アクセス部１３は、ステップＳ２０１において生成された発行データ１５０を削除する（Ｓ２０８）。続いて、排他制御部１４は、カレントトランザクションによってロックされていたレコードに対してアンロック（ロックの解放）を実行する（Ｓ２０９）。より具体的には、カレントトランザクションに係る占有情報をクリア（削除）する。
【００４０】
一方、並列参照処理が存在する場合（Ｓ２０２でＹＥＳ）、アクセス部１３は並列参照処理が存在することをゴーストデーモン１５に通知する。当該通知に応じ、ゴーストデーモン１５は、並列参照処理によって参照されているレコードが削除されてしまうのを防止するためゴーストトランザクションを生成する（Ｓ２１０）。この際、ゴーストデーモン１５は、当該ゴーストトランザクションにカレントトランザクションの占有情報及び発行データ１５０を引き継がせる。占有情報がゴーストトランザクションに引き継がれることにより（すなわち、カレントトランザクションのコミットに応じてその占有情報が削除されることが回避されることにより）、並列参照処理によって参照されているレコードの削除が防止される。すなわち、排他制御部１４は、ゴーストトランザクションが有する占有情報にも基づいて（をも参照して）、データベース１１に対する排他制御を行うからである。ゴーストトランザクションの実体は、例えば、占有情報を管理するためにメモリ装置１０３内に生成されるデータでよい。したがって、占有情報のゴーストトランザクションへの引継ぎとは当該占有情報をメモリ装置１０３内に保持させることに相当する。なお、

当該データが、他の通常のトランザクションの情報と同様の形式の場合は、ゴーストトランザクションであるか否かを示す情報を当該データに記録すればよい。続いて、アクセス部１３は、カレントトランザクションをコミットする（Ｓ２１１）。これにより、カレントトランザクションは、ゴーストトランザクションに移行したことになる。

【００４１】

図８より明らかなように、ゴーストトランザクションは、例えば、次のような契機によって生成される（発生する）。図１１は、ゴーストトランザクションの発生の契機の具体例を示す図である。図中、左から右へ時間が経過している。また、太線の矢印は、参照処理又は更新トランザクションを示す。太い破線の矢印はゴーストトランザクションを示す。また、参照処理の対象データと更新トランザクションの対象データの少なくとも一部は重複する。

【００４２】

同図では、まず、参照処理が開始されている（Ｓ２１）。参照処理の開始に応じ、当該参照処理に対する発行データ１５０ｒが発行され、発行データ格納領域１６０に登録される（Ｓ２２）。その後更新トランザクションが開始される（Ｓ２３）。本実施の形態では、参照処理の際に参照ロックは行われない。したがって、既に参照中のデータに対して更新を行おうとしても排他されない。更新トランザクションがコミットされると更新トランザクションに対する発行データ１５０ｕが発行され、発行データ格納領域１６０に登録される（Ｓ２４）。このタイミングで、更新トランザクションがゴーストトランザクションへ移行すべきか否かが判定される。ここでは、発行データ１５０ｕの時系列番号よりも小さな時系列番号を有する発行データ１５０ｒが存在する（Ｓ２５）。したがって、更新トランザクションはゴーストトランザクションへ移行する（Ｓ２６）。その後、ゴーストトランザクションへの移行の要因となった参照処理が終了すると（Ｓ２７）、当該参照処理に対する発行データ１５０ｒが削除される（Ｓ２８）。その結果、発行データ１５０ｕより時系列番号が小さい発行データ１５０は存在しないため、発行データ１５０ｕが削除され（Ｓ２９）、ゴーストトランザクションは終了する（Ｓ３０）。

【００４３】

続いて、データの参照処理について説明する。図１２は、参照処理の処理手順を説明するためのフローチャートである。

【００４４】

アプリケーションサーバ２０よりデータの参照指示を受信したＳＱＬ実行制御部１２からの指示に応じ、アクセス部１３は、当該参照処理を実行するトランザクション（カレントトランザクション）に対して発行データ１５０（図９参照）を発行し、発行データ格納領域１６０に登録する（Ｓ３０１）。なお、ここで生成される発行データ１５０の発行時期種別には参照の開始を示す値が登録される。

【００４５】

続いて、データベースサーバ１０は、参照対象のデータ（以下、「参照データ」）の参照を開始する（Ｓ３０２）。なお、上記したように、本実施の形態では参照時において参照ロックは行われない。

【００４６】

まず、排他制御部１４は、参照データに係るレコードは既にロック（占有）されているか否かを確認（判定）する（Ｓ３０３）。ロックされているか否かは、現在存続（実行）中のトランザクション（ゴーストトランザクションも含む。）の占有情報に基づいて判定される。なお、以下の図１２の説明におけるレコードは、参照データに係るレコードをいう。

【００４７】

ロックされていない場合（Ｓ３０３でＮＯ）、又はロックされていたとしてもロックの主体がカレントトランザクションである場合（Ｓ３０４でＹＥＳ）、アクセス部１３は、管理レコードにおけるカレントレコードアドレスが指す物理レコードアドレスに係る標準レコードを参照対象とする（Ｓ３０５）。

【００４８】
一方、他のトランザクションによってロックされている場合（Ｓ３０４でＮＯ）、排他制御部１４は、ロックの主体がゴーストトランザクションであるか否かを判定する（Ｓ３０６）。ゴーストトランザクションであるか否かは、占有情報に関連付けて記録されている情報（ゴーストトランザクションであるか否かを示す情報）に基づいて判定すればよい。

【００４９】
ロックの主体がゴーストトランザクションで無い場合（これは、ロックの主体のトランザクションがまだ存続している場合に相当する。）（Ｓ３０６でＮＯ）、アクセス部１３は、標準レコード及び削除レコードのうち、ＳＸロックが適用されているレコードを参照対象とする（Ｓ３０７）。例えば、図７（Ｂ）又は（Ｃ）の状態であれば、削除レコードが参照される。但し、例えば、（Ｃ）において、標準レコードが削除レコードへ変更される前であれば（図６参照）、標準レコードが参照される。一方、ロックの主体がゴーストトランザクションである場合（これは、ロックの主体のトランザクションが既にコミットされている場合に相当する。）（Ｓ３０６でＹＥＳ）、アクセス部１３は、生成されている発行データ１５０に基づいて参照対象を決定（判定）する（Ｓ３０８）。当該ステップの詳細については後述する。

【００５０】
続いて、アクセス部１３は、参照対象とされたレコードを参照し（Ｓ３０９）、参照を終了させる（Ｓ３１０）。続いて、アクセス部１３は、ステップＳ３０１において発行した発行データ１５０を削除する（Ｓ３１１）。

【００５１】
続いて、発行データ１５０を削除することにより終了するゴーストトランザクションの有無の判定、及び終了が可能なゴーストトランザクションが存在する場合は、終了可能なすべてのゴーストトランザクションの終了処理が実行される。したがって、ゴーストトランザクションが存在しない場合、以降の図１２中の破線で囲まれた処理Ｐ４は実行されない。なお、Ｐ４の処理はゴーストデーモン１５にて実施される処理であり、参照処理とは非同期に実施される。

【００５２】
まず、アクセス部１３は、削除した発行データ１５０がゴーストトランザクションの最後の要因の参照処理に関するものであるか否かを判定する（Ｓ３１２）。ゴーストトランザクションの要因となった参照処理とは、ゴーストトランザクションの移行元のトランザクションのコミット前に同一データに対して行われた参照処理をいう。

【００５３】
当該判定内容の具体例を図１３に示す。図１３は、ゴーストトランザクションの最後の要因となった参照処理の終了の判定内容を説明するための図である。図１３中、図１０と同一部分には同一符号を付している。但し、同図において発行データ１５０ｂはゴーストトランザクションに係る発行データ１５０であるとする。したがって、発行データ１５０ａは、ゴーストトランザクションの要因となった参照処理に係る発行データ１５０である。

【００５４】
同図では、発行データ１５０ｂより時系列番号が小さい発行データ１５０ａは削除されている（図中×は削除を示す。）。また、発行データ１５０ａ以外に、発行データ１５０ｂより時系列番号が小さい発行データ１５０は存在しない。したがって、この場合、削除された発行データ１５０ａがゴーストトランザクションの最後の要因であった（ゴーストトランザクションの要因となった参照処理は全て終了している）と判定する。一方、図１０のような状態の場合、ゴーストトランザクションの要因となった参照処理の全ては終了していない（終了した参照処理以外にまだゴーストトランザクションの要因となる参照処理が存在する）と判定する。

【００５５】

終了可能と判断されたゴーストトランザクションが存在する場合（Ｓ３１２でＹＥＳ）、アクセス部１３はその旨をゴーストデーモン１５に通知する。当該通知に応じ、ゴーストデーモン１５は、削除レコードが有る場合は削除レコードを削除する（Ｓ３１３）。続いて、ゴーストデーモン１５は、管理レコードに登録されている物理データアドレスのうち、削除レコードに対応する物理データアドレスをクリアする（Ｓ３１４）。続いて、ゴーストデーモン１５は、当該ゴーストトランザクションが開始されたタイミングが削除処理を含むトランザクションのコミットであるか否かを判定する（Ｓ３１５）。当該判定は、当該ゴーストトランザクションに係る発行データ１５０の発行時期種別に基づいて行えばよい。削除処理を含むトランザクションのコミットである場合（Ｓ３１５でＹＥＳ）、ゴーストデーモン１５は、削除されたデータに係る管理レコードを削除する（Ｓ３１６）。続いて、ゴーストデーモン１５は、当該ゴーストトランザクションに係る発行データ１５０を削除する（Ｓ３１７）。続いて、ゴーストデーモン１５は、当該ゴーストトランザクションを終了させる（Ｓ３１８）。具体的には、当該ゴーストトランザクションの実体としてのデータを削除する。したがって、当該データに含まれる占有情報もクリア（削除）され、ゴーストトランザクションによるロックがアンロックされる。

【００５６】

このように、ゴーストトランザクションの要因となった参照処理が全て終了することにより、ゴーストトランザクションは終了し、ゴーストトランザクションによるロックは解放される。

【００５７】

続いて、ステップＳ３０８の詳細について説明する。図１４は、ゴーストトランザクションによって占有されているデータの参照対象レコードの判定処理を説明するためのフローチャートである。

【００５８】

ステップＳ３０８１において、アクセス部１３は、ステップＳ３０１において参照処理に対して発行された発行データ１５０の時系列番号（参照番号）と、ロックの主体のゴーストトランザクションに係る発行データ１５０の時系列番号（コミット番号）とを比較する。参照番号の時系列番号がコミット番号の時系列番号未満の場合（Ｓ３０８２でＹＥＳ）、アクセス部１３は、ＳＸロックが適用されているレコードを参照対象とする（Ｓ３０８３）。参照番号がコミット番号より大きい場合（Ｓ３０８２でＮＯ）、アクセス部１３は、管理レコードのカレントレコードアドレスが指す物理レコードアドレスに係るＥＸロックが適用されたレコードを参照対象とする（Ｓ３０８４）。

【００５９】

図１４の処理の具体例を示す。図１５は、ゴーストトランザクションによって占有されているデータの参照対象レコードの判定処理の具体例を示す図である。図中、左から右へ時間が経過している。また、太線の矢印は参照処理又は更新トランザクションを示す。太い破線の矢印はゴーストトランザクションを示す。

【００６０】

同図では、まず、参照処理Ａが開始されて、時系列番号が９９の発行データ１５０Ａが発行されている。続いて、更新トランザクションがコミットされ、時系列番号が１００の発行データ１５０Ｕが発行されている。更新トランザクションのコミット時において、参照処理Ａは並列参照処理であるため、更新トランザクションはゴーストトランザクションに移行する。続いて、参照処理Ｂが開始され、時系列番号１０１の発行データ１５０Ｂが発行されている。

【００６１】

その後、時間ｔ１のタイミングで参照処理Ａ及びＢの双方が、ゴーストトランザクションによって占有されているデータを参照しようとする。この際、参照処理Ａは、ゴーストトランザクションの時系列番号（１００）より小さい時系列番号（９９）を有するため（すなわち、参照処理Ａは更新トランザクションのコミットより前に開始しているため）、ＳＸロックのレコード（更新前のレコード（削除レコード））を参照対象とする（Ｓ３０

８３に対応）。一方、参照処理Ｂは、ゴーストトランザクションの時系列番号（１００）より大きい時系列番号（１０１）を有するため（すなわち、参照処理Ｂは更新トランザクションのコミットより後に開始しているため）、ＥＸロックのレコード（更新後のレコード（標準レコード））を参照対象とする（Ｓ３０８４に対応）。このように、時系列番号は、参照対象とする物理レコードの判定にも用いられる。

【００６２】

上述したように、第一の実施の形態によれば、更新中のデータについても、参照処理の開始時点のコミット済みの整合性の有るレコード（削除レコード）を、排他待ちすることなく参照させることができる。したがって、データベースへのアクセスの効率化を図ることができる。

【００６３】

また、発行データ１５０により更新処理等のコミットの時期や参照処理の開始時期の前後関係を把握することができる。したがって、コミット時において同一データに対する参照処理の有無を適切に判定することができる。したがって、更新処理等のコミットに応じて参照中のレコード（削除レコード）が削除されることを防止することができる。また、参照の終了時において、発行データ１５０が削除され、それに応じて更新処理等のコミットより前に発行された発行データ１５０の存在の有無が判定される。その結果、斯かる発行データ１５０が存在しない場合は削除レコードが削除される。したがって、適切なタイミングで削除レコードを削除することができる。

【００６４】

また、更新処理等のコミット時において、同一データに対する参照処理が存在する場合は当該更新処理等に係るトランザクションの占有情報はコミットに応じて削除されずにゴーストトランザクションに引き継がれて保持される。そして、ゴーストトランザクションは、当該参照処理が終了するまで存続する。したがって、参照対象とされているデータに対して適切に排他制御を行うことができる。

【００６５】

また、各データについて管理されるバージョンは最大で２バージョン（標準レコードと削除レコード）であるため、データベース格納領域の容量設計を容易化することができる。

【００６６】

また、従来技術における参照カウンタと異なり、発行データ１５０は高速にアクセス可能なメモリ装置１０３内に記録される（データベース１１の物理レコードには記録されない。）。したがって、参照処理の性能の劣化を防止することができる。仮に、従来技術の参照カウンタの全てをメモリ上で管理したとしても、本実施の形態の方が有利である。すなわち、参照処理が１度に複数のレコード、例えば１０００件のレコードを参照する場合を考える。この場合、従来技術であれば、１０００件分の参照カウンタを更新することになり、その分多くのＣＰＵやメモリを消費することになる。しかし、本実施の形態では、１０００件分の対象データＩＤが一つの発行データ１５０によって管理される。したがって、一つの発行データ１５０を生成又は削除すればよい。このように、本実施の形態では、１度に参照するデータ数が増えるほど、従来技術に対して有利となる。

【００６７】

また、発行データ１５０がメモリ装置１０３において管理されることにより、データベース１１において障害が発生した場合であっても、参照カウンタの補正等の処理は不要である。したがって、高速にリカバリ処理を行うことができる。

【００６８】

上記より、本実施の形態は、参照処理が多く、参照処理の負荷が高いデータベースに対して特に効果的であると言える。

【００６９】

また、本実施の形態では、不要となった（参照されなくなった）削除データの削除は、ゴーストデーモン１５によってバックグラウンドで行われるため、参照処理自体に対する

負荷の増大が抑止される。

【００７０】

なお、発行データ１５０における対象データＩＤは、必須項目ではない。対象データＩＤが無い場合、コミット時（図８）や参照終了時（図１２）等において、参照対象又は更新対象等が異なる発行データ１５０も比較対象とされる。しかし、この場合であっても、更新と参照との並列性を維持しつつ、更新等のコミットによる参照中のレコードの削除を防止することができる。但し、この場合、ゴーストデーモンの終了時期が遅れ、それによりゴーストデーモンによるロックが必要以上に長く維持される可能性がある。その結果、他の更新処理等が排他される可能性がある。このような可能性を考慮すれば、本実施の形態のように、発行データ１５０において対象データＩＤを管理することが望ましい。

【００７１】

次に、第二の実施の形態について説明する。第二の実施の形態では第一の実施の形態と異なる点について説明する。したがって、特に言及しない点については第一の実施の形態と同様でよい。

【００７２】

図１６は、第二の実施の形態におけるデータ管理システムの構成例を示す図である。図１６中、図１と同一部分には同一符号を付し、その説明は省略する。

【００７３】

同図に示されるように、第二の実施の形態では２つ以上のノード（データベースサーバ１０（データベースサーバ１０ａ、１０ｂ））がアプリケーションサーバ２０に接続されている。すなわち、第二の実施の形態におけるデータ管理システムではロードシェア環境が実現されている。したがって、アプリケーションサーバ２０は任意のデータベースサーバ１０に接続（コネクション）してＳＱＬ文を実行させることがきる。なお、各データベースサーバ１０の構成例は、図１に示されるものと同様でよい。二つのデータベース１０の構成要素を区別する場合、データベースサーバ１０ａの構成要素については参照番号の末尾に「ａ」を付加する。データベースサーバ１０ｂの構成要素については参照番号の末尾に「ｂ」を付加する。

【００７４】

図１７は、第二の実施の形態におけるデータ管理システムの動作概要を説明するための図である。なお、同図では、アクセス部１３、排他制御部１４、及びゴーストデーモン１５は、便宜上省略されている。

【００７５】

アプリケーションサーバ２０といずれか一方のデータベースサーバ１０（同図の例では、データベースサーバ１０ａ）とのコネクションが確立された後、他方のデータベースサーバ１０（データベースサーバ１０ｂ）によって管理されているデータに対するＳＱＬの実行要求が行われるとする（Ｓ２１）。そうすると、データベースサーバ１０ａのＳＱＬ実行御部１２ａは、他ノードのＳＱＬ実行制御部１２ｂにＳＱＬの実行制御を依頼する（Ｓ２２）。この場合、アプリケーションサーバ２０とのコネクションを確立したデータベースサーバ１０ａを「チーフノード」という。また、チーフノードからＳＱＬの実行制御を依頼されたデータベースサーバ１０ｂを「アシスタントノード」という。

【００７６】

ところで、ロードシェア環境では、各ノードのシステム時刻が必ずしも一致するとは限らない。したがって、各ノードのシステム時刻に基づいてトランザクション等の前後関係を判定した場合、正しい前後関係が得られない可能性がある。そこで、第二の実施の形態では、ロードシェア環境における斯かる問題点を考慮して、第一の実施の形態を利用してデータの一貫性及びトランザクションの同時実行性を適切に確保した例を説明する。

【００７７】

図１８は、第二の実施の形態の処理概要を説明するための図である。また、図１９は、第二の実施の形態の処理概要において各ノードで管理される発行データの遷移を示す図である。図中、ノード０はデータベースサーバ１０ａを示す。また、ノード１はデータベー

スサーバ１０ｂを示す。なお、図１８において参照処理の対象データと更新トランザクションの対象データは同じであるとする。

【００７８】

例えば、ノード１において参照処理が開始されると（Ｓ３１）、ノード１において発行データ１５０ａ（時系列番号：２００）が発行される（Ｓ３２）。なお、ノード１は当該参照処理のチーフである。続いて、ノード１において参照処理が開始されたことがノード０に通知され、ノード０において当該参照処理に対する発行データ１５０ｂ（時系列番号：１００）が発行される（Ｓ３３）。なお、時系列番号はノードごとに通番である。続いて、ノード０において発行された発行データ１５０ｂがノード１の参照に送信される（Ｓ３４）。この段階において、各ノードに管理されている発行データ１５０は、図１９（Ａ）の通りとなる。すなわち、ノード０には発行データ１５０ｂが管理されている。また、ノード１には発行データ１５０ａが、ノード1の参照には１５０ａ及び１５０ｂが管理されている。

【００７９】

続いて、ノード０において、ノード１に属するデータに対する更新を含む要求に応じ更新トランザクションが開始されたとする（Ｓ３５）。この場合、ノード０は更新トランザクションのマスタとして、ノード１に対して当該更新トランザクションを依頼する（Ｓ３６）。したがって、ノード１はアシスタントとして更新トランザクションを開始する（Ｓ３７）。続いて、ノード０において更新トランザクションのコミットが開始されると、ノード０において発行データ１５０ｃ（時系列番号：１０１）が発行される（Ｓ３８）。

【００８０】

続いて、ノード０において更新トランザクションのゴーストトランザクションへの移行の要否が判定される。図１９（Ｂ）を参照すると、ノード０には更新のコミットに係る発行データ１５０ｃより時系列番号が小さい参照に係る発行データ１５０ｂが存在する。したがって、更新トランザクションをゴーストトランザクションに移行させる（Ｓ４０）。

【００８１】

ノード０は、発行された発行データ１５０ｃをコミットの通知及びゴーストトランザクションへの移行の要否の判定結果と共にノード１に送信する（Ｓ３９）。この段階において、各ノードに管理されている発行データ１５０は、図１９（Ｂ）の通りとなる。すなわち、（Ａ）の発行データ１５０に加え、発行データ１５０ｃがノード０及びそれぞれのノードの更新トランザクションに追加されている。

【００８２】

また、ノード１はコミットの通知及びゴーストトランザクションへの移行の要否の判定結果に応じ、アシスタントとしての更新トランザクションのコミット及びゴーストトランザクションへの移行を行う（Ｓ４１）。なお、図中において、ノード番号が記載されていない発行データ１５０は、自ノードで発行された発行データ１５０であることを示す。

【００８３】

続いて、ノード１において参照処理が終了すると、ノード１は、当該参照処理に対応する発行データ１５０ａを削除する（Ｓ４２）。また、ノード１は、当該参照処理に対してノード０で発行された発行データ１５０ｂの削除をノード０に要求する（Ｓ４３）。当該要求に応じ、ノード０は、ノード０において管理されている発行データ１５０ｂを削除する。この段階において、各ノードに管理されている発行データ１５０は、図１９（Ｃ）の通りとなる。発行データ１５０ａ及び１５０ｂの削除により、ノード０及び各ノードのゴーストトランザクションが、発行データ１５０ｃを保持している。

【００８４】

続いて、ノード１及びノード０のそれぞれは、参照処理の発行データ１５０ａ及び１５０ｂの削除に応じ、ゴーストトランザクションの終了判定を行う。図１９（Ｃ）より、ノード０には、発行データ１５０ｃより時系列番号が小さい参照の発行データ１５０は存在しない。したがって、ノード０が保持する発行データ１５０ｃを削除し、それぞれのゴーストトランザクションを終了させる。この段階において各ノード及び各種処理において管

理されている発行データ１５０は、図１９（Ｄ）の通りとなる。

【００８５】

図１８及び図１９を用いて説明した処理内容を一般化してフローチャートを用いて説明する。図２０は、第二の実施の形態における参照処理の処理手順を説明するためのフローチャートである。同図は、第一の実施の形態における図１２に対応する。なお、同図において、「チーフ」は、アプリケーションサーバ２０より参照処理の要求を受け付けたノード（データベースサーバ１０）を示す。「他ノード」は、チーフ以外の全ノードを示す。

【００８６】

アプリケーションサーバ２０よりデータの参照指示を受信したチーフのＳＱＬ実行制御部１２からの指示に応じ、チーフのアクセス部１３は、当該参照処理を実行するトランザクション（カレントトランザクション）に対して発行データ１５０（図９参照）を発行（メモリ装置１０３内に生成）する（Ｓ５０１）。なお、ここで生成される発行データ１５０の発行時期種別には参照の開始を示す値が登録される。続いて、チーフのアクセス部１３は、当該参照処理に対応する発行データ１５０の発行を他ノードの各アクセス部１３に対して要求する（Ｓ５０２）。他ノードの各アクセス部１３は、当該要求を受けて、発行データ１５０を発行し、各ノードの発行データ格納領域に登録する（Ｓ５０３）。続いて、他ノードの各アクセス部１３は、それぞれが発行した発行データ１５０をチーフのアクセス部１３に対して送信する（Ｓ５０４）。チーフのアクセス部１３は、他ノードからの発行データ１５０を受信する（Ｓ５０５）。これにより、他ノードの各発行データ１５０がチーフに収集される。

【００８７】

続いて、チーフは、参照対象のデータ（以下、「参照データ」）の参照を開始する（Ｓ５０６）。まず、チーフは、図１２の破線Ｐ１で囲まれた処理（Ｓ３０３～Ｓ３０８）と同様の処理手順によって参照対象のレコードを判定する（Ｓ５０７）。参照データがチーフのみに属する場合（Ｓ５０８でＮＯ）、チーフのアクセス部１３は、参照対象とされたレコードを参照する（Ｓ５０９）。一方、参照データが他ノード（アシスタント）に属し、参照処理をアシスタントに派生させる（依頼する）必要がある場合（Ｓ５０８でＹＥＳ）、チーフのアクセス部１３は、アシスタントのアクセス部１３に対して参照処理を派生させる（Ｓ５１０）。続いて、アシスタントにおいて図１２の破線Ｐ１で囲まれた部分（Ｓ３０３～Ｓ３０８）と同様の処理手順が実行され、参照対象のレコードが判定される（Ｓ５１１）。続いて、アシスタントのアクセス部１３は、参照対象とされたレコードを参照する（Ｓ５１２）。

【００８８】

ステップＳ５０９及びＳ５１２に続いて、チーフのアクセス部１３は参照処理を終了させる（Ｓ５１３）。続いて、チーフのアクセス部１３は、ステップＳ５０１において発行した発行データ１５０を削除する（Ｓ５１４）。続いて、チーフのアクセス部１３は、他ノードのアクセス部１３に対して発行データ１５０の削除を要求する（Ｓ５１５）。続いて、チーフは、図１２の破線Ｐ２で囲まれた処理（Ｓ３１２～Ｓ３１８）と同様の処理手順によってチーフにおけるゴーストトランザクションの終了化処理を実行する（Ｓ５１６）。

【００８９】

一方、発行データ１５０の削除要求を受信した他ノードの各アクセス部１３は、ステップＳ５０３においてそれぞれのノードで発行された発行データ１５０を削除する（Ｓ５１７）。続いて、他ノードは、図１２の破線Ｐ２で囲まれた処理（Ｓ３１２～Ｓ３１８）と同様の処理手順によって各他ノードにおけるゴーストトランザクションの終了化処理を実行する（Ｓ５１８）。

【００９０】

続いて、図２１は、第二の実施の形態におけるコミット処理の処理手順を説明するためのフローチャートである。同図は、第一の実施の形態における図８に対応する。

【００９１】

ステップＳ６０１～Ｓ６０３では、チーフにおいて、図８のステップＳ２０１～Ｓ２０３と同様の処理手順が実行される。続いて、チーフは、図８の破線Ｐ３で囲まれた処理（Ｓ２０４～Ｓ２０８）と同様の処理手順によってレコードの削除処理を実行する（Ｓ６０４）。続いて、チーフの排他制御部１４は、コミット対象のトランザクション（カレントトランザクション）によってロックされていたレコードに対してアンロック（ロックの解放）を実行する（Ｓ６０５）。

【００９２】

一方、並列参照処理が存在する場合であって（Ｓ６０２でＹＥＳ）、カレントトランザクションがアシスタントに依頼されている場合（Ｓ６０６でＹＥＳ）、チーフのアクセス部１３は、アシスタントにステップＳ６０１で発行された発行データ１５０とコミットの実行及びゴーストトランザクションへと移行することを通知する（Ｓ６０７、Ｓ６０８）。

【００９３】

一方、並列参照処理が存在する場合であって（Ｓ６０２でＹＥＳ）、カレントトランザクションがアシスタントに依頼されていない場合（Ｓ６０６でＮＯ）、及びステップＳ６０８に続いて、チーフは、図８のステップＳ２１０、Ｓ２１１と同様の処理手順によって、ゴーストトランザクションの生成、カレントトランザクションのコミットを実行する（Ｓ６０９、Ｓ６１０）。

【００９４】

上述したように、第二の実施の形態によれば、自ノードと他ノードとのシステム時間が異なる場合であっても他ノードから依頼された更新と自ノードの参照との前後関係を適切に把握することができる。

【００９５】

以上、本発明の実施例について詳述したが、本発明は斯かる特定の実施形態に限定されるものではなく、特許請求の範囲に記載された本発明の要旨の範囲内において、種々の変形・変更が可能である。

【００９６】

以上の説明に関し、更に以下の項を開示する。

（付記１）

データベースにおける第一のレコードの参照の開始時に時間の前後関係を識別可能な第一の時系列データをメモリ装置に生成後、該第一のレコードを参照する参照開始手順と、

前記第一のレコードの更新要求に応じ、前記第一のレコードに対応する第二のレコードを前記データベースに生成し、該第二のレコードを更新する更新手順と、

前記更新のコミット時に第二の時系列データを前記メモリ装置に生成するコミット手順と、

前記参照の終了時に前記第一の時系列データを削除する参照終了手順と、

前記コミット手順又は前記参照終了手順に応じて前記第二の時系列データより前に生成された前記第一の時系列データが無くなった場合は前記第一のレコードを削除する削除手順とをコンピュータに実行させるためのデータ管理プログラム。

（付記２）

前記第一及び第二の時系列データには、前記参照又は更新の対象とされたレコードの識別子が関連付けられ、

前記削除手順は、前記第二の時系列データより前に生成された前記第一の時系列データであって、当該第二の時系列データと同一の前記レコードの識別子に関連付けられている前記第一の時系列データが無くなった場合は前記第一のレコードを削除する付記１記載のデータ管理プログラム。

（付記３）

前記更新手順は、前記第一のレコード及び前記第二のレコードに対してロックを適用し、

前記コミット手順に応じて、前記第二の時系列データより前に生成された前記第一の時

系列データが有る場合は前記ロックの内容を示す占有情報を前記メモリ装置に保持させる占有情報保持手順と、
前記メモリ装置に保持されている占有情報に基づいて前記データベースの排他制御を行う排他制御手順と、
前記参照終了手順に応じて前記第二の時系列データより前に生成された前記第一の時系列データが無くなった場合は前記占有情報を削除する占有情報削除手順とを有する付記１又は２記載のデータ管理プログラム。
（付記４）
前記第二の時系列データより前に生成された前記第一の時系列データに係る参照については前記第一のレコードを参照対象とし、前記第二の時系列データより後に生成された前記第一の時系列データに係る参照については前記第二のレコードを参照対象とする参照手順を有する付記１乃至３いずれか一項記載のデータ管理プログラム。
（付記５）
前記第一及び第二の時系列データには、複数ノード環境における各ノードを識別するノード識別子が関連付けられ、
前記参照開始手順は、他ノードに対して前記第一の時系列データの生成を要求し該要求に応じて該他ノードで生成された前記第一の時系列データを取得し、
前記更新手順は、前記他ノードからの更新要求に応じ、前記第一のレコードに対応する第二のレコードを前記データベースに生成し、該第二のレコードを更新し、
前記他ノードにおける前記更新に対するコミットに応じて、該他ノードにおいて生成された前記第二の時系列データを取得する他ノードコミット手順と、
前記他ノードコミット手順に応じて、前記他ノードより取得された前記第二の時系列データより前に生成された前記他ノードより取得された前記第一の時系列データが無くなった場合は前記第一のレコードを削除する第二の削除手順とを有する付記１乃至４いずれか一項記載のデータ管理プログラム。
（付記６）
コンピュータが実行するデータ管理方法であって、
データベースにおける第一のレコードの参照の開始時に時間の前後関係を識別可能な第一の時系列データをメモリ装置に生成後、該第一のレコードを参照する参照開始手順と、
前記第一のレコードの更新要求に応じ、前記第一のレコードに対応する第二のレコードを前記データベースに生成し、該第二のレコードを更新する更新手順と、
前記更新のコミット時に第二の時系列データを前記メモリ装置に生成するコミット手順と、
前記参照の終了時に前記第一の時系列データを削除する参照終了手順と、
前記コミット手順又は前記参照終了手順に応じて前記第二の時系列データより前に生成された前記第一の時系列データが無くなった場合は前記第一のレコードを削除する削除手順とを有するデータ管理方法。
（付記７）
前記第一及び第二の時系列データには、前記参照又は更新の対象とされたレコードの識別子が関連付けられ、
前記削除手順は、前記第二の時系列データより前に生成された前記第一の時系列データであって、当該第二の時系列データと同一の前記レコードの識別子に関連付けられている前記第一の時系列データが無くなった場合は前記第一のレコードを削除する付記６記載のデータ管理方法。
（付記８）
前記更新手順は、前記第一のレコード及び前記第二のレコードに対してロックを適用し、
前記コミット手順に応じて、前記第二の時系列データより前に生成された前記第一の時系列データが有る場合は前記ロックの内容を示す占有情報を前記メモリ装置に保持させる占有情報保持手順と、

前記メモリ装置に保持されている占有情報に基づいて前記データベースの排他制御を行う排他制御手順と、
前記参照終了手順に応じて前記第二の時系列データより前に生成された前記第一の時系列データが無くなった場合は前記占有情報を削除する占有情報削除手順とを有する付記６又は７記載のデータ管理方法。
（付記９）
前記第二の時系列データより前に生成された前記第一の時系列データに係る参照については前記第一のレコードを参照対象とし、前記第二の時系列データより後に生成された前記第一の時系列データに係る参照については前記第二のレコードを参照対象とする参照手順を有する付記６乃至８いずれか一項記載のデータ管理方法。
（付記１０）
前記第一及び第二の時系列データには、複数ノード環境における各ノードを識別するノード識別子が関連付けられ、
前記参照開始手順は、他ノードに対して前記第一の時系列データの生成を要求し該要求に応じて該他ノードで生成された前記第一の時系列データを取得し、
前記更新手順は、前記他ノードからの更新要求に応じ、前記第一のレコードに対応する第二のレコードを前記データベースに生成し、該第二のレコードを更新し、
前記他ノードにおける前記更新に対するコミットに応じて、該他ノードにおいて生成された前記第二の時系列データを取得する他ノードコミット手順と、
前記他ノードコミット手順に応じて、前記他ノードより取得された前記第二の時系列データより前に生成された前記他ノードより取得された前記第一の時系列データが無くなった場合は前記第一のレコードを削除する第二の削除手順とを有する付記６乃至９いずれか一項記載のデータ管理方法。
（付記１１）
データベースにおける第一のレコードの参照の開始時に時間の前後関係を識別可能な第一の時系列データをメモリ装置に生成後、該第一のレコードを参照する参照開始手段と、
前記第一のレコードの更新要求に応じ、前記第一のレコードに対応する第二のレコードを前記データベースに生成し、該第二のレコードを更新する更新手段と、
前記更新のコミット時に第二の時系列データを前記メモリ装置に生成するコミット手段と、
前記参照の終了時に前記第一の時系列データを削除する参照終了手段と、
前記コミット手段又は前記参照終了手段に応じて前記第二の時系列データより前に生成された前記第一の時系列データが無くなった場合は前記第一のレコードを削除する削除手段とを有するデータ管理装置。
（付記１２）
前記第一及び第二の時系列データには、前記参照又は更新の対象とされたレコードの識別子が関連付けられ、
前記削除手段は、前記第二の時系列データより前に生成された前記第一の時系列データであって、当該第二の時系列データと同一の前記レコードの識別子に関連付けられている前記第一の時系列データが無くなった場合は前記第一のレコードを削除する付記１１記載のデータ管理装置。
（付記１３）
前記更新手段は、前記第一のレコード及び前記第二のレコードに対してロックを適用し、
前記コミット手段に応じて、前記第二の時系列データより前に生成された前記第一の時系列データが有る場合は前記ロックの内容を示す占有情報を前記メモリ装置に保持させる占有情報保持手段と、
前記メモリ装置に保持されている占有情報に基づいて前記データベースの排他制御を行う排他制御手段と、
前記参照終了手段に応じて前記第二の時系列データより前に生成された前記第一の時系

列データが無くなった場合は前記占有情報を削除する占有情報削除手段とを有する付記１１又は１２記載のデータ管理装置。
（付記１４）
前記第二の時系列データより前に生成された前記第一の時系列データに係る参照については前記第一のレコードを参照対象とし、前記第二の時系列データより後に生成された前記第一の時系列データに係る参照については前記第二のレコードを参照対象とする参照手段を有する付記１１乃至１３いずれか一項記載のデータ管理装置。
（付記１５）
前記第一及び第二の時系列データには、複数ノード環境における各ノードを識別するノード識別子が関連付けられ、
前記参照開始手段は、他ノードに対して前記第一の時系列データの生成を要求し該要求に応じて該他ノードで生成された前記第一の時系列データを取得し、
前記更新手段は、前記他ノードからの更新要求に応じ、前記第一のレコードに対応する第二のレコードを前記データベースに生成し、該第二のレコードを更新し、
前記他ノードにおける前記更新に対するコミットに応じて、該他ノードにおいて生成された前記第二の時系列データを取得する他ノードコミット手段と、
前記他ノードコミット手段に応じて、前記他ノードより取得された前記第二の時系列データより前に生成された前記他ノードより取得された前記第一の時系列データが無くなった場合は前記第一のレコードを削除する第二の削除手段とを有する付記１１乃至１４いずれか一項記載のデータ管理装置。
【図面の簡単な説明】
【００９７】
【図１】第一の実施の形態におけるデータ管理システムの構成例を示す図である。
【図２】本発明の実施の形態におけるデータベースサーバのハードウェア構成例を示す図である。
【図３】本発明の実施の形態におけるデータベースのデータ構造の例を示す図である。
【図４】挿入トランザクションの処理手順を説明するためのフローチャートである。
【図５】更新トランザクションの処理手順を説明するためのフローチャートである。
【図６】削除トランザクションの処理手順を説明するためのフローチャートである。
【図７】操作対象とされたレコードのロック状態を示す図である。
【図８】コミット処理の処理手順を説明するためのフローチャートである。
【図９】発行データの構成例を示す図である。
【図１０】発行データに基づく並列参照処理の存否の判定内容を説明するための図である。
【図１１】ゴーストトランザクションの発生の契機の具体例を示す図である。
【図１２】参照処理の処理手順を説明するためのフローチャートである。
【図１３】ゴーストトランザクションの最後の要因となった全ての参照処理の終了の判定内容を説明するための図である。
【図１４】ゴーストトランザクションによって占有されているデータの参照対象レコードの判定処理を説明するためのフローチャートである。
【図１５】ゴーストトランザクションによって占有されているデータの参照対象レコードの判定処理の具体例を示す図である。
【図１６】第二の実施の形態におけるデータ管理システムの構成例を示す図である。
【図１７】第二の実施の形態におけるデータ管理システムの動作概要を説明するための図である。
【図１８】第二の実施の形態の処理概要を説明するための図である。
【図１９】第二の実施の形態の処理概要において各ノードで管理される発行データの遷移を示す図である。
【図２０】第二の実施の形態における参照処理の処理手順を説明するためのフローチャートである。

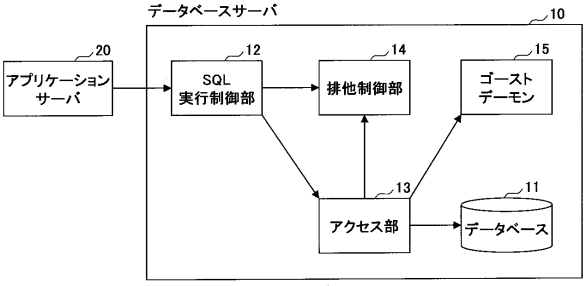
【図２１】第二の実施の形態におけるコミット処理の処理手順を説明するためのフローチャートである。

【符号の説明】

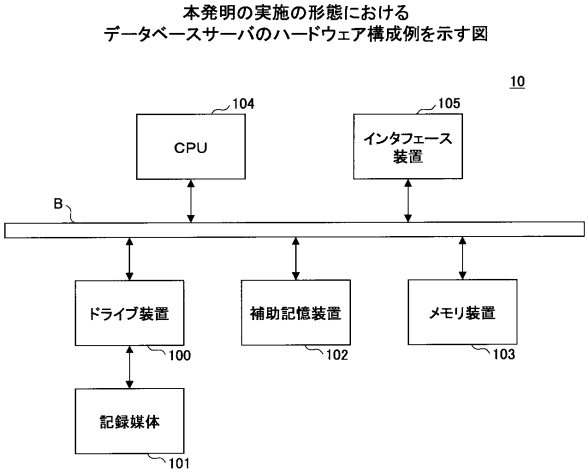
【００９８】

１０、１０ａ、１０ｂ　　データベースサーバ
１１、１１ａ、１１ｂ　　データベース
１２、１２ａ、１２ｂ　　ＳＱＬ実行制御部
１３　　アクセス部
１４　　排他制御部
１５　　ゴーストデーモン
２０　　アプリケーションサーバ
１００　　ドライブ装置
１０１　　記録媒体
１０２　　補助記憶装置
１０３　　メモリ装置
１０４　　ＣＰＵ
１０５　　インタフェース装置
Ｂ　　バス

【図１】

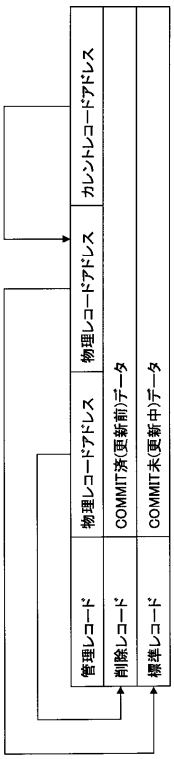


【図２】



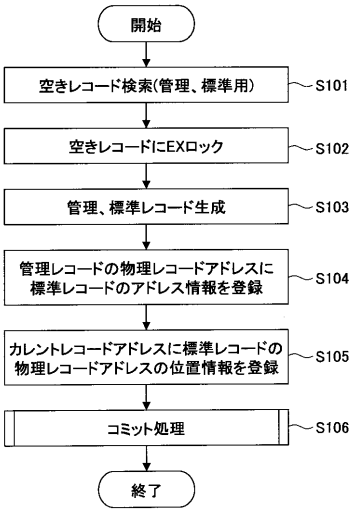
【図３】

本発明の実施の形態におけるデータベースのデータ構造の例を示す図



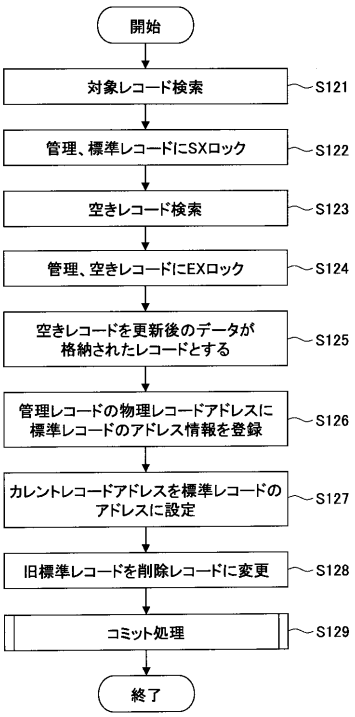
【図４】

挿入トランザクションの処理手順を説明するためのフローチャート



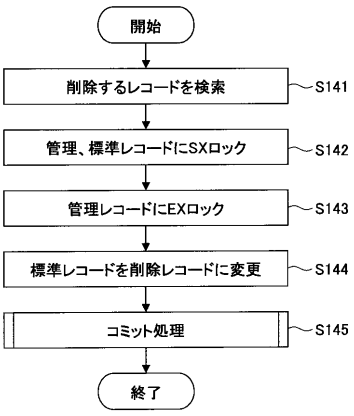
【図５】

更新トランザクションの処理手順を説明するためのフローチャート



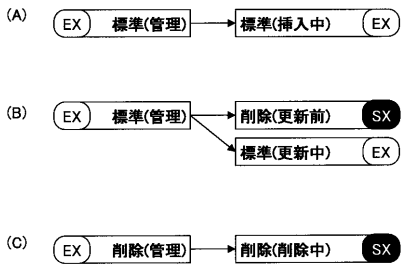
【図６】

削除トランザクションの処理手順を説明するためのフローチャート



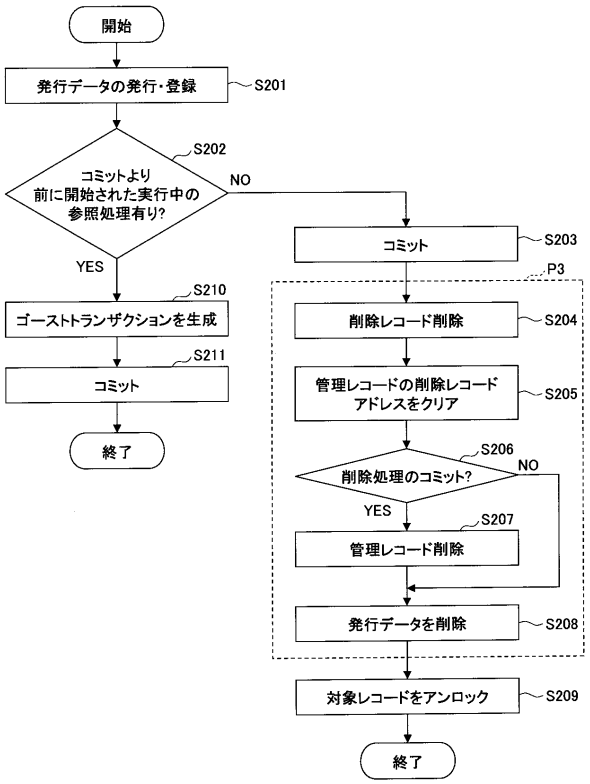
【図７】

操作対象とされたレコードのロック状態を示す図



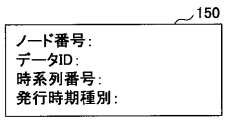
【図８】

コミット処理の処理手順を説明するためのフローチャート



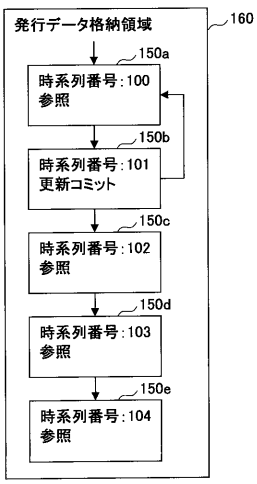
【図９】

発行データの構成例を示す図



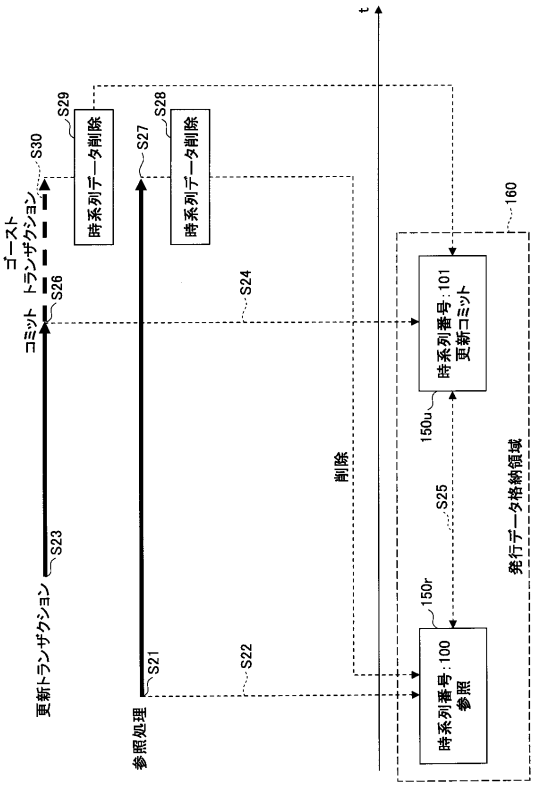
【図１０】

発行データに基づく並列参照処理の存否の判定内容を説明するための図



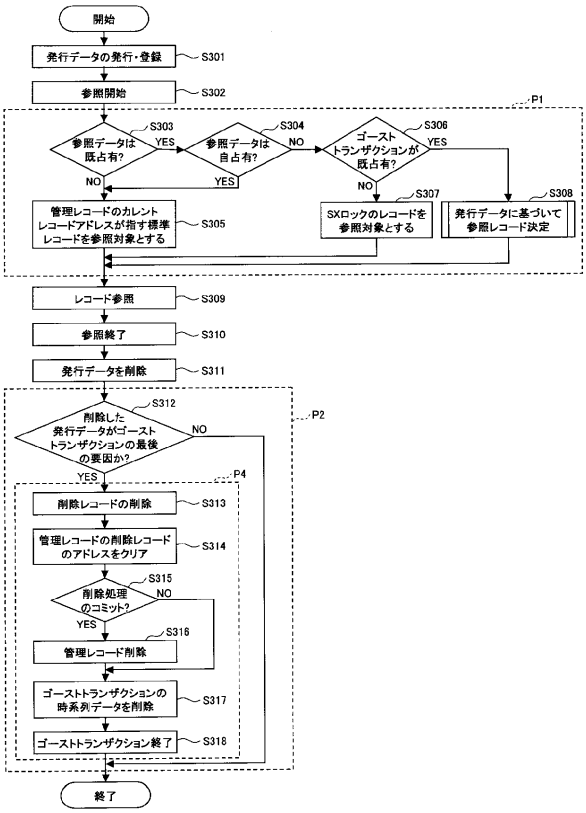
【図１１】

ゴーストトランザクションの発生の契機の具体例を示す図



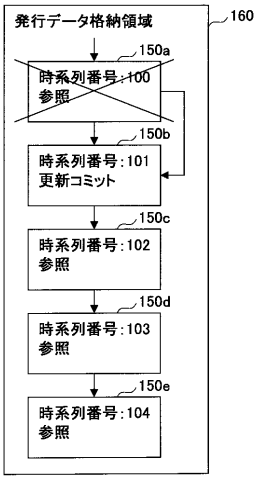
【図１２】

参照処理の処理手順を説明するためのフローチャート



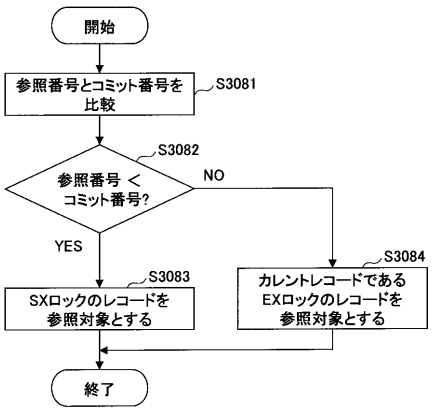
【図１３】

ゴーストトランザクションの最後の要因となった
全ての参照処理の終了の判定内容を説明するための図



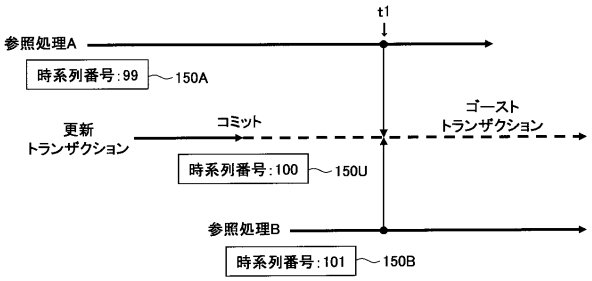
【図１４】

ゴーストトランザクションによって占有されているデータの
参照対象レコードの判定処理を説明するためのフローチャート



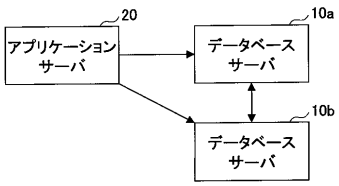
【図１５】

ゴーストトランザクションによって占有されているデータの
参照対象レコードの判定処理の具体例を示す図



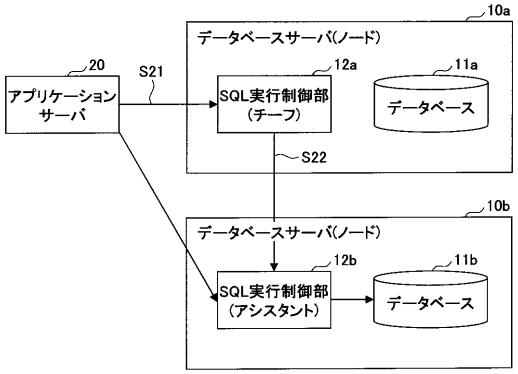
【図１６】

第二の実施の形態におけるデータ管理システムの構成例を示す図



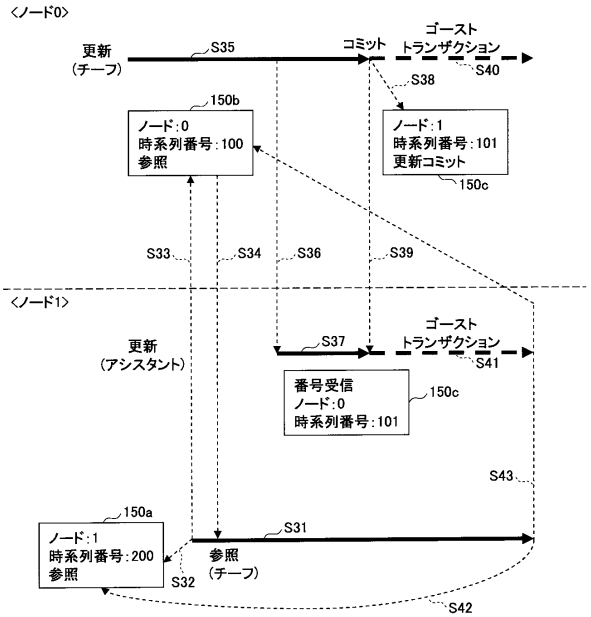
【図１７】

第二の実施の形態における
データ管理システムの動作概要を説明するための図



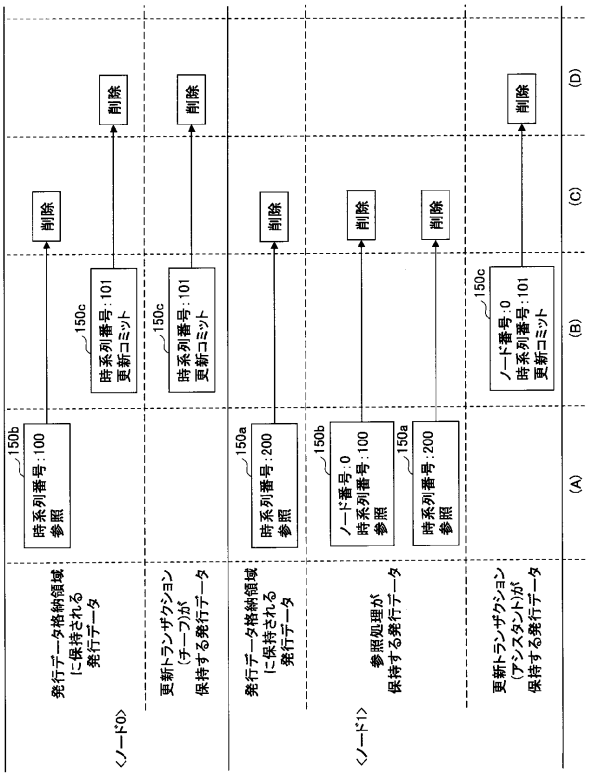
【図１８】

第二の実施の形態の処理概要を説明するための図



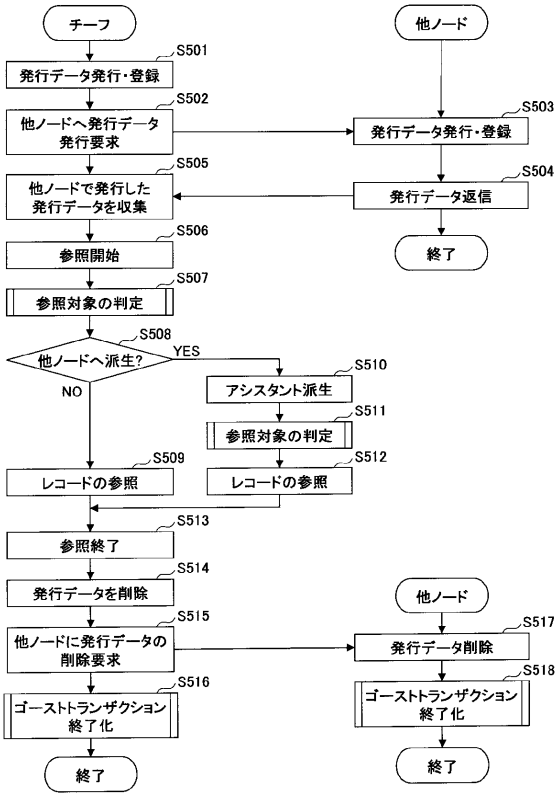
【図１９】

第二の実施の形態の処理概要において
各ノードで管理される発行データの遷移を示す図



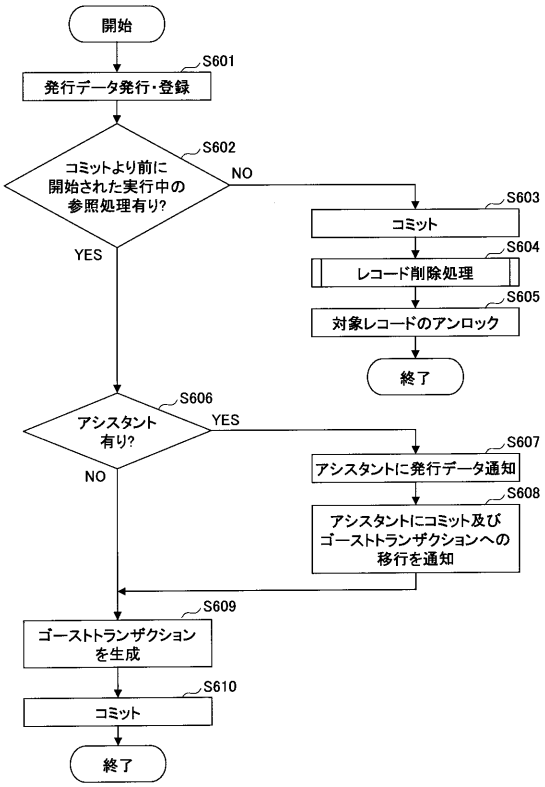
【図２０】

第二の実施の形態における
参照処理の処理手順を説明するためのフローチャート



【図２１】

第二の実施の形態における
コミット処理の処理手順を説明するためのフローチャート



フロントページの続き

(72)発明者　丹羽　雅史
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

審査官　田川　泰宏

(56)参考文献　特表２００７－５０１４６８（ＪＰ，Ａ）
特開平０８－２３５０４６（ＪＰ，Ａ）
特開２００５－２３４９４５（ＪＰ，Ａ）
特開２００３－２７１４３６（ＪＰ，Ａ）
片岡　良治，部分更新と全数検索の混在処理に適した多版並行処理制御方式，電子情報通信学会論文誌，社団法人電子情報通信学会，１９９１年　3月２５日，第J74-D-I巻　第3号，ｐ．２２４－２３１
國枝　和雄，適応型時刻印方式に基づく同時実行制御方式，情報処理学会論文誌，社団法人情報処理学会，１９９２年　6月１５日，第３３巻　第6号，ｐ．８０２－８１１
鈴木　幸市，第8回　トランザクション制御とMVCC，ＤＢ　Ｍａｇａｚｉｎｅ，株式会社翔泳社，２００４年　7月　1日，第１４巻　第4号，ｐ．１９９－２０３

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１２／００