(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4537751号
(P4537751)
(45)発行日　平成22年9月8日(2010.9.8)　(24)登録日　平成22年6月25日(2010.6.25)

(51)Int.Cl.　F I
G06F 12/00 (2006.01)　G06F 12/00 533J

請求項の数 30 (全 31 頁)

(21)出願番号　特願2004-116087(P2004-116087)
(22)出願日　平成16年4月9日(2004.4.9)
(65)公開番号　特開2004-334858(P2004-334858A)
(43)公開日　平成16年11月25日(2004.11.25)
審査請求日　平成19年2月28日(2007.2.28)
(31)優先権主張番号　10/434,647
(32)優先日　平成15年5月9日(2003.5.9)
(33)優先権主張国　米国(US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　ラム　ピー．シン
アメリカ合衆国　９８０７４　ワシントン州　サマミッシュ　２３７　アベニュー　サウスイースト　５１３

最終頁に続く

(54)【発明の名称】クライアント／サーバ環境における同期を容易にするシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
ソースレプリカ及び複数の宛先レプリカを用いてクライアント／サーバ環境における同期を容易にするシステムであって、前記クライアント／サーバ環境は、サーバ及び複数のクライアントを含み、第１のクライアントにおける第１の宛先レプリカは、第１の同期セッション中に前記サーバにおける前記ソースレプリカと同期され、他のクライアントにおける他の宛先レプリカは、前記第１の宛先レプリカが前記ソースレプリカで同期された後の同期セッション中に前記サーバにおける前記ソースレプリカと同期される前記システムは、
前記第１のクライアントから、最後の同期から変更されたデータ行を含む前記第１の宛先レプリカのパーティションを受信する、前記サーバ上の受信器構成要素と、
前記他のクライアントと同期をとる前に、前記他のクライアントのうちのどれが前記変更されたデータ行を含むパーティションの影響を受けるかを判定する、前記サーバ上の判定構成要素であって、前記サーバは、メンバーシップ情報を判定し、かつ記憶し、前記メンバーシップ情報は、前記第１の宛先レプリカと前記サーバにおける前記ソースレプリカとを同期する前記同期セッション中に変更される前記第１の宛先レプリカの行エントリのパーティションメンバーシップを特定し、変更された行についての前記メンバーシップ情報は、変更された行の行識別子と、前記変更された行を受信するように判定された前記宛先レプリカのパーティション識別子との間のマッピングを含む、前記判定構成要素と、
前記変更されたデータ行を含むパーティションの影響を受けると判定された前記他のク

ライアントの同期セッション中に、前記他のクライアントの前記他の宛先レプリカを、前記メンバーシップ情報を利用して更新する、前記サーバ上の更新構成要素と
を備えたことを特徴とするシステム。
【請求項２】
前記パーティションは、レプリケーションメタデータの形で受信され、
前記レプリケーションメタデータは、行メタデータ、生成メタデータ、パーティションメタデータ、生成パーティションマッピング、現在変更パーティションマッピング、および過去変更パーティションマッピングのうちの少なくとも１つのデータを表すテーブルを含む
ことを特徴とする請求項１に記載のシステム。
【請求項３】
前記第１のクライアントのパーティションと前記第１の宛先レプリカは、更新を受ける行が更新されるように揃えられる
ことを特徴とする請求項１に記載のシステム。
【請求項４】
前記第１のクライアントのパーティションは、更新を受ける行が、更新され、かつフィルタを用いて、結合されたテーブルから関連する行を得ることによって展開されるように、前記第１の宛先レプリカに揃えられる
ことを特徴とする請求項１に記載のシステム。
【請求項５】
前記行は、ジョインフィルタである前記フィルタを用いて展開され、展開計算は、集合ベースのクエリーに従う
ことを特徴とする請求項４に記載のシステム。
【請求項６】
前記フィルタはダイナミック行フィルタである
ことを特徴とする請求項４に記載のシステム。
【請求項７】
前記パーティションは、前記パーティションの影響を受ける前記他のクライアントと同期している間持続するメンバーシップ情報であるメタデータの宛先レプリカを含む
ことを特徴とする請求項１に記載のシステム。
【請求項８】
行削除と行更新の少なくとも一方を含む持続するメンバーシップ情報は、過去のメンバーシップに関する行情報を必要とする
ことを特徴とする請求項７に記載のシステム。
【請求項９】
前記パーティションは、最後の更新時に更新されなかった前記他のクライアントのみが更新されるように同期アンカー値に従って伝播させられる
ことを特徴とする請求項１に記載のシステム。
【請求項１０】
前記パーティションは、前記第１の宛先レプリカの行の更新、挿入、および削除の少なくとも１つが行われるときに前記パーティションの変更を処理する変更追跡論理によって追跡される
ことを特徴とする請求項１に記載のシステム。
【請求項１１】
前記他のクライアントとの同期中に前記他のクライアントの前記パーティションの変更を列挙し、前記最後の同期以後に起こった前記他のクライアントの変更を処理する変更列挙機構
をさらに備えたことを特徴とする請求項１に記載のシステム。
【請求項１２】
削除およびパーティションの変更は、同期アンカーがセッションの調停された同期アン

カーよりも新しい行を選択することによって列挙され、
前記行は、前記行の過去の変更を反映するメタデータから選択される
ことを特徴とする請求項１１に記載のシステム。
【請求項１３】
挿入およびパーティションの変更は、同期アンカーが前記セッションの調停された同期アンカーよりも新しい行を選択することによって列挙され、
前記行は、前記行の現在の変更を反映するメタデータから選択される
ことを特徴とする請求項１１に記載のシステム。
【請求項１４】
前記第１の宛先レプリカから伝播させられる変更に割り当てられる同期アンカーをさらに備え、前記同期アンカーは生成識別情報の形である
ことを特徴とする請求項１に記載のシステム。
【請求項１５】
前記それぞれのクライアントが同期するときに、第１のパーティションの前記生成識別情報に第２のパーティションの生成識別情報が上書きされる
ことを特徴とする請求項１４に記載のシステム。
【請求項１６】
ソースレプリカ及び複数の宛先レプリカを用いてクライアント／サーバ環境における同期を容易にする方法であって、前記クライアント／サーバ環境は、サーバ及び複数のクライアントを含み、第１のクライアントにおける第１の宛先レプリカは、第１の同期セッション中に前記サーバにおける前記ソースレプリカと同期され、他のクライアントにおける他の宛先レプリカは、前記第１の宛先レプリカが前記ソースレプリカで同期された後の同期セッション中に前記サーバにおける前記ソースレプリカと同期される前記方法は、
前記サーバが、前記第１のクライアントから、最後の同期から変更されたデータ行を含む前記第１の宛先レプリカのパーティションを受信するステップと、
前記サーバが、前記他のクライアントと同期をとる前に、前記他のクライアントのうちのどれが前記変更されたデータ行を含むパーティションの影響を受けるかを判定するステップであって、前記サーバは、メンバーシップ情報を判定し、かつ記憶し、前記メンバーシップ情報は、前記第１の宛先レプリカと前記サーバにおける前記ソースレプリカとを同期する前記同期セッション中に変更される前記第１の宛先レプリカの行エントリのパーティションメンバーシップを特定し、変更された行についての前記メンバーシップ情報は、変更された行の行識別子と、前記変更された行を受信するように判定された前記宛先レプリカのパーティション識別子との間のマッピングを含むステップと、
前記サーバが、前記変更されたデータ行を含むパーティションの影響を受けると判定された前記他のクライアントの同期セッション中に、前記他のクライアントの前記他の宛先レプリカを、前記メンバーシップ情報を利用して更新するステップと
を備えることを特徴とする方法。
【請求項１７】
前記パーティションは、レプリケーションメタデータの形で受信され、
前記レプリケーションメタデータは、行メタデータ、生成メタデータ、パーティションメタデータ、生成パーティションマッピング、現在変更パーティションマッピング、および過去変更パーティションマッピングのうちの少なくとも１つのデータを表すテーブルを含む
ことを特徴とする請求項１６に記載の方法。
【請求項１８】
前記サーバが、前記第１のクライアントのパーティションと前記第１の宛先レプリカを、更新を受ける行が更新されるように揃えるステップ
をさらに備えることを特徴とする請求項１６に記載の方法。
【請求項１９】
更新を受ける行が、更新され、かつフィルタを用いて結合されたテーブルから関連する

行を得ることによって展開されるように、前記サーバが、前記第１のクライアントのパーティションと前記第１の宛先レプリカとを揃えるステップ
をさらに備えることを特徴とする請求項１６に記載の方法。
【請求項２０】
前記行は、ジョインフィルタである前記フィルタを用いて展開され、展開計算は、集合ベースのクエリーに従う
ことを特徴とする請求項１９に記載の方法。
【請求項２１】
前記フィルタはダイナミック行フィルタである
ことを特徴とする請求項１９に記載の方法。
【請求項２２】
前記パーティションは、前記パーティションの影響を受ける前記他のクライアントと同期している間持続するメンバーシップ情報であるメタデータの部分レプリカを含む
ことを特徴とする請求項１６に記載の方法。
【請求項２３】
行削除と行更新の少なくとも一方を含む持続するメンバーシップ情報は、過去のメンバーシップに関する行情報を必要とする
ことを特徴とする請求項２２に記載の方法。
【請求項２４】
前記パーティションは、最後の更新時に更新されなかった前記他のクライアントのみが更新されるように同期アンカー値に従って伝播させられる
ことを特徴とする請求項１６に記載の方法。
【請求項２５】
前記パーティションは、前記第１の宛先レプリカの行の更新、挿入、および削除の少なくとも１つが行われるときに前記パーティションの変更を処理する変更追跡論理によって追跡される
ことを特徴とする請求項１６に記載の方法。
【請求項２６】
前記サーバが、前記最後の同期以後に起こった前記他のクライアントの変更を処理する変更列挙機構を用いて前記他のクライアントとの同期中に前記他のクライアントの変更を列挙するステップ
をさらに備えることを特徴とする請求項１６に記載の方法。
【請求項２７】
削除およびパーティションの変更は、同期アンカーがセッションの調停された同期アンカーよりも新しい行を選択することによって列挙され、
前記行は、前記行の過去の変更を反映するメタデータから選択される
ことを特徴とする請求項２６に記載の方法。
【請求項２８】
挿入およびパーティションの変更は、同期アンカーがセッションの調停された同期アンカーよりも新しい行を選択することによって列挙され、
前記行は、前記行の現在の変更を反映するメタデータから選択される
ことを特徴とする請求項２６に記載の方法。
【請求項２９】
前記サーバが、前記第１の宛先レプリカから伝播させられる変更に同期アンカーを割り当てるステップをさらに備え、前記同期アンカーは生成識別情報の形である
ことを特徴とする請求項１６に記載の方法。
【請求項３０】
前記それぞれのクライアントが同期するときに、第１のパーティションの前記生成識別情報に第２のパーティションの生成識別情報が上書きされる
ことを特徴とする請求項２９に記載の方法。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、クライアント／サーバ環境における同期を容易にするシステムおよび方法に関する。より詳細には、ネットワークデータアーキテクチャに関連し、特に異なるシステム間のそのようなネットワークベースのデータの更新に係る、クライアント／サーバ環境における同期を容易にするシステム、サーバベースのシステム、ネットワーク、クライアント／サーバ環境における同期を容易にする方法、記録媒体、およびデータ収集の同期を容易にするシステムに関する。
【背景技術】
【０００２】
インターネットのような世界的な通信網が出現したため、様々な地理的位置にある別々の法人実体または支社によって通常利用されるそれぞれの異なるデータベースの形で企業内で情報を広く伝播させるのが容易になっている。このような異種データ源を同種データベースとしてマージすると、多数のシステムプロセスが重複する、システムを著しく多用するプロセスが生じる。
【０００３】
それぞれの異なるデータベースは、レプリケーションによって収束させることができる。レプリケーションとは、データおよびデータベースオブジェクトをコピーし、１つのデータベースから他のデータベースに分散し、情報をデータベース間で整合するように同期させるプロセスである。
【０００４】
マージレプリケーションは困難なレプリケーションタイプである。マージレプリケーション機能は、切断されている移動ユーザがアプリケーションをオフラインで実行し、次に周期的に再接続し、メインデータベースとの同期を取るのを可能にする。これによって、ユーザが接続されているか、切断されているかにかかわらず、ソース（パブリッシャとも呼ばれる）および宛先（サブスクライバとも呼ばれる）に関する複製されたデータを自律的に変更し、次いでサイトが接続されているときにサイト間の更新をマージすることが可能になる。マージレプリケーションを用いた場合、サーバは、増分的なデータ変更をソースデータベースおよび宛先データベースに取り込み、事前に構成された規則に従ってコンフリクトを調停するか、またはカスタムリゾルバを用いることによってコンフリクトを解決する。
【０００５】
マージレプリケーションは通常、ソースおよび／または宛先での複製されたデータの自律的な変更をサポートする際に用いられる。データは、スケジューリングされた時間に、または必要に応じて各サーバ間で同期がとられる。更新は複数のサーバで独立に（たとえば、コミットプロトコルなしに）行われ、したがって、同じデータをソースまたは複数の宛先によって更新することができる。したがって、データの修正がマージされるときにコンフリクトが起こる可能性がある。マージレプリケーションは、マージ源が構成されるときに定義することのできるコンフリクト解決のためのデフォルトおよびカスタムの選択肢を含む。コンフリクトが起こると、マージエージェントはコンフリクトリゾルバを呼び出し、どのデータを許容し他の宛先サイトに伝播させるかを判定する。マージレプリケーションと共に利用可能なオプションには、ジョインフィルタおよびダイナミックフィルタの使用を含む、ソースデータの水平方向および垂直方向のフィルタリング、オルタネート同期パートナーの使用、マージ性能を向上させるための同期の最適化、同期を保証するための更新されたデータの妥当性判定、接続可能な加入データベースの使用が含まれる。
【０００６】
マージレプリケーションは水平フィルタリング、ダイナミックフィルタリング、およびジョインフィルタリングをサポートし、これらのフィルタリングはすべて、管理者が複製すべきデータのパーティション（またはテーブル）を作成するのを可能にする。複製され

たデータをフィルタリングすることによって、ネットワーク上で送信されるデータの量を最小限に抑えること、宛先レプリカで必要とされる記憶空間の量の削減、個々の宛先レプリカ要件に基づくデータ源およびアプリケーションのカスタム化、それぞれのデータパーティションをそれぞれの異なる宛先レプリカに送信できるためのコンフリクトの回避または軽減が少なくとも可能になる。マージレプリケーションは、複数のレプリカが同じデータを更新するのを可能にするにもかかわらず、レプリカが互いに疎の集合を受信するようなデータのフィルタリングによって、単一の宛先に対する２つのレプリカが同じデータ値を更新することはなくなる。

【０００７】

従来、マージレプリケーションは、ソースレプリカのパーティションを宛先レプリカに整合させる技術をサポートしている。しかし、既存のアルゴリズムは、ソースレプリカと宛先レプリカとの間の同時同期セッションを必要とすることによってシステムの顕著な性能劣化を生じさせる。従来のサーバでは、マージレプリケーションは、宛先レプリカがソースレプリカのデータの部分集合のみを受信するのを可能にする非常の精密なパーティショニング技術をサポートしている。このような技術は、ソースレプリカにおける多数のＣＰＵサイクルを必要とし、したがって、宛先データベースレプリケーションを維持するのに必要な同時同期数を増やすことによってネットワークを拡張する場合のボトルネックが生じる。ハードウェアおよびソフトウェアプロセッサを多用する関数は、宛先レプリケーションのパーティションをソースレプリケーションのパーティションに整合させるのに必要な変更のリストを作成する「パーティション計算」アルゴリズムに関連する関数である。

【０００８】

従来のパーティション計算手法の下では、クライアントがサーバとのデータベース同期を要求するとき、サーバのパーティション計算関数は、２つのデータベース間の違いを判定するために、同期のとられていない行のパーティションメンバーシップを算出することを含む。クライアントデータベースは通常、サーバ上に存在するデータベース全体の部分集合であるので、この計算関数は、場合によっては宛先レプリカに伝播させる必要のあるソースレプリカにおけるほぼすべての現在の変更を検討し、次いでこれらの変更のうちのどれが宛先レプリカに関連しているかを判定することをさらに含む。たとえば、宛先レプリカフィルタ基準を満たしている行は、宛先レプリカに属しており、宛先レプリカで更新または挿入しなければならない。更新を受けておりもはやフィルタ基準を満たさない行は、もはや宛先レプリカに属さず、したがって削除しなければならない。このことは、ソースレプリカにおける更新を削除動作として宛先レプリカに伝播できることを意味する。さらに、更新を受けており、現在宛先レプリカに属している行は、ジョインフィルタがある場合には、結合されたテーブルから関連する行を得るように展開しなければならない場合がある。

【０００９】

「同期セッション当たりパーティション計算」の性能およびスケーリングの観点から不利な点を以下に示す。従来のシステムでは冗長処理が負担になる。というのは、変更のたびに、パーティション更新があるかどうかにかかわらず、従来の同期セッションは、変更された行が宛先レプリカに属する行であるかどうかを評価する必要があるからである。行が各同期セッションの前に変更される場合、あらゆる後続の同期セッションは、この行の新しいパーティションメンバーシップメタデータを確立することによって、この行が宛先レプリカに属する行であるかどうかを再び評価する。行の前のパーティションメンバーシップに関する情報は、同期セッション間で記憶されない。このため、各同期セッションごとに冗長な処理が行われる。従来のシステムがプロセスを多用する点を例示するために、１０００個の宛先レプリカを有するネットワークを考える。ある行が、変更を受けており宛先へのパブリケーションが必要であるとソースによって判定された場合、１０００個の宛先レプリカは更新のために同期する。この行が再び変更された場合、別の１０００回の同期セッションが行われ、このプロセスは、この行が１０００回変更されるまで繰り返さ

れる。最終的に、すべての同期セッションは、同じ行に対してパーティション計算関数を合計で１００万回実行する。

【００１０】

従来のシステムは並列動作が不十分である。なぜなら、行が変更されるたびに、各宛先レプリカがそれ自体の評価を行う必要があるからである。ソースシステムと宛先システムとの間に複数の同時同期が存在する場合、パーティション計算が同時に実行される複数のインスタンスがあり、ソースサーバにおいてＣＰＵの利用度が高くなる。

【００１１】

宛先レプリカのパーティションをソースレプリカのパーティションと整合させるのに必要な変更のリストを作成するパーティション計算アルゴリズムは、所与の同期セッションで実行される関数のうちでＣＰＵおよびクエリープロセッサを最も多用する関数となる傾向がある。複製されるデータの集合を定義するのに用いられるフィルタリングが最適でない場合、並行パーティション計算関数を実行することによってソースサーバの機能はさらに劣化する。

【００１２】

より高速のハードウェアおよびより効率的なレプリケーションコンフィギュレーションは、この問題にある程度対処することができる。しかし、完全に構成され調節されたアプリケーションでも、あらゆるパブリッシャサーバは、それが有効に扱うことのできる同時サブスクライバマージプロセスの数に関する上限を有する。

【００１３】

この問題の１つの解決策は、各ソースマージプロセスが交互に実行されるように各プロセスのタイミングの調和を図ることである。現在、すべての宛先が１日の始めまたは終りに接続されて変更をマージしている場合、１日の非ピーク時にマージプロセスを実行しなければならないことがある。宛先がいつマージプロセスを実行するかが制御されていない場合、同時にマージできる宛先の数を制限する上限を定義することができる。

【００１４】

ソースサーバに対する要求を低減させる他の方法は、処理負荷をより多くのソース間に拡散することである。一般的な技術として、再発行階層で複数のサーバが使用されている。たとえば、現在、単一のサーバが国内のすべての販売代理店用のソースとして働いている場合、２つ以上のサーバを付加して、関連する負荷を分散させることができ、すなわち、中央パブリッシャがデータを東西サブスクライバに発行し、東部サブスクライバがこのデータを東海岸の販売代理店に再発行し、西部サブスクライバがデータを西海岸の販売代理店に再発行する。

【００１５】

しかし、従来の解決策のうちで、多数の同期に伴う問題に対処する費用有効で効率的なアーキテクチャを実現するものはない。従来のシステムは、不適切なメタデータをネットワーク上で伝播させる。同期セッションがパーティションを算出した後でも持続するメンバーシップメタデータはないので、ソースレプリカにおけるすべての変更は、宛先レプリカに伝播させるには不適切とみなされる。この手法はスケーリング問題に寄与する。なぜなら、多数の宛先レプリカが変更をソースレプリカに伝播させる場合、検討すべき変更のリストが大きくなるからである。さらに、ソースレプリカが宛先レプリカとの１回の同期セッション中に追加のメタデータを記録する場合、このメタデータを他の宛先レプリカに伝播させなければならないことがある。したがって、現在の手法のスケーリング特性は望ましいものとは言えない。当技術分野では、同期セッション中に検討されネットワーク上で伝播させられるメタデータの量を最小限に抑える必要がある。多数の並行同期セッションをサポートする従来技術の欠点に基づいて、あらゆる同期セッションの実行時にコストのかかるパーティション計算関数を不要にする必要がある。

【００１６】

いくつかの文献に上述のような従来の技術に関連した技術内容が開示されている（例えば、特許文献１参照）。

【００１７】
【特許文献１】米国特許第６，０９４，７１５号明細書
【発明の開示】
【発明が解決しようとする課題】
【００１８】
従来のシステムには上述したような種々の問題があり、さらなる改善が望まれている。
【００１９】
本発明は、このような状況に鑑みてなされたもので、その目的とするところは、効率的にパーティション計算を実行しデータレプリケーションの変更を伝播させることができる、クライアント／サーバ環境における同期を容易にするシステムおよび方法を提供することにある。
【課題を解決するための手段】
【００２０】
以下に、本発明のいくつかの態様を基本的に理解するために本発明の概要を示す。この概要は本発明の詳細な説明ではない。これは、本発明の重要／重大な要素を識別するためのものでも、本発明の範囲を示すものでもない。この唯一の目的は、本発明のいくつかの概念を以下に示されるより詳細な説明の序文として簡略化された形式で示すことである。
【００２１】
本発明は、パーティション計算用の新規のアーキテクチャを提供し、並行同期セッションの数をマージレプリケーションを介してスケールアップするのを可能にし、多数の並行同期セッションを必要とする構成をサポートする。本発明の一態様によれば、宛先におけるデータのパーティションをソースのデータのパーティションに整合させる新しいアルゴリズムが導入される。変更が行われると、このアーキテクチャは、簡単なクエリーの集合を用いることによってこの変更のパーティションメンバーシップが算出されるように動作する。第１の同期のパーティションメンバーシップが事前に算出されることによって、変更をソースレプリカと宛先レプリカの以後の同期セッションとの間で効率的に伝播させることができる。
【００２２】
本発明の顕著な態様は、ソースレプリカと宛先レプリカとの間のあらゆる同期セッション中に行のパーティションメンバーシップを算出するのではなく、パーティションメンバーシップを算出するペナルティが実際の行修正時に行われることである。この手法はさらに、あらゆる同期セッション中のパーティション計算を不要にすることができる。これは、パーティション情報が、事前に最初のセッション更新時に算出され、以後の同期セッションの間持続するからである。
【００２３】
したがって、ここに開示され請求される発明は、その一態様では、データレプリカを用いてクライアント／サーバ環境における同期を容易にするアーキテクチャを備える。複数のクライアントがサーバとの同期を要求すると、同期をとるクライアントとして選択された第１のクライアントは、データベース内のパーティションメンバーシップ行が、持続するメタデータを用いて算出され記録されるように処理される。第１のクライアントが更新された後、同期プロセスは、持続するメタデータによって求められる、更新の影響を受ける残りのクライアントに対して継続する。
【００２４】
本発明の新規の態様は、行が初めて変更されるときにのみ（「同期セッション当たりパーティション計算」と同様に）処理を実行できるようにし、その後は、パーティションメンバーシップ情報を持続させる。開示されるアーキテクチャとは異なり、従来のアーキテクチャは、メンバーシップ情報を「忘れ」、したがって、各同期セッションごとにこの情報を算出する必要がある。本発明の態様によれば、以後の同期セッション中に、持続するメンバーシップ情報を用いて、ソースレプリカにおける変更が宛先レプリカに伝播させられ、変更が残りの宛先に適切な変更であるかどうかが評価される。さらに、同じ行が、更

新を伴わずに複数回変更される場合（恐らく、更新に関する最も一般的な状況である）、パーティション計算は２回目以後も行われない。これは、更新が行われるまで持続するメンバーシップ情報が有効であり、したがって、以後の同期では依然として同じ持続するメンバーシップ情報を使用できるからである。

【００２５】

行が宛先において更新を受けると、その行のメンバーシップ情報が更新される。さらに、ジョインフィルタがある場合、更新を受けており現在宛先レプリカに属している行を展開し、結合テーブルから関連する行を得なければならないことがある。この計算の重要な態様は、それが簡単な集合ベースクエリーを用いて実行されることである。この集合ベースクエリーの性能は、多数のパーティションがある場合でも、変更された各行がすべてのパーティションの小さな部分集合に属するか、または最良ケースにおいて、適切に区画されたデータのうちで、厳密に１つのパーティションに属するに過ぎないかぎり非常にうまくスケーリングされる。

【００２６】

変更された行の持続するメンバーシップ情報は基本的に、変更された行の行識別子と、その行を受信する資格のある宛先レプリカのパーティションＩＤとのマッピングである。挿入および更新はパーティションメンバーシップを再評価することを必要とし、さらに、展開と呼ばれるプロセスを実行して関連する行を得ることを必要とする。すなわち、展開は、パーティションメンバーシップが変更された子行に対処する。なぜなら、親行が更新されており、そのパーティション識別子を再評価させなければならないからである。

【００２７】

削除および更新に関しては、次の同期セッション中に、行を宛先レプリカから削除する必要があるかどうかを容易に判定できるように、過去のメンバーシップに関する情報を持続させる必要がある。

【００２８】

本明細書では、前述のおよび関連する目的を実現する本発明のある例示的な態様を以下の説明および添付の図面に関連して説明する。しかし、これらの態様は、本発明の原則を使用できる様々な方法のうちのいくつかを示すものに過ぎず、本発明は、すべてのこのような態様およびその均等物を含むものである。本発明の他の利点および新規の特徴は、本発明の以下の詳細な説明を図面と一緒に検討したときに明らかになろう。

【発明の効果】

【００２９】

以上説明したように本発明によれば、効率的にパーティション計算を実行しデータレプリケーションの変更を伝播させることができる。

【発明を実施するための最良の形態】

【００３０】

以下、図面を参照して本発明を適用できる実施形態を詳細に説明する。

【００３１】

定義

この説明全体にわたって以下の用語が使用される。ここでは、本発明の様々な態様の理解を助けるためにこれらの用語の定義を示す。

【００３２】

ソースレプリカ：変更が伝播するデータセット。

【００３３】

宛先レプリカ：変更が伝播させられるデータセット。

【００３４】

部分レプリカ：ソースから宛先に発行され、ソースレプリカのデータの部分集合を含むデータセット。

【００３５】

パーティション：データの部分集合、およびデータの同じ部分集合を受信するすべての

レプリカは、同じパーティションに存在するとみなされ、同じパーティション識別子を割り当てることができる。

【００３６】

パーティションリアライメント：宛先レプリカ内の行のパーティションメンバーシップを変更させる修正。たとえば、パーティションレプリカ内の行のメンバーシップがＷＨＥＲＥ節を用いて叙述される場合、ある列を異なる値に更新するあらゆる行修正はパーティションリアライメントを構成する。たとえば、フィルタ「ｗｈｅｒｅ　ｓｔａｔｅ＝’ＷＡ’」をテーブル顧客上で使用する場合、行内の「ｓｔａｔｅ」列の値を変更すると、行のパーティションリアライメントが行われる。

【００３７】

ビフォー値：更新動作の前のデータの値

アフター値：更新動作の後のデータの値

行フィルタ：テーブルの行の部分集合をソースから宛先に発行できるようにするフィルタ。行フィルタは、クエリーのＷＨＥＲＥ節を使用し、特定の基準に基づいてパーティションに含まれる行を制限する。

【００３８】

ジョインフィルタ：１つのテーブルのフィルタがパブリケーション内の他のテーブルに基づくフィルタであるときレプリケーションフィルタの宛先でテーブル間関係を使用できるようにするフィルタ。ジョインフィルタは、同期セッション中に実施される、２つのテーブル間の関係を定義し、これは２つのテーブル間の結合を指定することに類似している。ジョインフィルタは、２つのテーブルを指定し、２つのテーブル間の関係を表す結合条件を指定する。結合条件は通常、ＴＡＢＬＥ１．ＣＯＬＵＭＮ１＝ＴＡＢＬＥ２．ＣＯＫＵＭＮ２の形をしている。

【００３９】

ダイナミックフィルタ：関数を用いて宛先レプリカから値を取り込み、この値に基づいてデータをフィルタリングする生フィルタ。このフィルタは一旦定義されるが、結果として得られる限定集合は、各宛先レプリカごとに異なり、宛先レプリカがそのニーズについてカスタム化されたデータの部分集合のみを受信するのを可能にする。

【００４０】

同期：データセットをソースレプリカおよび宛先レプリカから最終的な収束状態に収束させるプロセス。

【００４１】

同期アンカー：同期のとれていないレプリカの状態を決定するエンティティ。これは通常、最後の２つのレプリカの同期がとられた時間を示す「論理クロックエンティティ」としてモデル化される。

【００４２】

コンフリクト検出：同期中に実行され、ソースレプリカおよび宛先レプリカにおいてメタデータを問い合わせ、修正同士が衝突しないかどうかを判定するプロセス。

【００４３】

コンフリクト解決：同期中に実行され、コンフリクトが起こった後でコンフリクトの勝者および敗者を決定するプロセス。

【００４４】

次に図面を参照して本発明について説明する。図面において、同じ参照符号は全体にわたって同じ要素を指す。以下の説明では、説明の都合上、本発明を完全に理解するために多数の特定の事項について説明する。しかし、これらの特定の事項なしに本発明を実施できることは明白である。他の場合には、本発明の説明を容易にするために公知の構造および装置はブロック図の形式で示されている。

【００４５】

本出願では、語「構成要素」および「システム」は、コンピュータに関連するエンティティ、すなわち、ハードウェア、ハードウェアとソフトウェアの組合せ、ソフトウェア、

または実行中のソフトウェアを指すものである。たとえば、構成要素は、プロセッサ上で実行されるプロセス、プロセッサ、オブジェクト、実行可能ファイル、実行スレッド、プログラム、および／またはコンピュータに制限されない。一例として、サーバ上で実行されるアプリケーションとサーバの両方が構成要素であってよい。１つまたは複数の構成要素がプロセスおよび／または実行スレッド内に存在することができ、かつ構成要素は１つのコンピュータ上に配置し、かつ／または２つ以上の構成要素間に分散させることができる。

【００４６】

次に図１を参照すると、本発明のシステムブロック図が示されている。以下の解説は、システム動作における、すでにソースと複数の宛先との同期がとられている点から始まる。各宛先は、それぞれのデータベースを変更しており、次にソースとの通信を再確立しデータベースの同期を要求する。

【００４７】

サーバ（またはソース）１００は複数のＮ個のクライアント（または宛先）から同期要求を受信する。ソース１００は、同期要求を処理しすべてのソースシステム動作を制御する中央演算処理装置（ＣＰＵ）１０２を含んでいる。ＣＰＵ１０２は、複数の宛先によって利用されるすべてのデータベースエントリを記憶するソース（またはマスタ）データベース１０４とのインタフェースをとる。したがって、ソースデータベース１０４は、ソースレプリカとも呼ばれ、場合によってはＮ個の宛先に発行する必要のあるほとんどすべての情報を含んでいる。

【００４８】

Ｎ個の宛先から複数の同期要求を受信した後、ソース１００によって第１の宛先１０６が同期をとる宛先として選択される。第１の宛先１０６は、最後の同期の後で加えられた最新の変更を含む、宛先１０６に関連するすべてのデータベースエントリを記憶する第１の宛先データベース１０８を含んでいる。したがって、ソース１００のマスタデータベース１０４のデータとは異なるデータが、第１の宛先データベース１０８上に記憶されている。

【００４９】

ソース１００は、第１の宛先１０６を選択した後、宛先データベースとの前の同期セッション後に実行されたデータベース更新の集合を判定し、ソースデータベース１０４と宛先データベース１０８とのデータの違い（または変更）を確認する。ソース１００は、第１の宛先１０６に関連する変更のみを定義する、変更されたデータのパーティションを生成するパーティション計算アルゴリズム１１０を含んでいる。このパーティションは、ソース１００においてメンバーシップメタデータ１１４として持続し、前の同期プロセスの後に変更された、宛先データベース１０８の行エントリを示す。メンバーシップメタデータ１１４は、残りの宛先２．．．Ｎの以後の同期セッションに利用できるようにソース１００に記憶される。

【００５０】

第１の宛先が第１のレプリカデータ１１２によってソース１００との同期がとられた後、第１の宛先１０６の同期が完了する。

【００５１】

次に、第１の宛先１０６の更新されたデータベース情報を、マージレプリケーションによって、同期を待っている２．．．Ｎ個の宛先のうちのいくつかまたはすべてに伝播させる必要がある。本発明の一態様によれば、以後の宛先同期は、第１の宛先１０６の同期中に実行されるすべての計算動作を実行する必要はなく、第１の同期によって持続されるメンバーシップメタデータ１１４を利用する。したがって、メンバーシップメタデータ１１４は、２．．．Ｎ個の宛先にダウンロードされるレプリカを生成するのに利用される。これによって、従来のアーキテクチャと比べて、ソース１００における処理時間が著しく節約される。というのは、従来のアーキテクチャは、すべての以後の同期についてパーティションメンバーシップを再計算する必要があるからである。最も顕著な点として、パーテ

ィション計算が同期中ではなく実際の更新時に実行されるので、パーティション計算コストが多数の同期セッションに対して償却される。この技術は、多数の同期セッションがソースレプリカに対して並行して実行されているときに大きい負荷の下でよりうまくスケーリングされる。

【００５２】

動作時には、ソース１００によって、第２の宛先データベース１１８を有する第２の宛先１１６が同期をとられる宛先として選択される。ソース１００は、第２の宛先１１６に関連するフィルタ基準を得て、第２の宛先１１６のパーティションメンバーシップに基づいてどの変更が第２の宛先１１６に伝播させるのに適しているかを判定する。

【００５３】

同期プロセスは残りのＮ個の宛先について継続し、したがって、ソース１００は、メンバーシップメタデータ１１４を用いて、Ｎ個の宛先データベース１２４を有するＮ番目の宛先１２２との同期をとる。Ｎ番目の宛先１２２のフィルタ基準が得られ、分析され、メンバーシップメタデータ１１４に適用され、必要に応じてＮ番目の宛先１２２にダウンロードすべき変更の集合１２６が生成される。所与のレプリカで持続する同期メタデータがクリーンナップしても安全であると判定される（たとえば、保持ベース方式に基づくクリーンナップ、すなわち、所与の持続時間内に同期がとられないレプリカは、以後の同期を許容されない）と、対応するメンバーシップメタデータもクリーンナップしても安全であると判定される。

【００５４】

クライアント／サーバ環境に関して説明するが、本発明が、同期を必要とするあらゆる同種データ収集に適用できることに留意されたい。同様に、本発明は、ピアツーピア計算環境に適用することができる。たとえば、ソースデータ収集との同期を必要とする少なくとも２つの宛先データがある場合、データ収集をここで説明する新規の態様に従って同期させることができる。

【００５５】

次に図２を参照すると、本発明のレプリケーションプロセスのフローチャートが示されている。説明を簡単にするために、この方法を一連の動作として図示し説明することができるが、本発明が動作の順序によって制限されないことを理解されたい。というのは、本発明によれば、動作によっては異なる順序で行われ、かつ／またはここに図示し説明する動作以外の動作と並行して行われるものがあるからである。たとえば、当業者には、この方法を、状態図のように、一連の相互に関連する状態またはイベントとして表すことができることが理解されよう。さらに、本発明による方法を実施するのに例示されるすべての動作が必要なわけでない。

【００５６】

２００で、複数の宛先１．．．Ｎはソースとの同期を要求する。２０２で、ソースは、同期をとる宛先として第１の宛先を選択する。この選択プロセスは、最初に同期を要求した宛先、同期を要求している宛先の優先順位方式の利用を含むがそれに限らない、多数の方法で判定することができる。第１の宛先が選択された後、ソースは、２０４に示されているように、第１の宛先データベース内の変更行の集合を判定し、その現在の状態を確立する。２０６で、ソースは、パーティション計算アルゴリズムを用いて試験を行うことによってソースデータベースと第１の宛先データベースとの違いを判定し、残りの２．．．Ｎ個の宛先のうちの選択された宛先にどの変更を伝播させるかを判定する。２０８で、パーティション計算アルゴリズムは、１つまたは複数のメタデータテーブルの形の第１のメンバーシップメタデータを作成し、メンバーシップメタデータをソースに記憶する。２１０で、第１のパーティションレプリケーションが第１の宛先にダウンロードされ更新される。更新が完了すると、第１の宛先の同期が完了する。

【００５７】

２１２で、ソースは、同期をとるべき次の宛先を選択する。２１４で、ソースによって、次の宛先のフィルタ基準が得られ、第１の宛先のこの特定のデータの集合に関して次の

宛先に同期が必要であるかどうかが判定される。同期が必要である場合２１６で、ソースはフィルタ基準と第１のメンバーシップメタデータの両方を用いて次の宛先（または第２の宛先）用の第２のパーティションレプリカを作成する。２１８で、第２のパーティションレプリカがダウンロードされ、２２０で、パーティション更新が行われ、次の宛先についての同期プロセスのこの部分が完了する。プロセスは２１２の入力に戻り、同期をとるべき次の宛先が選択される。

【００５８】

プロセスは、すべての宛先要求同期を要求しているすべての宛先が、第１の宛先の変更された情報を受信するまで継続し、その後、同期は、第２の宛先の変更されたデータが他のすべての宛先に伝播するように実行され、以後同様にＮ個の宛先まで継続する。

【００５９】

次に図３を参照すると、宛先のメンバーシップメタデータを算出するために計算構成要素によって利用されるメタデータテーブル３００の相互関係が示されている。宛先データベースの変更情報を取り込むのに利用される主として６つのテーブルがある。パーティションメタデータテーブル３０２（PartitionsMetadataとして示されている）は他の３つのテーブル、すなわち、現在変更メタデータテーブル３０４（CurrentChangesPartitionMappingとして示されている）、過去変更メタデータテーブル３０６（PastChangesPartitionMappingとして示されている）、および生成パーティションメタデータテーブル３０８（GenerationPartitionMappingとして示されている）によってマップされている。生成パーティションマッピングテーブル３０８は生成メタデータテーブル３１０（GenerationMetadataとして示されている）にマップされ、生成メタデータテーブル３１０も行メタデータテーブル３１２（RowMetadataとして示されている）にマップされる。特定の実現形態に従って任意の適切な数のテーブルおよび／またはメタデータを使用できることを理解されたい。

【００６０】

レプリカにおける行のパーティションメンバーシップを算出するのに用いられる３つのメタデータには、宛先レプリカのパーティションを識別するそれぞれの異なるパーティション（パーティションメタデータテーブル３０２を使用する）、行の現在のパーティションメンバーシップ（現在変更パーティションマッピングテーブル３０４を使用する）、および行の過去パーティションメンバーシップ（過去変更パーティションマッピングテーブル３０６を使用する）が含まれる。現在および過去のメンバーシップ情報は、パーティション更新を他の宛先レプリカに効率的に伝播させるのを可能にする。行がすでに属しているパーティションに関する情報を維持することが好ましい。というのは、これによって「削除」を伝播できるからである。削除とは、宛先レプリカにおいてもはや必要とされないデータ（または行）である。

【００６１】

パーティションメタデータテーブル３０２は、フィルタ関数の関連する評価を追跡する。宛先レプリカがソースレプリカと同期しており、宛先レプリカを識別する異なるパーティションがパーティションテーブル３０２に存在しない場合、「パーティション値」を有する新しいエントリが作成され、新しいパーティション＿ｉｄパラメータが割り当てられる。同期をとるべき新しい宛先が始めて割り当てられると、各々のエントリは、それぞれのパーティション値と一緒にパーティションメタデータテーブル３０２に登録され、新しいパーティション＿ｉｄが割り当てられる。

【００６２】

従業員のダイナミックフィルタ表現が「ｗｈｅｒｅＴｅｒｒｉｔｏｒｙＩＤ＝ｆｎ＿ＥｍｐｌｏｙｅｅＴｅｒｒｉｔｏｒｙ（）」である場合、ｆｎ＿ＥｍｐｌｏｙｅｅＴｅｒｒｉｔｏｒｙと呼ばれる列がパーティションメタデータテーブル３０２に付加される。従業員行が変更されると、フィルタ表現を、ｆｎ＿ＥｍｐｌｏｙｅｅＴｅｒｒｉｔｏｒｙ（）がＰａｔｉｔｉｏｎｓＭｅｔａｄａｔａ．ｆｎ＿ＥｍｐｌｏｙｅｅＴｅｒｒｉｔｏｒｙで置き換えられた結合節として用いて、業員とパーティションメタデータテーブル３０２を

結合する単一の集合ベースクエリーを用いて、変更された行が属するすべてのパーティションが算出される。

【００６３】

この集合ベースクエリーの性能は、多数の登録されたパーティション＿ｉｄがある場合でも、変更された各行がすべてのパーティション＿ｉｄの小さな部分集合に属するか、または最良ケースにおいて、適切に区画されたデータのうちで、厳密に１つのパーティション＿ｉｄに属するに過ぎないかぎり非常にうまくスケーリングされる。

【００６４】

現在変更パーティションマッピングテーブル３０４は、所与の行の、それに関連するパーティションへの現在のマッピングを追跡する。したがって、テーブル３０４は、このテーブル内のパーティション＿ｉｄ列、すなわち、パーティションメタデータテーブル３０２から得られる値を含んでいる。現在変更パーティションマッピングテーブル３０４の行＿ｉｄ列は、レプリケーションによって使用される所与の行の固有の識別子を含んでいる。

【００６５】

過去変更パーティションマッピングテーブル３０６は、所与の行の、それが属している可能性があるあらゆるパーティションへのあらゆる過去のマッピングを追跡する。したがって、テーブル３０６は、パーティションテーブル３０２から得られた値であるパーティション＿ｉｄ列を含んでいる。テーブル３０６の行＿ｉｄ列は、レプリケーションによって使用される所与の行の固有の識別子を含んでいる。同期アンカー列（ｓｙｎｃｈ＿ａｎｃｈｏｒ）は論理的に、行のパーティションマッピングがいつ変更されたかに関する情報を含んでいる。パーティション更新中に同期アンカーを取り込むことによって、同期プロセスは、ソース１００からのパーティション更新を、最後にソースと宛先の同期がとられてからこれらの変更を受信していない宛先にのみ伝播させることができる。サンプルデータでは、ｓｙｎｃ＿ａｎｃｈｏｒ列は、説明を簡単にするために、ＵＴＣ時間（協定世界時）またはグリニッジ標準時間値のフォーマットの値を利用する。

【００６６】

生成パーティションテーブル３０８は、一群の変更に割り当てられる同期アンカーである生成＿ｉｄ列を含む。変更が宛先からソース１００に伝播すると、ソース１００で変更に新しい生成＿ｉｄが割り当てられる。この生成の一部である行は異なるパーティションＰ１に属するので、生成パーティションマッピングテーブル３０８は、宛先のパーティション＿ｉｄＰ１にマッピングすべきパーティション＿ｉｄ列を含んでいる。しかし、一般に、生成は複数のパーティション識別子にマップすることができることに留意されたい。これは、パーティション＿ｉｄについて特殊な値「－１」を有する生成とは異なる。他の異なるパーティションＰ２に属する異なる宛先レプリカがソース１００と同期すると、生成マッピングはＰ１個の生成を除去する。パーティション＿ｉｄについての特殊な値－１は、生成がグローバルであり、したがってすべてのパーティションに適切であることを示している。

【００６７】

生成テーブル３１０は、各生成に固有の生成ＩＤを割り当てる生成＿ｉｄ列を含んでいる。生成パーティションテーブル３０８は、このテーブル３１０をマップする生成＿ｉｄを含み、現在のレプリカにどの生成が伝播したか、またどの生成がローカル変更を表しているかを追跡する。生成パーティションテーブル３０８は、同期プロセスが、所与のセッションについて検討する必要のある関連する生成のリストを得るのを可能にする。

【００６８】

行メタデータテーブル３１２は、行ごとにレプリケーションメタデータを追跡し、論理クロックを用いて時間を表すことによって、行がいつ変更されたかに関する情報を含んでいる。テーブル３１２はまた、行のこのバージョンにどのレプリカが寄与したかに関する情報を、各列の現在のバージョンに関する情報と一緒に含んでいる。生成パーティションテーブル３０８と行メタデータテーブル３１２はどちらも、生成テーブル３１０にマップ

される。

【００６９】

以下に、それぞれの異なるメタデータテーブルおよびそれぞれの関数を概略的に示すテーブル１を示す。

【００７０】

【表１】

テーブル１．メタデータテーブルおよびその概要

メタデータテーブル名	目的
RowMetadata	レプリカメタデータを行ごとに追跡し、（論理クロックを用いて時間を表すことによって）行がいつ変更されたかに関する情報を含むと共に、行および列バージョンベクトルを含んでいる。
GenerationMetadata	どの生成が現在のレプリカに伝播したか、またどの生成がローカル変更を表しているかを追跡し、同期プロセスが、その所与のセッションについて検討する必要のある関連する生成のリストを得るのを可能にする。
PartitionsMetadata	宛先レプリカを表すあらゆるパーティション値について異なるエントリを含む。
GenerationPartitionMapping	GenerationMetadataテーブル内の生成のうちのどれがどのパーティションに関連するかに関するマッピング情報を含む。
CurrentChangesPartitionMapping	現在どの変更がどのパーティションに関連しているかに関するメタデータを含む。
PastChangesPartitionMapping	前にどの変更がどのパーティションに関連していたかに関するメタデータを含む。

【００７１】

変更を効率的に伝搬させるための生成の区分

本発明の新規の態様はまた、変更の伝播中に使用される最適化を容易にする。マージレプリケーションは現在、「生成」の概念を用いて、ソースレプリカから宛先レプリカに伝播する変更を論理的にグループ分けする。生成メタデータテーブル３１０は、どの生成が現在のレプリカに伝播したか、またどの生成がローカル変更を表しているかを追跡する。生成区分はまた、同期プロセスが、所与のセッションについて検討する必要のある関連する生成のリストを得るのを可能にする。

【００７２】

ソースレプリカにおいて変更が加えられると、ソースは、生成値、すなわち論理クロックエンティティをテーブル上の変更の集合に割り当てる。このグループ分け概念は、割り込まれた可能性のある前の同期セッションから、または異なるソースレプリカを有する同期セッションを介して、一群の変更が受信されているかどうかを、宛先レプリカが効率的に識別するのを可能にする。基本的に、現在宛先レプリカに存在しない生成値のリストは、ソースから宛先に伝播させるべき変更とみなすのが適切な変更を反映している。宛先レプリカがソースレプリカからデータの部分集合、たとえば別の宛先レプリカを受信した場合、このレプリカのパーティション基準を満たす変更は、この宛先に伝播する変更だけである。

【００７３】

しかし、マージレプリケーションの現在のバージョンは依然として、ソースレプリカに存在するすべての生成に関する情報を、これらの生成が宛先レプリカに関連する変更を含

むかどうかにかかわらず、宛先レプリカに伝搬させる必要がある。これは、ソースレプリカにある生成の一部である変更がこのパーティションに関連しているか、それとも関連していないかを追跡するメタデータがないからである。

【００７４】

各行のパーティションメンバーシップを識別するパーティション群は、ソースレプリカから宛先レプリカへの生成の伝播を効率的なものにする。変更の集合がすでにパーティション識別子にマップされているので、行のグループ分けである生成もパーティション識別子にマップすることができる。したがって、ソースレプリカで利用可能な特定のパーティションにおいて宛先レプリカが重要であるとき、宛先レプリカに関連するパーティションの変更を含む生成値は、関連する変更の集合を算出する際に迅速に除去することができる。このアルゴリズムは、計算効率を高めるだけでなく、ネットワーク性能特性も向上させる。これは、宛先パーティションに関連する生成のみがネットワーク上で伝播するからである。

【００７５】

次に図４を参照すると、フィルタリングおよび展開を利用するサンプル更新スキーマが示されている。この例は、従業員が配置替えされることを前提条件としている。更新プロセスは、配置替えされる従業員の顧客データを別の従業員に割り当て直すことを含む。この例では、クライアント情報は、少なくとも顧客情報、顧客注文情報、および顧客注文詳細を含む。パーティションテーブルは、従業員ＩＤパラメータを有する配置替え従業員を一意に識別する従業員パーティションテーブル４００である。この例では、テーブル４００は、少なくとも３つの列、すなわち名前列、名字列、および受持ち区域ＩＤ列を含んでいる。従業員パーティションのダイナミックジョインフィルタ表現が「ｗｈｅｒｅＴｅｒｒｉｔｏｒｙＩＤ＝ｆｎ＿ＥｍｐｌｏｙｅｅＴｅｒｒｉｔｏｒｙ（）」である場合、ｆｎ＿ＥｍｐｌｏｙｅｅＴｅｒｒｉｔｏｒｙと呼ばれる列が従業員パーティションメタデータテーブル４００に付加される。

【００７６】

新しいパーティションが到着して初めて同期をとられると、これらの新しいパーティションは、それぞれのパーティション値と一緒にエントリをこのテーブル４００に登録し、新しいパーティション＿ｉｄが割り当てられる。従業員テーブル４００の行が変更されると、フィルタ表現を結合節として用いて各従業員を従業員パーティションメタデータテーブル４００に結合する単一の集合ベースクエリーを用いて、変更された行が属するすべてのパーティションが算出される。次いで、関数呼出しｆｎ＿ＥｍｐｌｏｙｅｅＴｅｒｒｉｔｏｒｙ（）がＰａｒｔｉｔｉｏｎｓＭｅｔａｄａｔａ．ｆｎ＿ＥｍｐｌｏｙｅｅＴｅｒｒｉｔｏｒｙで置き換えられる。

【００７７】

たとえば、従業員パーティションテーブル４００内の受持ち区域ＩＤを米国内の５１の異なる地理的領域として評価する場合、従業員パーティションテーブル４００は、この５１個の異なる値のそれぞれに固有のパーティションＩＤを割り当てる。宛先がソース１００と同期しており、宛先レプリカを識別する異なるパーティションが従業員テーブル４００内に存在しない場合、異なる「パーティション値」を有する新しいエントリが作成され、新しいパーティション＿ｉｄが割り当てられる。

【００７８】

以下のテーブル（テーブル２）は、従業員の姓名の列を有さない従業員パーティションメタデータテーブル４００のサンプルである。最後の列は、サンプルデータの解釈を示しているが、スキーマの必要な部分ではない。

【００７９】

【表２】

テーブル２　サンプルPartitionsMetadataテーブル

パーティション_id	fn_EmployeeTerritory	テーブルの解釈 （スキーマの一部ではない）
1	「WA」	パーティションID=1はワシントン地区に対応する。
2	「CA」	パーティションID=2はカリフォルニア地区に対応する。
3	「OR」	パーティションID=3はオレゴン地区に対応する。
...	...	...

【００８０】

この集合ベースクエリーの性能は、多数のパーティションｉｄが登録されている場合でも、それぞれの変更された行がすべてのパーティション＿ｉｄの小さな部分集合に属するか、または最良ケースにおいて、適切に区画されたデータのうちの、厳密に１つのパーティション＿ｉｄに属するに過ぎないかぎり非常にうまくスケーリングされる。

【００８１】

上記で指摘したように、すべてのクライアント情報を従業員と一緒に伝播させる必要がある。ジョインフィルタは、同期プロセス中に使用されるテーブル間関係を定義することによってこのプロセスを容易にするのに用いられる。サンプルスキーマでは、従業員テーブル４００内の行のパーティションメンバーシップは、従業員テーブル４００上の「行フィルタ」定義を用いて叙述される。顧客テーブル４０４内の行のパーティションメンバーシップは、従業員と顧客との間のジョインフィルタ定義４０２を用いて叙述された、従業員テーブル４００内の行のメンバーシップに基づくメンバーシップである。注文テーブル４０８内の行のメンバーシップは、顧客と注文との間のジョインフィルタ定義を用いて叙述された、顧客テーブル４０４内の行のメンバーシップに基づくメンバーシップである。同様に、注文詳細テーブル４１２内の行のメンバーシップは、注文と注文詳細との間のジョインフィルタ定義を用いて叙述された、注文テーブル４０８内の行のメンバーシップに基づくメンバーシップである。

【００８２】

したがって、第１のジョインフィルタ４０２は、従業員テーブル４００と顧客テーブル４０４とのテーブル間関係（顧客.従業員ＩＤ＝従業員.従業員ＩＤ）を定義する。顧客テーブル４０４は、顧客を一意に識別する顧客ＩＤに関連付けされており、従業員テーブル４００にマップされる従業員ＩＤ列と、名前、住所、郵便番号、電話番号、その他の情報などの顧客会計情報に関連するその他の列とを少なくとも含んでいる。第２のジョインフィルタ４０６は、顧客テーブル４０４と注文テーブル４０８とのテーブル関係（注文・顧客ＩＤ＝顧客.顧客ＩＤ）を定義する。注文テーブル４０８は、顧客テーブル４０４と共に注文情報を一意に識別する注文ＩＤに関連付けされており、顧客テーブル４０４にマッピングされる顧客ＩＤ列を少なくとも含んでいる。テーブル４０８は、出荷情報、税率、運賃のような、顧客注文に関連する情報の、その他の列を含んでいる。

【００８３】

第３のジョインフィルタ４１０は、注文テーブル４０８と注文詳細テーブル４１２とのテーブル関係（注文詳細.注文ＩＤ＝注文.注文ＩＤ）を定義する。注文詳細テーブル４１

２は、注文テーブル４０８と共に注文詳細情報を一意に識別する注文詳細ＩＤに関連付けされており、注文テーブル４０８にマッピングされる注文ＩＤ列を少なくとも含んでいる。テーブル４１２は、注文詳細テーブル４１２を製品テーブル４１６にマッピングする製品ＩＤ列を含んでいる。テーブル４１２は、注文された製品に関する情報を提供する数量列および単位価格も含んでいる。したがって、注文詳細テーブル４１２は製品テーブル４１６の単位価格情報を必要とする。

【００８４】

第４のジョインフィルタ４１６は、注文詳細テーブル４１２と製品テーブル４１６とのテーブル関係（注文詳細.製品ＩＤ＝製品.製品ＩＤ）を定義する。製品テーブル４１６は固有の製品ＩＤに関連付けされており、製品名および単位価格の列をさらに含んでいる。

【００８５】

したがって、展開アルゴリズムはジョインフィルタと共に、クライアントが購入した製品、購入された製品の注文詳細、その詳細および特定の製品を含む注文、その製品を注文した顧客に関するすべての情報が従業員ＩＤと一緒に伝搬するのを容易にする。

【００８６】

以下に図５に示されているサンプルデータを使用した場合、顧客テーブル４０４に対するこのパーティション更新によって、顧客行に関する過去および現在のパーティションマッピングが再計算される。次いで、顧客テーブル４０４と注文テーブル４０８との間のジョインフィルタ４０６に続いて、親パーティションメンバーシップが変更されたすべての注文行のそれぞれの、過去および現在のパーティションマッピングが再評価される。

【００８７】

注文テーブル４０８と注文詳細テーブル４１２との間のジョインフィルタ４１０に続いて、親パーティションメンバーシップが変更されたすべての注文詳細行のそれぞれの、過去および現在のパーティションマッピングが再評価される。最後に、注文詳細テーブル４１２と製品テーブル４１６との間のジョインフィルタ４１４に続いて、親パーティションメンバーシップが変更されたすべての製品行のそれぞれの、過去および現在のパーティションマッピングが再評価される。次いで、製品テーブル４１６に子行がなくなったため、アルゴリズムは終了する。展開アルゴリズムが終了すると、メタデータテーブルは行のパーティションメンバーシップを正確に反映する。行がソースレプリカから宛先レプリカに伝播するまで、もはやパーティション計算は必要とされない。

【００８８】

このことを反映するために、子テーブルに対して生成されるデータベースビューは、中間的な親に対して生成されるビューを参照する。サンプルスキーマでは、顧客に関するビューは従業員に関するビューを参照する。同様に、注文に関するビューは顧客に関するビューを参照する。同様に、注文詳細に関するビューは注文に関するビューを参照する。最上位の親、すなわち、この例では従業員テーブル４００に関するビューは、パーティションメタデータテーブル３０２を用い、さらにサンプル内のフィルタリング列（受持ち区域ＩＤ）の値を用いて行のパーティションメンバーシップを評価する。

【００８９】

一例として、以下のビュー定義がサンプルスキーマに使用される。

【００９０】

従業員テーブル用のビュー定義（ビュー＿パーティション＿従業員）

```
select[Employees].*,partition_id=[PartitionsMetadata].[partition_id]from
Employees,PartitionsMetadata where PartitionsMetadata.fn_EmployeeTerritory=
Employees.TerritoryID
```

【００９１】

顧客テーブル用のビュー定義（ビュー＿パーティション＿顧客）

```
select[Customers].*,partition_id=[Employees].[partition_id] from Customers,
[view_partition_Employees]Employees where Customers.EmployeeID=
Employees.EmployeeID
```

【００９２】

注文テーブル用のビュー定義（ビュー＿パーティション＿注文）

```
select[Orders].*,partition_id=[Custorners].partition_id from[Orders],
[view_partition_Customers][Customers]where(Orders.CustomerID=
Customers.CustomerID)
```

【００９３】

注文詳細テーブル用のビュー定義（ビュー＿パーティション＿注文詳細）

```
select[Order_Details).*,partition_id=[Orders].partition_id from
[Order Details],[view_partition_Orders][Orders]where(Order_Details.OrderID
=Orders.OrderID)
```

【００９４】

子行のパーティションメンバーシップは、変更された行の集合に関するビューを通じて行の変更された集合を選択することによって判定される。「ビフォー（before）」値から算出されたすべての過去のパーティションメンバーシップは、過去変更パーティションマッピングテーブル３０６において持続し、すべての現在のパーティションマッピングは現在変更パーティションマッピングテーブル３０４において持続する。

【００９５】

以下のテーブル（テーブル３）は、同期更新の一部として、従業員ＪｏｅがＣＡ地区からＷＡ地区へ配置替えされる変更を反映するサンプル現在変更パーティションマッピングテーブル３０４である。最後の列は、解釈のためのみのものであり、スキーマの一部として必要なものではない。

【００９６】

【表３】

テーブル３　サンプルCurrentChangesPartitionMappingテーブル

行_id	パーティション_id	テーブルの解釈 （スキーマの一部ではない）
R1	1	行R1は受持ち区域が「WA」である従業員「Joe」に対応する。
R2	2	行R2は受持ち区域が「CA」である従業員「May」に対応する。
R3	3	行R3は受持ち区域が「OR」である従業員「Jane」に対応する。
...	...	...

【００９７】

以下にテーブル４、すなわち、Ｊｏｅが以前ＣＡ地区担当であったことを示すサンプル過去変更パーティションマッピングテーブル３０６を示す。最後の列は、解釈のためのみのものであり、スキーマの一部として必要なものではない。

【００９８】

【表４】

テーブル４　サンプルPastChangesPartitionMappingテーブル

行_id	パーティション_id	同期アンカー	テーブルの解釈 （スキーマの一部ではない）
R1	2	2002年10月8日 火曜日 午前10時27分	行R1は、受持ち区域が2002年 10月8日火曜日午前10時27分まで 「CA」であった従業員「Joe」に 対応する。
R2	3	2002年10月7日 月曜日 午前12時25分	行R2は、受持ち区域が2002年 10月7日月曜日午前12時25分まで 「OR」であった従業員「Mary」に 対応する。
...	...	...	...

【００９９】

次に、図５を参照すると、顧客データがある従業員から別の従業員に割り当て直されているパーティション更新の例が示されている。既存の例示されているテーブル関係は、「Ｊｏｅ」の従業員ＩＤについては以下のとおりである。従業員１テーブル５００は、「Ｊｏｅ」を示す従業員ＩＤ列と、従業員１テーブル５００を従業員パーティションメタデータテーブル４００にリンクする「ＷＡ」のエントリを有する受持ち区域ＩＤ列とを有する従業員１行を表している。従業員１テーブル５００の子としての顧客テーブル５０２は、顧客１行を、顧客ＩＤ列エントリ「Ａｌｆｒｅｄ」を有する行として表し、かつ従業員ＩＤエントリを、顧客１テーブル５０２を従業員１テーブル５００にリンクする「Ｊｏｅ」として表している。

【０１００】

顧客１テーブル５０２は、３つの子エントリ、すなわち第１の注文＿１エントリ５０４、第２の注文＿２エントリ５０６、および第３の注文＿３エントリ５０８を有する。注文＿１エントリ５０４は、エントリ「１」を持つ注文ＩＤ列と、エンティティ５０４を親顧客１テーブル５０２にリンクするエントリ「Ａｌｆｒｅｄ」を持つ顧客ＩＤ列とを有する行として注文＿１行を表している。注文＿２エントリ５０６は、エントリ「２」を持つ注文ＩＤ列と、エンティティ５０６を親顧客１テーブル５０２にリンクするエントリ「Ａｌｆｒｅｄ」を持つ顧客ＩＤ列とを有する行として注文＿２行を表している。注文＿３エントリ５０８は、エントリ「３」を持つ注文ＩＤ列と、テーブル５０８を親顧客１テーブル５０２にリンクするエントリ「Ａｌｆｒｅｄ」を持つ顧客ＩＤ列とを有する行として注文＿３行を表している。

【０１０１】

第１の注文＿１エンティティ５０４は、その特定の注文ＩＤ＝１の注文詳細を定義する４つの子注文詳細エンティティを有している。注文詳細１エンティティ５１０は、行注文詳細１を、エンティティ５１０を親エンティティ５０４にリンクするエントリ「１」を持つ注文ＩＤ列と、図示されていない他の列詳細とを有する行として表している。注文詳細２エンティティ５１２は、行注文詳細２を、エンティティ５１２を親エンティティ５０４にリンクするエントリ「１」を持つ注文ＩＤ列と、図示されていない他の列詳細とを有する行として表している。注文詳細３エンティティ５１４は、行注文詳細３を、エンティティ５１４を親エンティティ５０４にリンクするエントリ「１」を持つ注文ＩＤ列と、図示されていない他の列詳細とを有する行として表している。注文詳細４エンティティ５１６は、行注文詳細４を、エンティティ５１６を親エンティティ５０４にリンクするエントリ「１」を持つ注文ＩＤ列と、図示されていない他の列詳細とを有する行として表している。

【０１０２】

注文＿２エンティティ５０６は、注文詳細５行を、エンティティ５１８を親エンティティ５０６にリンクするエントリ「２」を持つ注文ＩＤ列と、図示されていない他の列詳細とを有する行として表している１つの子エントリ、すなわち、注文詳細５エントリ５１８を有している。

【０１０３】

注文＿３エンティティ５０８は、２つの子エンティティ、すなわち、注文詳細６エンティティ５２０および注文詳細７エンティティ５２２を有している。注文詳細６エンティティ５２０は、注文詳細６行を、エンティティ５２０を親エンティティ５０８にリンクするエントリ「３」を持つ注文ＩＤ列と、図示されていない他の列詳細とを有する行として表している。注文詳細７エンティティ５２２は、注文詳細７行を、エンティティ５２２を親エンティティ５０８にリンクするエントリ「３」を持つ注文ＩＤ列と、図示されていない他の列詳細とを有する行として表している。

【０１０４】

顧客１テーブル５０２内の顧客１行の従業員ＩＤ列は、「Ｊｏｅ」から「Ｍａｒｙ」に更新されているところである。従業員２テーブル５２４は、従業員２行を、エントリ「Ｍａｒｙ」を持つ列と、エントリ「ＣＡ」を持つ受持ち区域ＩＤ列とを有する従業員２行を表している。この更新では、基本的に顧客＿１行パーティションメンバーシップが「Ｊｏｅ」から「Ｍａｒｙ」に変更されている。したがって、注文行、すなわち注文＿１、注文＿２、注文＿３のパーティションメンバーシップも「Ｊｏｅ」から「Ｍａｒｙ」に変更される。同様に、注文詳細エンティティ５１０、５１２、５１４、５１６、５１８、５２０、および５２２内の対応する注文詳細行も今や、異なるパーティション、すなわち「Ｍａｒｙ」に属している。各行のパーティションメンバーシップを親エンティティから子エンティティに伝播させるために、展開プロセスが実行される。

【０１０５】

レプリケーション処理－変更追跡および変更列挙

影響を受けるレプリケーション処理の２つの主要な段階には、変更追跡と変更列挙が含まれる。ユーザデータベースでデータが変更されると、ユーザデータベース内の変更追跡機構は、後でこれらの変更と他のレプリカの同期をとるのを可能にするレプリケーションメタデータを追跡する。変更列挙は、ソースレプリカと宛先レプリカとの前の同期以来このレプリカで起こった変更を列挙する同期プロセスの段階である。

【０１０６】

レプリケーション処理の変更調停段階および変更適用段階は有意の影響を受けない。

【０１０７】

次に、行が挿入、更新、または削除されるときのレプリケーションメタデータの修正に関して、変更追跡機構について説明する。行が挿入される際、その行の現在のパーティションメンバーシップが評価され、この情報が現在変更パーティションマッピングテーブル３０４において持続する。この所与の行の子行がすでに存在するいくつかのまれな場合には、新しい行を挿入すると、展開プロセスを用いて子行のパーティションメンバーシップも評価される。

【０１０８】

ある行が非フィルタ列更新によって更新される際、この所与の行に関するメタデータがメタデータテーブル内にない場合、この行の現在のパーティションメンバーシップが評価され、この情報が現在変更パーティションマッピングテーブル３０４において持続する。パーティションメンバーシップはメタデータがすでに存在する場合には評価されない。なぜなら、メタデータが存在することは、パーティションメンバーシップがすでに評価されていることを示すからである。

【０１０９】

ある行が列フィルタリング更新によって更新される際、その行が過去に属していたパーティションは、更新の前に、変更追跡中に得られる「ビフォー」値を用いて評価される。子行のパーティションメンバーシップは親行のパーティションメンバーシップの影響を受

けるので、展開プロセスを用いて子行の過去のパーティションメンバーシップが評価される。すべての過去のパーティションメンバーシップは過去変更パーティションマッピングテーブル３０６に記憶される。さらに、パーティション更新が行われる論理時間が過去変化テーブル３０６の同期アンカー列に記録される。これによって、同期プロセスは、パーティション更新の伝播を、すでに前の同期セッション中にまたは異なるソースレプリカとの同期を介してこの変更を見ているレプリカまでに減衰させることができる。過去パーティションマッピングを評価すると、基本的に、現在変更パーティションマッピングテーブル３０４内のあらゆるエントリがクリーンナップされることに留意されたい。これは、これらのエントリがパーティション更新によって無効にされているからである。

【０１１０】

ある行が列フィルタリング更新によって更新される際、その行が現在属しているパーティションは、更新後に、変更追跡中に得られる「アフター」値を用いて評価される。子行のパーティションメンバーシップは親行のパーティションメンバーシップの影響を受けるので、展開プロセスを用いて子行の現在のパーティションメンバーシップが評価される。すべての現在のパーティションメンバーシップは現在変更パーティションマッピングテーブル３０４に記憶される。

【０１１１】

削除を処理する際、行が削除の前に属していたパーティションは、変更追跡中に得られる「ビフォー」値を用いて評価される。子行のパーティションメンバーシップは親行のパーティションメンバーシップの影響を受けるので、展開プロセスを用いて子行の過去のパーティションメンバーシップが評価される。すべての過去のパーティションメンバーシップは過去変更パーティションマッピングテーブル３０６に記憶される。さらに、パーティション更新が行われる論理時間がテーブル３０６の同期アンカー列に記録される。これによって、同期プロセスは、パーティション更新を、すでに前の同期セッション中にこの変更を見ているレプリカまで、パーティション更新を冗長に伝播させることができる。行およびその子は削除後このパーティションには存在しないので、現在変更パーティションマッピングテーブル３０４内のエントリを作成する必要はない。過去パーティションマッピングを評価すると、基本的に、現在変更パーティションマッピングテーブル３０４内のあらゆるエントリがクリーンナップされることに留意されたい。これは、これらのエントリが行削除プロセスによって無効にされているからである。

【０１１２】

パーティションを更新する場合、変更追跡機構は、更新が宛先レプリカまで効率的に伝播するようにメタデータを更新する必要がある。重要な点は、行のパーティションメンバーシップが変更される際、行のパーティション＿ｉｄを更新しなければならず、かつ親行が更新されたためにパーティションメンバーシップが変更された子行はそのパーティションｉｄを再評価してもらわなければならないことである。この情報は、ここに示される例で利用される展開プロセスを用いて取り込まれる。

【０１１３】

変更列挙機構は、パーティションメンバーシップメタデータを用いて変更をソースレプリカから宛先レプリカまで効率的に伝播させる。変更追跡機構はすでにパーティションを評価し展開を実行しているので、実行時の変更列挙の複雑さは大幅に簡略化される。

【０１１４】

削除およびパーティション更新は、「過去パーティションマッピングエントリ」を有する変更の集合に寄与する。これらの変更の集合は、同期アンカーがこのセッションの調停された同期アンカーよりも新しく、パーティション＿ｉｄが宛先レプリカのパーティション＿ｉｄに一致する行を過去変更パーティションマッピングテーブル３０６から選択することによって列挙される。次いで、これらの行は、削除として宛先レプリカに伝播させられる。

【０１１５】

区分された挿入および更新と、区分されていない挿入および更新は、「現在パーティシ

ョンマッピングエントリ」を有する変更の集合に寄与する。変更のこれらの集合は、同期アンカーがこのセッションに関して調停された同期アンカーよりも新しく、かつ宛先レプリカのパーティション＿ｉｄに一致するパーティション＿ｉｄを有するエントリを現在変更パーティションマッピングテーブル３０４内に有する行を、行メタデータテーブル３１２から選択することによって列挙される。これらの行は、宛先レプリカへの更新として伝播する。

【０１１６】

所与のパーティションに対する変更が列挙されると、これらの変更は、コンフリクト検出解決機構によって既存の技術を用いて伝播させられる。同様に、所与のパーティションに対する変更が列挙され、コンフリクトが検出され解決されると、これらの変更は変更適用機構によって既存の技術を用いて伝播させられる。

【０１１７】

生成区分

パーティションへの行のマッピングがソースレプリカで維持されるため、ソースレプリカと宛先レプリカとの間で伝播させられる関連する生成のリストを効率的に算出することができる。たとえば、生成パーティションマッピングテーブル３０８は、列生成＿ｉｄ、すなわち一群の生成に割り当てられた同期アンカーを含んでいる。変更が宛先レプリカからソースレプリカまで伝播する際、これらの変更にはソースレプリカで新しい生成＿ｉｄが割り当てられる。この生成の一部である行は宛先パーティションＰ１に属するので、この生成はこの宛先のパーティション＿ｉｄＰ１にマップされる。他の異なるパーティションＰ２に属する異なる宛先レプリカがソースレプリカと同期すると、生成マッピングによってＰ１個の生成が削除される。パーティション＿ｉｄが－１という特別の値を有する場合、この生成がグローバルであり、したがって、すべてのパーティションに関連する生成であることを示している。

【０１１８】

以下にテーブル５、すなわちサンプル生成パーティションマッピングテーブル３０８を示す。最後の列は、サンプルデータの解釈のためのみのものであり、スキーマの一部として必要なものではない。

【０１１９】

【表５】

テーブル５　サンプルGenerationPartitionMappingテーブル

生成_id	パーティション_id	テーブルの解釈 （スキーマの一部ではない）
G1	1	生成G1は受持ち区域が「WA」であるパーティションに対応する。
G2	2	生成G2は受持ち区域が「CA」であるパーティションに対応する。
G3	3	生成G3はグローバル生成であり、すべてのパーティションに伝播させる必要がある。
...	...	...

【０１２０】

生成の区分（partitioning）は、本発明のパーティショングループ分け態様に基づく最適化であり、３つの利益を与え、すなわち、関連する変更が効率的に削除され、多数の宛先レプリカがソースレプリカと同期しているときでも同期セッションの持続時間が予測可能であり、生成の伝播によってネットワーク性能が向上する。

【０１２１】

各々が異なるパーティションを有する多数の宛先レプリカとソースレプリカが同期する

トポロジーでは、生成区分によって、同期プロセスは関連する変更を効率的に削除することができる。たとえば、各部分集合が１人の販売員に特有の販売情報を含むデータのいくつかの異なる部分集合を与える１つのソースレプリカがあるトポロジーを考える。最初、１０００個の宛先レプリカが１００００個の変更をソースレプリカに伝播させ、１００００個の変更がソースレプリカにおいて１００個の異なる生成に含められると仮定した場合、すべての１００個の宛先レプリカがその変更をソースレプリカに伝播させると、合計で１０００×１００００個、すなわち１０００万個の変更がソースレプリカにおける１０万個の生成に含められる。したがって、生成パーティションマッピングテーブル３０８は、各集合が単一のパーティションにマッピングされる１００個の生成の集合を持つ１０万個のエントリを有する。

【０１２２】

異なるパーティション＿ｉｄ＝Ｐ１００１を有する第１の宛先レプリカがソースレプリカと同期すると、すべての生成がパーティションＰ１～Ｐ１０００に対応するパーティションにマップされるため、すべての１０万個の生成（したがって、１０００万個の変更）はパーティション＿ｉｄ＝Ｐ１００１を有する第１の宛先レプリカと無関係になる。一方、パーティション＿ｉｄ＝Ｐ５００を有する第２の宛先レプリカがソースレプリカと同期した場合、同期によって、第２の宛先レプリカ（パーティション＿ｉｄ＝Ｐ５００を有する）に関連する生成であり、したがって、厳密に１００００個の変更を第２の宛先レプリカに伝播させる厳密に１００個の生成が効果的に列挙される。

【０１２３】

生成区分最適化の他の利点は、パーティションに関連する変更の数に対して宛先レプリカによって調べられる同期メタデータの量が調節されるので、ソースレプリカと宛先レプリカとの間の同期セッションの持続時間が予測可能であることである。一例として、パーティション＿ｉｄ＝Ｐ５００を有する宛先レプリカが長時間にわたってソースレプリカと同期していないと仮定する。マージレプリケーションの既存のバージョンでは、この宛先レプリカは最終的にソースレプリカと同期する際にペナルティが科される。なぜなら、この宛先レプリカは、ソースレプリカに累積しているすべての生成を列挙する必要があるからである。すなわち、たとえば、１０００個の宛先レプリカがそれぞれ１００個の生成を伝播させた場合、関連する生成および無関係の生成から成るこの集合は合計で１０万個の生成を有し、これは顕著な数である。１０万個の生成が検討された後、「同期セッション当たりパーティション計算」アルゴリズムは無関係の変更を削除し、関連する変更のみが宛先レプリカに伝播する。

【０１２４】

生成が関連するパーティション識別子に区分されている場合を考える。本発明の態様によれば、宛先レプリカ（たとえば、パーティション＿ｉｄ＝Ｐ５００）との同期セッションにはペナルティが科されない。これは、この宛先レプリカがただちに関連する生成を識別できるからである。無関係の生成が宛先レプリカに伝播するのが回避されるので、ネットワーク化特性に対する生成区分の影響は顕著である。引き続きこの例で言えば、このことは、１０万マイナス１００、すなわち９９９００個の生成（宛先レプリカに伝播させる必要のない生成）の節約に相当する。ソースレプリカでローカル変更が加えられたとき、それらの変更が単一のパーティションのみに関連する場合でも、パーティションの集合に関連する場合でも、これらのローカル変更をグローバル生成として伝播させた方がより好ましくかつより効率的である場合がある。この場合の前提は、ソースレプリカで加えられる変更が実質的にすべての宛先レプリカに関連する変更であり、一方、宛先レプリカで加えられる変更が、同じパーティション＿ｉｄを共用する他の宛先レプリカにのみ関連する変更であることである。

【０１２５】

本発明の態様が他の用途を有することが理解されよう。たとえば、多数のクライアントユーザからアクセスできるサーバファーム（farm）では、第１のサーバの特定のユーザに対して起こったパーティション変更を用いて、このユーザに対して、第１のサーバとの関

連付けを解除して第２のサーバに関連付けすべきかどうかが判定される。この「関連付けプロセス」は、変更がデータベース情報のあるフィールドに加えられたものとして追跡された後、関連するプロセスが自動的にトリガされるように所定の基準に従って自動的に実行することができる。

【０１２６】

本発明をデータベースの変更に関して説明したが、本発明がそれに限らず、それぞれの異なるソース間でデータ同期が必要とされるあらゆる環境に適用できることがさらに理解されよう。たとえば、本発明は、ネットワーク上のあらゆるユーザおよびリソースの名前、プロファイル情報、ユーザアカウント、ネットワーク許可、マシンアドレスを調停することのできるディレクトリサービスに適用することができる。

【０１２７】

次に図６を参照すると、開示されるアーキテクチャを実行するように動作することのできるコンピュータのブロック図が示されている。本発明の様々な態様についてさらに説明するために、図６および以下の解説では、本発明の様々な態様を実施できる適切なコンピューティング環境６００について簡単にかつ概略的に説明する。上記では、１つまたは複数のコンピュータ上で実行できるコンピュータ実行可能な命令の一般的な状況に関して本発明を説明したが、当業者には、本発明を他のプログラムモジュールと組み合わせ、かつ／またはハードウェアとソフトウェアの組合せとして実施することもできることが認識されよう。一般に、プログラムモジュールは、特定のタスクを実行するかまたは特定の抽象データ型を実現するルーチン、プログラム、構成要素、データ構造などを含む。さらに、当業者には、各々が、１つまたは複数の関連する装置に動作可能に結合された、シングルプロセッサコンピュータシステムまたはマルチプロセッサコンピュータシステム、ミニコンピュータ、メインフレームコンピュータや、パーソナルコンピュータ、ハンドヘルドコンピューティングデバイス、マイクロプロセッサベースのまたはプログラム可能な家庭用電化製品などを含む、他のコンピュータシステム構成によって本発明の方法を実施できることが理解されよう。本発明の例示される態様は、ある種のタスクが通信網を通して互いにリンクされたリモート処理装置によって実行される分散型コンピューティング環境で実施することもできる。分散型コンピューティング環境では、プログラムモジュールはローカルメモリ記憶装置とリモートメモリ記憶装置の両方に配置することができる。

【０１２８】

再び図６を参照すると、本発明の様々な態様を実施する例示的な環境６００は、プロセッサ６０４、システムメモリ６０６、およびシステムバス６０８を含むコンピュータ６０２を含んでいる。システムバス６０８は、システムメモリ６０６を含むがそれに限らないシステム構成要素をプロセッサ６０４に結合する。プロセッサ６０４は、様々な市販のプロセッサのうちのどれであってもよい。デユアルマイクロプロセッサおよびその他のマルチプロセッサアーキテクチャをプロセッサ６０４として使用することもできる。

【０１２９】

システムバス６０８は、様々な市販のバスアーキテクチャのうちのどれかを用いたメモリバスまたはメモリコントローラ、周辺バス、およびローカルバスを含むいくつかの種類のバス構造のうちのどれであってもよい。システムメモリ６０６は、読取り専用メモリ（ＲＯＭ）６１０およびランダムアクセスメモリ（ＲＡＭ）６１２を含んでいる。立上げ時などにコンピュータ６０２内の各要素間での情報の転送を助ける基本ルーチンを含むＢＩＯＳ（Basic Input/Output System）がＲＯＭ６１０に記憶されている。

【０１３０】

コンピュータ６０２は、ハードディスクドライブ６１４、磁気ディスクドライブ６１６（たとえば、リムーバブルディスクに対する読取りまたは書込みを行う）、光ディスクドライブ６２０（たとえば、ＣＤ（compact disc）－ＲＯＭディスク６２２からの読取りや他の光媒体に対する読取りまたは書込みを行う）をさらに含んでいる。ハードディスクドライブ６１４、磁気ディスクドライブ６１６、および光ディスクドライブ６２０はそれぞれ、ハードディスクドライブインタフェース６２４、磁気ディスクドライブインタフェー

ス６２６、および光ドライブインタフェース６２８によってシステムバス６０８に接続することができる。各ドライブおよびそれに関連するコンピュータ読取り可能な媒体は、データ、データ構造、コンピュータ実行可能な命令などの非揮発性記憶を行う。コンピュータ６０２の場合、各ドライブおよび媒体は、適切なデジタルフォーマットでのブロードキャストプログラミング（broadcast programming）のストレージに対処する。コンピュータ読取り可能な媒体についての上記の説明はハードディスク、リムーバブル磁気ディスク、およびＣＤに関するものであるが、当業者には、ｚｉｐドライブ、磁気カセット、フラッシュメモリカード、デジタルビデオディスク、カートリッジなどによって読み取ることのできる他の種類の媒体を例示的な動作環境で使用することもでき、さらに、このような媒体が、本発明の方法を実行するコンピュータ実行可能な命令を含んでよいことを理解されたい。

【０１３１】

オペレーティングシステム６３０、１つまたは複数のアプリケーションプログラム６３２、他のプログラムモジュール６３４およびプログラムデータ６３６を含む多数のプログラムモジュールをドライブおよびＲＡＭ６１２に記憶することができる。本発明を様々な市販のオペレーティングシステムまたは複数のオペレーティングシステムの組合せによって実現できることが理解されよう。

【０１３２】

ユーザは、キーボード６３８、およびマウス６４０などのポインティングデバイスによってコンピュータ６０２にコマンドおよび情報を入力することができる。他の入力装置（図示せず）には、マイクロフォン、ＩＲ（infrared）リモートコントロール、ジョイスティック、ゲームパッド、衛星放送受信アンテナ、スキャナなどを含めてよい。これらおよびその他の入力装置は、システムバス６０８に結合されたシリアルポートインタフェース６４２を通じてプロセッサ６０４に接続されることが多いが、パラレルポート、ゲームポート、ＵＳＢ（Ｕｎｉｖｅｒｓａｌ　Ｓｅｒｉａｌ　Ｂｕｓ）、ＩＲインタフェースのような他のインタフェースによって接続してよい。モニタ６４４または他の種類の表示装置も、ビデオアダプタ６４６などのインタフェースを介してシステムバス６０８に接続されている。コンピュータは通常、モニタ６４４だけでなく、スピーカ、プリンタのような他の周辺出力装置（図示せず）を含んでいる。

【０１３３】

コンピュータ６０２は、１つまたは複数のリモートコンピュータ６４８との論理接続を用いてネットワーク化環境で動作することができる。リモートコンピュータ６４８は、ワークステーション、サーバコンピュータ、ルータ、パーソナルコンピュータ、ポータブルコンピュータ、マイクロプロセッサベースの娯楽電化製品、ピアデバイス、またはその他の一般的なネットワークノードであってよく、通常、コンピュータ６０２に関して説明した要素のうちの多くまたはすべてを含む。ただし、説明を簡単にするために、記憶装置６５０を例示する。図示の論理接続はＬＡＮ（local area network）６５２およびＷＡＮ（wide area network）６５４を含んでいる。このようなネットワーク化環境は、オフィス、企業内コンピュータネットワーク、イントラネット、およびインターネットで一般的に用いられている。

【０１３４】

コンピュータ６０２は、ＬＡＮネットワーク化環境で用いられるときは、ネットワークインタフェースまたはアダプタ６５６を通じてＬＡＮ６５２に接続される。コンピュータ６０２は、ＷＡＮネットワーク化環境で用いられるときは通常、モデム６５８を含むか、またはＬＡＮ上の通信サーバに接続されるか、あるいはインターネットなどのＷＡＮ６５４上の通信を確立する他の手段を有する。モデム６５８は、内部モデムでも外部モデムでもよく、シリアルポートインタフェース６４２を介してシステムバス６０８に接続されている。ネットワーク化環境では、コンピュータ６０２に対して図示されているプログラムモジュールまたはその一部をリモートの記憶装置６５０に記憶することができる。図示のネットワーク接続は例示的なものであり、各コンピュータ間に通信リンクを確立する他の

手段を使用できることが理解されよう。

【０１３５】

次に図７を参照すると、本発明によるサンプルコンピューティング環境７００の概略ブロック図が示されている。システム７００は１つまたは複数のクライアント７０２を含んでいる。クライアント７０２はハードウェアおよび／またはソフトウェア（たとえば、スレッド、プロセス、コンピューティング装置）であってよい。クライアント７０２は、たとえば本発明を使用することによって、クッキーおよび／または関連するコンテキスト情報を格納することができる。システム７００は１つまたは複数のサーバ７０４も含んでいる。サーバ７０４はハードウェアおよび／またはソフトウェア（たとえば、スレッド、プロセス、コンピューティング装置）であってよい。サーバ７０４は、たとえば本発明を使用することによって、変換を実行するスレッドを格納することができる。クライアント７０２とサーバ７０４との間の１つの可能な通信は、２つ以上のコンピュータプロセス間に送信されるようになっているデータパケットの形であってよい。データパケットは、たとえばクッキーおよび／または関連するコンテキスト情報を含んでよい。システム７００は、クライアント７０２とサーバ７０４との間の通信を容易にするのに用いることのできる通信フレームワーク７０６を含んでいる。クライアント７０２は、クライアント７０２に対してローカルな情報（たとえば、クッキーおよび／または関連するコンテキスト情報）を記憶するのに用いることのできる１つまたは複数のクライアントデータストア７０８に動作可能に接続されている。同様に、サーバ７０４は、サーバ７０４に対してローカルな情報を記憶するのに用いることのできる１つまたは複数のサーバデータストア７１０に動作可能に接続されている。

【０１３６】

上記で説明した内容は本発明の例を含む。もちろん、本発明について説明するために、構成要素または方法の考えられるあらゆる組合せを説明するのは不可能であるが、当業者には、本発明の他の多数の組合せおよび変形が可能であることが認識されよう。したがって、本発明は添付の特許請求の範囲の要旨および範囲内のすべてのそのような変更形態、修正形態、および変形形態を包含するものである。さらに、語「含む（include）」は、詳細な説明または特許請求の範囲で使用される範囲で、語「備える（ｃｏｍｐｒｉｓｉｎｇ）」が特許請求の範囲において移行句として使用されるときに包括的と解釈されるのと同様に包括的なものである。

【図面の簡単な説明】

【０１３７】

【図１】本発明を適用できる実施形態のシステムブロック図である。

【図２】本発明を適用できる実施形態のレプリケーションプロセスのフローチャートの図である。

【図３】本発明を適用できる実施形態の宛先のメンバーシップメタデータを算出する計算アルゴリズムによって利用されるメタデータテーブルの相互関係を示す図である。

【図４】本発明を適用できる実施形態のフィルタリングおよび展開を利用するサンプル更新スキーマを示す図である。

【図５】本発明を適用できる実施形態の顧客データがある従業員から別の従業員に再割当てされるパーティション更新の例を示す図である。

【図６】本発明を適用できる実施形態の開示されるアーキテクチャを実行するように動作させることのできるコンピュータのブロック図である。

【図７】本発明を適用できる実施形態のサンプルコンピューティング環境の概略ブロック図である。

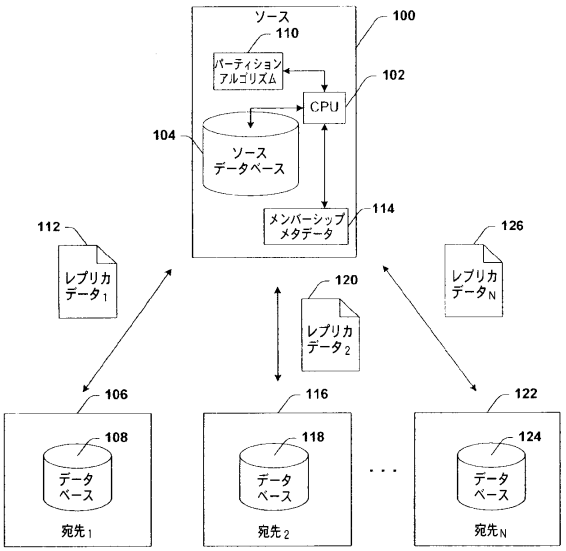
【符号の説明】

【０１３８】

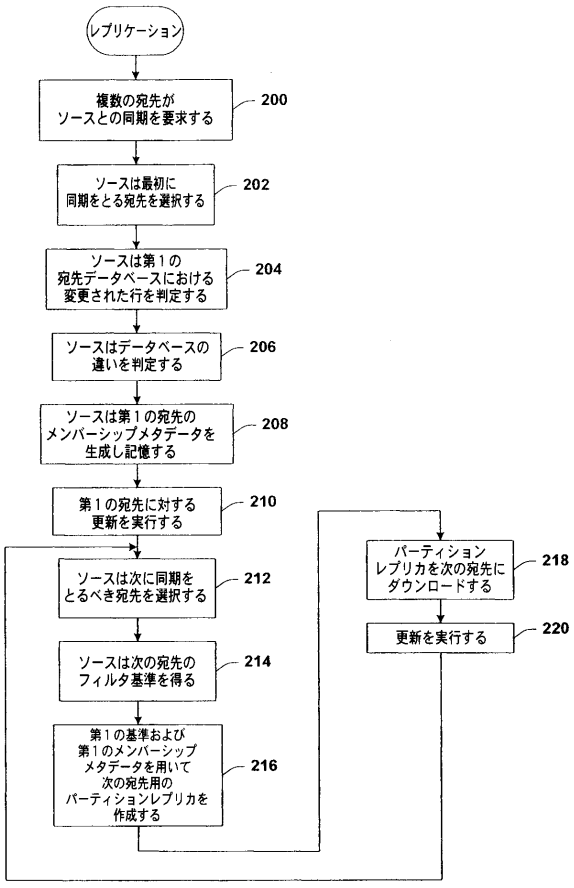
１００　ソース
１０２　ＣＰＵ
１０４　ソースデータベース

１０６　第１の宛先
１０８、１１８、１２４　宛先データベース
１１０　パーティション計算アルゴリズム
１１２　第１のレプリカデータ
１１４　メンバーシップメタデータ
１１６　第２の宛先
１２０　レプリカデータ$_2$
１２２　Ｎ番目の宛先
１２６　ダウンロードすべき変更の集合

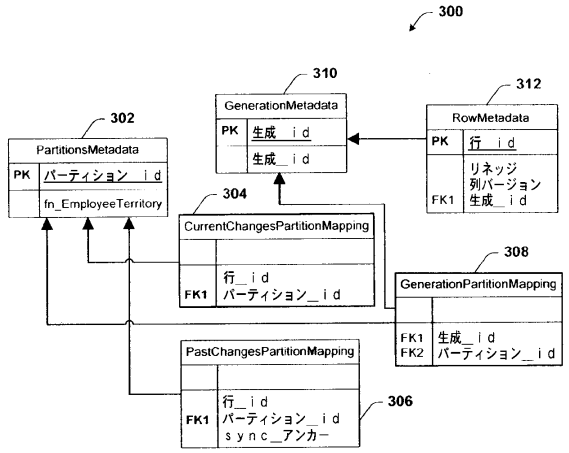
【図１】



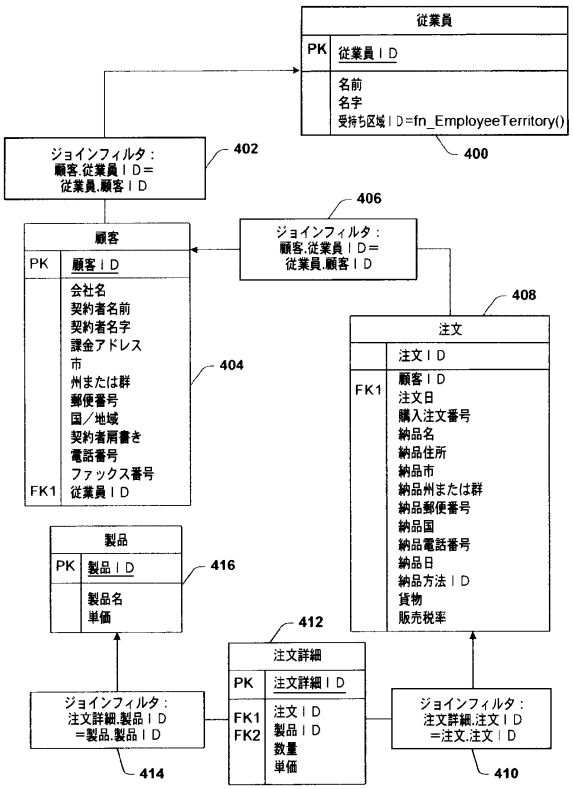
【図２】



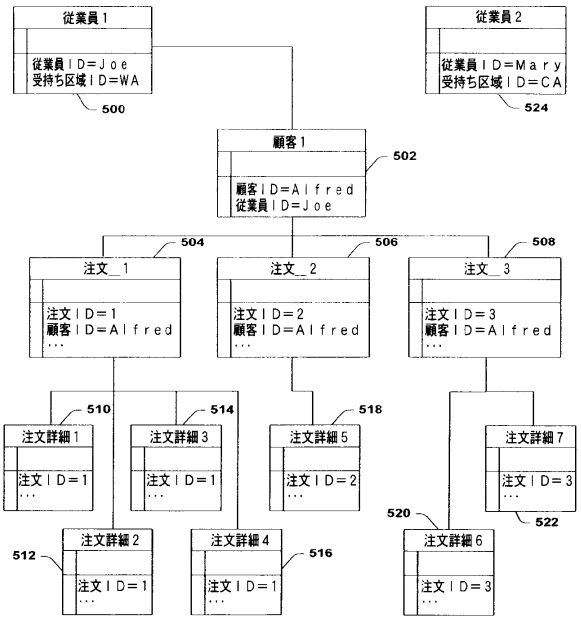
【図３】


300
310
GenerationMetadata
PK 生成_id
生成_id
312
RowMetadata
PK 行_id
リネッジ
列バージョン
FK1 生成_id
302
PartitionsMetadata
PK パーティション_id
fn_EmployeeTerritory
304
CurrentChangesPartitionMapping
行_id
FK1 パーティション_id
308
GenerationPartitionMapping
FK1 生成_id
FK2 パーティション_id
PastChangesPartitionMapping
行_id
FK1 パーティション_id
sync_アンカー
306

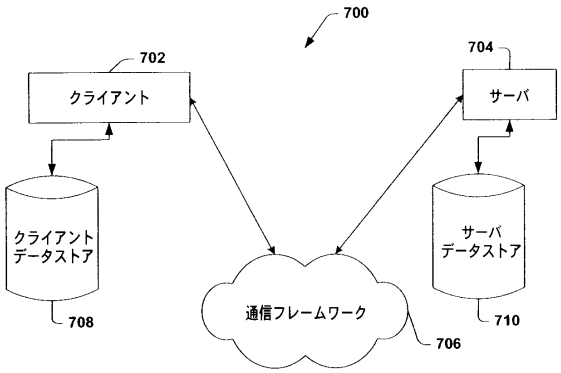
【図４】


従業員
PK 従業員ID
名前
名字
受持ち区域ID=fn_EmployeeTerritory()
400
ジョインフィルタ：
顧客.従業員ID=
従業員.顧客ID
402
406
ジョインフィルタ：
顧客.従業員ID=
従業員.顧客ID
顧客
PK 顧客ID
会社名
契約者名前
契約者名字
課金アドレス
市
州または群
郵便番号
国／地域
契約者肩書き
電話番号
ファックス番号
FK1 従業員ID
404
408
注文
注文ID
FK1 顧客ID
注文日
購入注文番号
納品名
納品住所
納品市
納品州または群
納品郵便番号
納品国
納品電話番号
納品日
納品方法ID
貨物
販売税率
製品
PK 製品ID
製品名
単価
416
412
注文詳細
PK 注文詳細ID
FK1 注文ID
FK2 製品ID
数量
単価
ジョインフィルタ：
注文詳細.製品ID
=製品.製品ID
414
ジョインフィルタ：
注文詳細.注文ID
=注文.注文ID
410

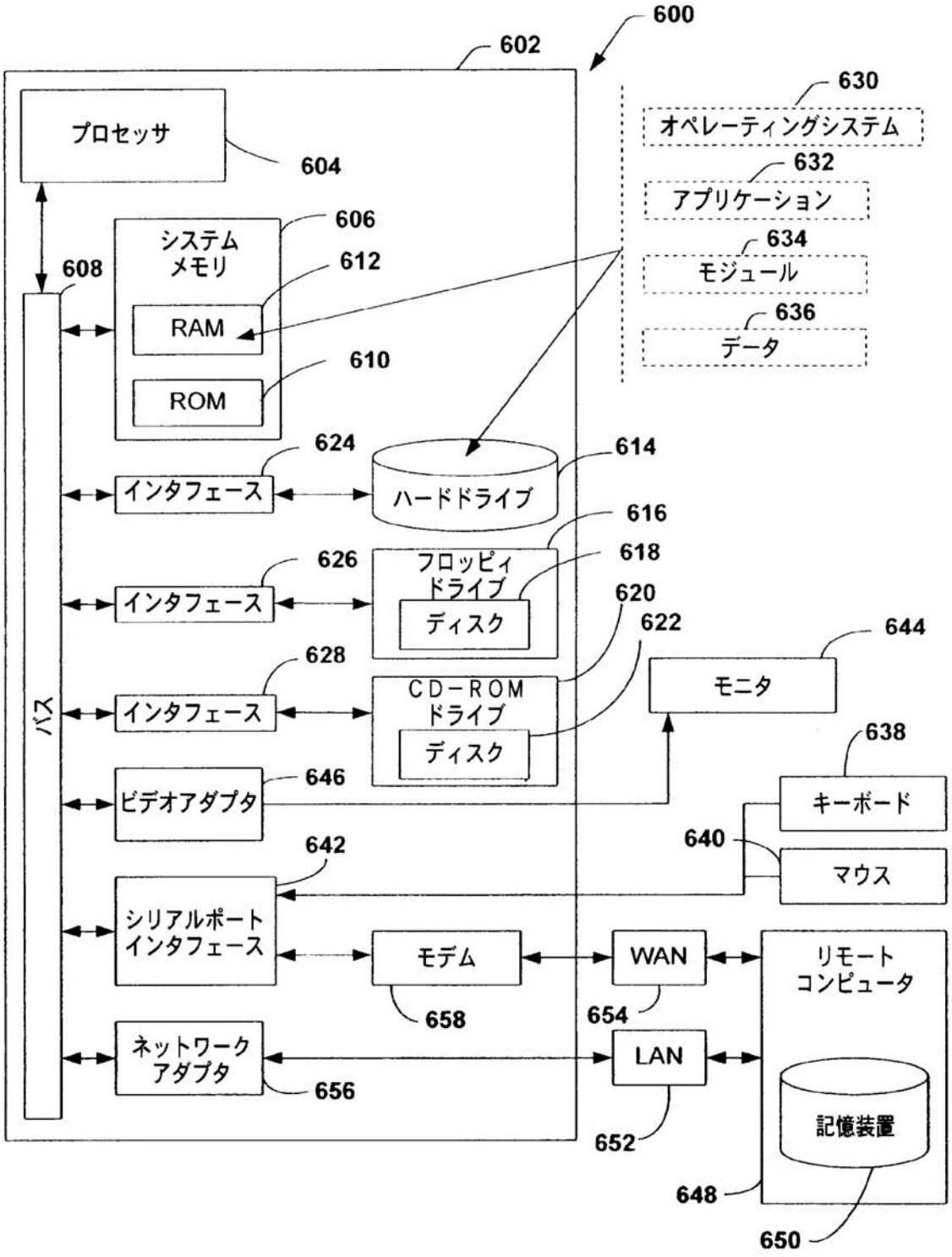
【図５】


従業員1
従業員ID=Joe
受持ち区域ID=WA
500
従業員2
従業員ID=Mary
受持ち区域ID=CA
524
顧客1
顧客ID=Alfred
従業員ID=Joe
502
504
注文_1
注文ID=1
顧客ID=Alfred
506
注文_2
注文ID=2
顧客ID=Alfred
508
注文_3
注文ID=3
顧客ID=Alfred
510
注文詳細1
注文ID=1
514
注文詳細3
注文ID=1
518
注文詳細5
注文ID=2
520
注文詳細7
注文ID=3
522
512
注文詳細2
注文ID=1
516
注文詳細4
注文ID=1
注文詳細6
注文ID=3

【図７】


700
702
クライアント
704
サーバ
クライアント
データストア
708
通信フレームワーク
706
サーバ
データストア
710

【図６】


600
602
プロセッサ
604
システム
メモリ
606
608
612
RAM
610
ROM
630
オペレーティングシステム
632
アプリケーション
634
モジュール
636
データ
624
インタフェース
614
ハードドライブ
616
626
フロッピィ
ドライブ
618
インタフェース
ディスク
620
622
644
モニタ
628
CD−ROM
ドライブ
インタフェース
ディスク
バス
638
646
ビデオアダプタ
キーボード
640
642
マウス
シリアルポート
インタフェース
モデム
WAN
リモート
コンピュータ
658
654
ネットワーク
アダプタ
LAN
656
652
記憶装置
648
650

フロントページの続き

(72)発明者　チャルマシー　ナラヤナン
アメリカ合衆国　９８０７４　ワシントン州　サマミッシュ　１９１　プレイス　ノースイースト　４６０９

審査官　田川　泰宏

(56)参考文献　ブラム　ダニエル, Ｗｉｎｄｏｗｓ　２０００　Ａｃｔｉｖｅ　Ｄｉｒｅｃｔｏｒｙ　入門, 株式会社アスキー, ２０００年　1月２１日, 初版, p.189-215
ガルシア　マルシ　フロホック, Ｍｉｃｒｏｓｏｆｔ　ＳＱＬ　Ｓｅｒｖｅｒ　２０００　オフィシャルマニュアル　下, 日経ＢＰソフトプレス, ２００１年　3月１９日, 初版, p.21-150

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ　１２／００