(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2004-303211
(P2004-303211A)
(43)公開日　平成16年10月28日(2004.10.28)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 12/00	G06F 12/00	533J	5B082
	G06F 12/00	513J	
	G06F 12/00	514M	
	G06F 12/00	546M	

審査請求 未請求　請求項の数 45　ＯＬ　外国語出願　（全 63 頁）

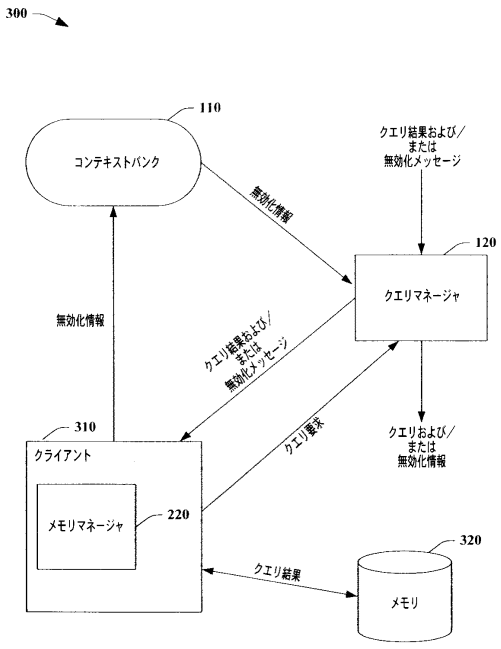
(21)出願番号　特願2004-58206 (P2004-58206)
(22)出願日　平成16年3月2日 (2004.3.2)
(31)優先権主張番号　10/402,031
(32)優先日　平成15年3月28日 (2003.3.28)
(33)優先権主張国　米国 (US)

(71)出願人　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　マイケル　ジェイ．ピゾ
アメリカ合衆国　９８００８　ワシントン州　ベルビュー　ウエスト　レイク　サマミッシュ　５２８

最終頁に続く

(54)【発明の名称】データベース結果および導出オブジェクトをキャッシュし、無効にするためのシステムおよび方法

(57)【要約】
【課題】　生データ、オブジェクト、クエリ可能なデータセット、完全または部分Ｗｅｂ応答などデータベースクエリから生成された結果を、現在およびその後の使用のためにメモリに保存し、保存されている結果が一致しなくなるとそれらを無効にして、一致しない結果の使用を軽減するためのシステムおよび方法を提供すること。
【解決手段】　保存されている結果は、１つまたは複数の構成要素によって使用され、一般に、その後のクエリが実質的に同じ結果を戻し得るときに、使用することができる。保存されている結果の整合性に影響を与えるデータベース変更が行われたとき、および／または有効期限が切れたとき、無効化メッセージを送信して、保存されている結果を無効にすることができる。したがって、一致しない保存結果の使用が軽減される。さらに、自動再クエリ技術を使用して、一致しない結果を自動的に更新して一致した結果を取得することができる。
【選択図】　図３



【特許請求の範囲】
【請求項１】
無効化情報を格納するためのコンテキストバンクと、
データベースクエリおよび関連する無効化情報を前記コンテキストバンクからデータベースに送信するためのクエリマネージャと、
データベースクエリに基づいた結果を格納し、前記関連する無効化情報が前記データベースによって維持されるためのメモリと
を備えたことを特徴とするメモリ無効化登録システム。
【請求項２】
前記無効化情報は、無効化文字列、サービス情報、および有効期限の少なくとも１つを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３】
前記無効化文字列は、一意の識別子、マシン名、ポート、アドレス、およびキュー名のうちの少なくとも１つを備えたことを特徴とする請求項２に記載のシステム。
【請求項４】
前記無効化文字列は、認証、暗号化、およびプロトコル情報のうちの少なくとも１つをさらに備えたことを特徴とする請求項３に記載の方法。
【請求項５】
前記クエリマネージャは、クエリとともに送信する無効化情報を取得するために前記コンテキストバンクとともに動作することを特徴とする請求項１に記載のシステム。
【請求項６】
前記無効化情報は、複数のクエリに関連し格納されている結果に、関連付け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７】
前記クエリマネージャは、前記格納されている結果に影響を与えることになるデータベース変更、および前記無効化情報で定義されている期限切れのうち少なくとも一方の後で、無効化メッセージを受信することを特徴とする請求項１に記載のシステム。
【請求項８】
前記データベース変更は、前記変更が同期的に配信されるまで、阻止され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】
前記データベース変更は、前記変更が非同期的に配信される間に、終了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１０】
前記無効化メッセージが受信された後で前記メモリ内に保存されている結果を自動的に無効にす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】
前記無効化メッセージが受信された後で前記データベースの再クエリを自動的に行い、メモリ内に保存されている結果を更新することを特徴とする請求項１に記載のシステム。
【請求項１２】
前記一意の識別子は、前記保存されている結果と前記ソースクエリの間の関連を定義するためにランタイム時に生成されることを特徴とする請求項３に記載のシステム。
【請求項１３】
１つまたは複数のクエリを保存されている結果とともにグループ化するために、前記一意の識別子を使用することを特徴とする請求項１２に記載のシステム。
【請求項１４】
前記一意の識別子をセキュリティ機構として使用して、偽りの無効化情報および偽りの無効化メッセージのうちの少なくとも１つを軽減することを特徴とする請求項１２に記載のシステム。
【請求項１５】

結果の格納と無効化を調整するためのメモリマネージャ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１６】

前記自動無効化は、一致しない結果を前記メモリから削除すること、前記メモリ内の一致しない結果を上書きすること、および前記メモリ内の一致しない結果にタグを付けることのうちの少なくとも１つを備えたことを特徴とする請求項１０に記載のシステム。

【請求項１７】

前記メモリはキャッシュメモリを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１８】

前記メモリは複数の構成要素によって共有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１９】

ＳＱＬベースの環境で使用することを特徴とする請求項１に記載のシステム。

【請求項２０】

その後の要求による使用のために前記結果をメモリに格納すること、および前記格納されている結果が前記データベースと一致しなくなったときに前記結果を無効にすることを要求するためのコメントおよびページ指示のうちの少なくとも１つ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２１】

前記コンテキストバンクは、前記格納されている結果の生成に使用するクエリへの前記無効化情報の伝達を容易にすることを特徴とする請求項１に記載のシステム。

【請求項２２】

前記格納されている結果は、生データ、変換データ、例外的なデータ、集計データ、概要データ、ピボットテーブルデータ、前記生データから生成されたオブジェクト、部分Ｗｅｂ応答、完全Ｗｅｂ応答、およびクエリ可能データのうちの少なくとも１つを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２３】

１つまたは複数の構成要素に関する無効化情報を格納するコンテキストと、

データベースクエリおよび無効化情報を送信するためのクエリマネージャと、

前記データベースクエリから生成された結果を格納するための１つまたは複数のメモリと

を備えたことを特徴とする自動メモリ無効化システム。

【請求項２４】

前記１つまたは複数のメモリはローカルキャッシュメモリ、補助キャッシュメモリ、およびリモート高速メモリを備えたことを特徴とする請求項２３に記載のシステム。

【請求項２５】

偽りの無効化情報および偽りの無効化メッセージについての悪意のあるクエリのうちの少なくとも１つを軽減するために、クエリエンジンおよび前記依存構成要素に対して動作するセキュリティ層をさらに備えたことを特徴とする請求項２３に記載のシステム。

【請求項２６】

保存されている結果が一致しなくなったときに前記保存されている結果を自動的に無効にすること、および保存されている結果が無効になったことを構成要素に通知することのうち少なくとも一方を提供するための機構をさらに備えたことを特徴とする請求項２３に記載のシステム。

【請求項２７】

セキュリティ層としてファイアウォールをさらに備えたことを特徴とする請求項２３に記載のシステム。

【請求項２８】

前記１つまたは複数のメモリが前記ファイアウォール内に存在することを特徴とする請求項２７に記載のシステム。

【請求項２９】
前記ファイアウォールは、前記１つまたは複数のメモリにおいて、結果の格納、結果へのアクセス、および結果の無効化の少なくとも１つを容易にすることを特徴とする請求項２７に記載のシステム。
【請求項３０】
前記１つまたは複数のメモリは前記１つまたは複数の構成要素から同時にアクセス可能であることを特徴とする請求項２７に記載のシステム。
【請求項３１】
登録された結果を保存して前記結果が一致しなくなったときに自動的に無効化するための方法であって、
無効化情報をコンテキストに格納すること、
前記無効化情報を関連するクエリとともに送信すること、および
前記クエリから生成された前記結果をメモリに保存すること
を備えることを特徴とする方法。
【請求項３２】
一致した結果を確実にするために、前記クエリの後、かつ前記結果を格納する前に変更が行われたかどうかについて前記クエリが行われたデータベースをチェックすることをさらに備えることを特徴とする請求項３１に記載の方法。
【請求項３３】
前記無効化情報は無効化文字列、サービス情報、および有効期限のうちの少なくとも１つを備えることを特徴とする請求項３１に記載の方法。
【請求項３４】
前記無効化文字列は識別子、マシン名、ポート、アドレス、およびキュー名のうちの少なくとも１つを備えることを特徴とする請求項３３に記載の方法。
【請求項３５】
前記無効化文字列は認証、暗号化、およびプロトコル情報のうちの少なくとも１つをさらに備えることを特徴とする請求項３４に記載の方法。
【請求項３６】
前記メモリはキャッシュメモリであることを特徴とする請求項３１に記載の方法。
【請求項３７】
結果が一致しなくなったときにデータベースクエリから生成された保存されている前記結果を自動的に無効にするための方法であって、
無効化情報の少なくとも一部を含む無効化メッセージを受信すること、および
前記無効化情報の少なくとも一部を使用して保存されている結果を無効にすること
を備えることを特徴とする方法。
【請求項３８】
前記結果を自動的に無効にすることをさらに備えることを特徴とする請求項３７に記載の方法。
【請求項３９】
一致した結果を生成し保存するために自動的に再クエリを行うことをさらに備えることを特徴とする請求項３７に記載の方法。
【請求項４０】
前記無効化メッセージは前記格納されている結果に影響を与えることになるデータベース変更および有効期限切れのうちの少なくとも一方に関連付けられることを特徴とする請求項３７に記載の方法。
【請求項４１】
結果をメモリに格納すること、および前記格納されている結果が一致しなくなったときに前記格納されている結果を自動的に無効化することを容易にする２つ以上のコンピュータ構成要素の間で送信されるデータパケットであって、
取り出し、クエリとともに送信できるようにコンテキスト内に格納され、保存されてい

る結果を自動的に無効にする無効化メッセージに使用される無効化情報
を備えることを特徴とするデータパケット。
【請求項４２】
結果を保存し、前記結果が一致しなくなったときに前記結果を無効にするためにメモリ無効化登録システムのコンピュータ実行可能構成要素を格納するコンピュータ可読媒体であって、
構成要素に関連する無効化情報を格納するためのコンテキストと、
前記コンテキストから無効化情報を取得し、要求されたクエリとともに前記無効化情報を送信するためのクエリマネージャと、
結果を保存し、一致しない結果を無効にすることを容易にするためのメモリマネージャと
を備えることを特徴とするコンピュータ可読媒体。
【請求項４３】
メモリ内に保存されている結果を無効にするためのシステムであって、
データベースへのクエリを行うよう要求する構成要素の無効化情報を取得するための手段と、
前記無効化情報をクエリとともに送信するための手段と、
無効化メッセージを受信するための手段と、
メモリ内に格納されている一致しない結果を無効にするための手段と
を備えることを特徴とするシステム。
【請求項４４】
一致した結果を得るために前記データベースの再クエリを自動的に行うための手段をさらに備えることを特徴とする請求項４３に記載のシステム。
【請求項４５】
無効化情報および無効化メッセージが偽りではないことを確認するための手段をさらに備えることを特徴とする請求項４３に記載のシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は一般にデータベースに関し、より詳細には、データベースクエリから生成された結果をその後使用するために保存し、保存されている結果がデータベースと一致しなくなったときにそれらを無効にするためのシステムおよび方法に関する。
【背景技術】
【０００２】
コンピューティング技術およびネットワーキング技術によって、日常生活の多くの重要な側面が変化してきた。コンピュータは、贅沢な教育用ツールおよび／または娯楽センターの代わりに家庭の必需品となり、ユーザに、資金の管理および予測、暖房、冷房、照明、セキュリティなど家庭用器具の運転の制御、および永続的で信頼できる媒体への記録および画像の保管を行うためのツールを提供する。インターネットなどのネットワーキング技術によって、ユーザに、リモートシステム、情報および関連するアプリケーションへの実質的に無限のアクセスが提供される。
【０００３】
コンピューティング技術およびネットワーキング技術の堅牢化、安全化、高信頼化が進むにつれて、多くの消費者、卸売業者、小売業者、企業、教育機関などは、パラダイムを変更しつつあり、従来の手段の代わりにインターネットを使用して業務を行うようになってきている。例えば、多くの企業および消費者は、Ｗｅｂサイトおよび／またはオンラインサービスを提供している。例えば、今日消費者は、インターネットを介して各自のアカウントにアクセスし、残高照会、振込み、請求書の支払いなどますます多くの利用可能なトランザクションを行うことができる。
【０００４】

一般に、インターネットセッションは、ユーザがクライアントアプリケーション（Ｗｅｂサーバなど）とインターフェースをとり、クライアントアプリケーションからアクセス可能なデータベースに情報を格納するデータベースサーバと対話することを含む。例えば、株式市場のＷｅｂサイトは、株価を取得し、株を購入するためのツールをユーザに提供することができる。ユーザは、銘柄記号を入力し、マウスをクリックしてクエリをアクティブにすることによって株価を要求することができる。次いでクライアントアプリケーションは、株情報についてデータベースへのクエリを行い、株価を戻す。同様に、ユーザは、適した情報を提供することによって株を購入することができ、注文を送信することによってデータベースクエリが開始して現在の価格情報および注文状況が戻される。

【０００５】

より多くのユーザがこうしたサービスを活用するようになるにつれて帯域幅の消費が増大し、帯域幅は有限のリソースであるため、パフォーマンスおよび／または速度が低下する可能性がある。別の欠点は、使用可能なデータの量に対する使用可能なデータ転送率が限られることである。例えば、大量のデータ（例えば様々なサーバにわたって分散されている）を取得する要求は、帯域幅を使用可能なときでさえ、時間がかかり、パフォーマンスが制限される可能性がある。

【０００６】

【特許文献１】米国特許第６，０６１，６７７号明細書
【特許文献２】米国特許第６，３４７，３１２号明細書
【特許文献３】米国特許第６，３５６，８８９号明細書
【特許文献４】米国特許第６，３８９，４１４号明細書
【特許文献５】米国特許第６，５１９，５８７号明細書

【発明の開示】

【発明が解決しようとする課題】

【０００７】

上記のことは、複数のユーザが頻繁に同様の要求を行うことによっていっそうひどくなる。例えば、現在の株価を要求しているユーザは、更新された株価の取得要求を頻繁に行う可能性がある。したがって、ユーザは頻繁に帯域幅を消費し、実質的に同様の情報を取得する。さらに、他の複数のユーザが、実質的に同様の情報の取得要求を同時に行う可能性がある。その後データベーステーブルが変更されると戻された結果が無効になる可能性があるため、取得したデータの整合性が保証できないという点で、ユーザはさらに制約を受ける。

【課題を解決するための手段】

【０００８】

以下に、本発明の一部の態様を基本的に理解できるようにするために本発明の概略を示す。この概略は、本発明の広範にわたる概要ではない。本発明の鍵となる／重要な要素を識別するもの、または本発明の範囲を画定するものではない。唯一の目的は、後述するより詳細な説明の前置きとして簡略化した形で本発明の一部の概念を示すことである。

【０００９】

本発明は、データベースの変更および／または有効期限切れのために結果が無効になったとき、データベースクエリから生成された保存されている結果を無効にするシステムおよび方法に関する。したがって本発明は、データベースへのクエリを行い、データベースクエリから生成された結果を保存し、データベースの変更および／または有効期限によって結果が無効になるまで保存されている結果を使用する機構を提供する。

【００１０】

一般に、データベースは、データベースへのクエリによって取得可能な情報を格納する。ユーザは、アプリケーションとインターフェースをとり、例えばデータベース内に格納されている情報の少なくとも一部を取得することができる。ユーザは、例えばＷｅｂサービスを使用し、サーバに格納されている情報に関連する要求を送信することができる。要求によって関連性のあるデータベースの１つまたは複数のクエリが開始され、次いで１つ

または複数のクエリの結果を使用して応答を生成し、Ｗｅｂサービスから戻され、表示され、消費され、かつ／またはさらに使用されることができる。

【００１１】

多くの場合、データベースは、一般的なリソース競合の元となる可能性がある。例えば、複数のユーザが、データベースへのクエリを同時に行い、同様の結果を戻す同様の要求を送信する可能性がある。同様の結果についてのこうした同時のクエリによって、例えばデータベースの負荷の増大、帯域幅の低減、パフォーマンスの低下、リソースの消費がもたらされる可能性がある。他の例では、１人または複数のユーザが、同様の結果を戻す要求を頻繁に行う可能性がある。同様に、システムパフォーマンスは、例えばデータベース負荷によって低下する可能性がある。上記の場合、データベースが、複数のユーザ要求および／またはユーザの間で共有された状態を表し、共有される競合状態は、システムのパフォーマンスおよびスケーラビリティを制限する可能性がある。

【００１２】

クエリ数を低減するために一般的に使用される技術は、頻繁に使用されるデータをデータベースから外部に（例えば生データ、変換データ、クエリ可能な形のデータの態様で）保存し、クエリをさらに実行する代わりに保存された結果を使用することを含む。しかし、ユーザへの通知なしに、その後データベースの変更が行われることによって保存されているデータがデータベースと一致しなくなる可能性があるため、保存されているデータの整合性が損なわれる。

【００１３】

本発明のシステムおよび方法によって、データベースの負荷が低減（例えば実質的に同様の情報に対するクエリが低減）され、拡張性が向上し、処理の反復が低減され、データの整合性が保証される。このシステムおよび方法は、コンテキストバンク（ｃｏｎｔｅｘｔ　ｂａｎｋ）に無効化情報を格納することを含み、そこから無効化情報を取得し、クエリとともに送信することにより、クエリをクエリ要求側に関連付けてデータベースに登録することができる。次いでクエリから生成された結果（生データ、変換データ、クエリ可能な形のデータ、部分応答および完全応答など）を、様々なユーザ（Ｗｅｂサーバなど）がデータベースの外部にあるアクセス可能な記憶媒体に保存し、１回または複数回使用することができ、したがって実質的に同様の情報に対するクエリが軽減される。

【００１４】

格納されている結果に影響を与えることになるデータベースの変更が行われると、保存されている結果の無効を容易にするための無効化メッセージが送信されて、ユーザが一致しないデータを使用するのを軽減することができる。一般にクエリとともに送信された無効化情報を使用して無効化メッセージが構築される。さらに、有効期限を定義して、たとえデータベースの変更が行われていない場合でも、有効期限が切れた後は保存されている結果が無効になるようにすることができる。保存されている結果を無効にした後、その結果を例えば自動的に破棄し、かつ／または自動的に更新することができる。

【００１５】

さらに、本発明のシステムおよび方法は、Ｗｅｂ環境で使用することができ、データベースクエリから生成されたＷｅｂ応答を全部または一部格納し、その後の要求に応答して戻すことができる場合、Ｗｅｂサーバリソースの待ち時間および使用をさらに低減することができる。例えば、開発者は、Ｗｅｂページ内にコメントを埋め込むことによって、クエリの結果が変更されるまで、データベースクエリに基づいて生成された応答がキャッシュされるよう要求することができる。コメントは、Ｗｅｂサーバを呼び出し、データベース依存関係をキャッシュされる応答に関連付け、その依存関係に関連する無効化情報および／または依存関係自体をコンテキストバンクに格納する。格納された無効化情報は、自動的に取得され、その後のクエリとともにデータベースに送信される。上述のように、以前クエリされた結果が一致しなくなるようなデータベースへの変更が行われたとき、および／または有効期限が切れたときに無効化情報を無効化メッセージに使用することができる。

【００１６】
無効化情報をコンテキストバンクに格納し、使用することによって、無効化情報をクエリと関連付けるためのインフラストラクチャが容易にインスタンス化され、クエリ応答が容易に保存（例えばキャッシュ）されるようになる。したがって、格納されている無効化情報を使用することによって、Ｗｅｂサーバ（および／またはクエリ要求を処理することができる他の構成要素）が応答の生成についての知識を持つ必要が軽減される。さらに、格納されている無効化情報を使用することによって、Ｗｅｂサーバによって応答がどのように使用されるかを、応答を生成する構成要素が知る必要が軽減される。
【００１７】
本発明によれば、本明細書に記載したシステムおよび方法は、無効化情報を格納し、クエリを実行し、そのクエリに基づいた結果を保存し、無効化メッセージを受信し、かつ／または保存されている情報を無効にするための構成要素を使用する。さらに、非同期および同期の通信の様々なデータ転送技術、および安全でない偽りの情報およびメッセージの存在および／または転送を軽減するセキュリティ技術を使用することができる。実際には、このシステムおよび方法を、クライアント－サーバおよび／またはＳＱＬベースの環境など様々な環境で使用することができる。
【００１８】
上記および関連の目的の達成のために、本発明は、以下で十分説明し、特に特許請求の範囲で指摘する特徴を含む。以下の説明および添付の図面では、本発明の一部の態様および実装形態の例を詳細に述べる。ただしこれらは、本発明の原理を使用することができる様々な方法のうちのごく一部を示すものである。本発明の他の目的、利点、および新しい特徴は、図面と併せ読めば、本発明の以下の詳細な説明から明らかになる。
【発明を実施するための最良の形態】
【００１９】
次に、図面を参照して本発明を説明する。図中、全体を通じて同様の要素への言及には同様の参照符号を使用する。以下の説明では、説明上、本発明を完全に理解できるようにするために、特定の詳細を数多く記載している。しかし、本発明をこうした特定の詳細の限外で実施できることは理解できよう。他の例では、本発明を説明しやすくするために、よく知られている構造および装置をブロック図の形で示している。
【００２０】
本願で使用する場合、「構成要素」という用語は、ハードウェア、ハードウェアおよび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のいずれかのコンピュータ関連のエンティティを指すものとする。例えば、構成要素は、それだけには限定されないが、プロセッサ上で動作するプロセス、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とすることができる。一例として、サーバ上で動作するアプリケーションおよびサーバをコンピュータ構成要素とすることができる。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していてもよく、１つの構成要素を１つのコンピュータ上に配置する、および／または２つ以上のコンピュータ間に分散することもできる。「スレッド」とは、オペレーティングシステムカーネルが実行する予定のプロセス内のエンティティである。各スレッドは関連する「コンテキスト」を有しており、これは、スレッドの実行に関連する揮発性データである。スレッドのコンテキストは、システムレジスタの内容およびスレッドのプロセスに属する仮想アドレスを含む。したがって、スレッドのコンテキストを含む実際のデータは、その実行に伴って変化する。
【００２１】
本発明は、（例えば動的および静的）データベースクエリから生成された結果をメモリに格納するためのシステムおよび方法に関し、そこで結果を取得し、使用し、（データベースの変更や有効期限切れなど）一致しなくなったときに無効にすることができる。このシステムおよび方法によって、実質的に同様の（例えば重複する）情報についてのクエリの低減によってシステム負荷、およびリソース競合が軽減される。さらに、このシステム

および方法によって、一致しない保存結果の使用が軽減される。したがって、このシステムおよび方法は、情報を得るためにデータベースへのクエリを行い、結果をキャッシュなどのメモリに格納し、保存された結果を１回、または実質的に同様の結果が必要な場合は複数回使用し、一致しなくなったときに保存されている結果を自動的に無効にし、任意選択で、自動的な再クエリおよび保存によってメモリを更新するための機構を提供する。

【００２２】

図１を参照すると、本発明の一態様による無効化登録システム１００の例を示している。無効化登録システム１００は、コンテキストバンク１１０、およびクエリマネージャ１２０を含む。

【００２３】

コンテキストバンク１１０を使用して、保存されている結果に影響を及ぼし得るデータベース変更が行われたとき、および／または有効期限が切れたときに保存されている結果を容易に無効にするための情報を提供することができる。例えば、コンテキストバンク１１０を使用して、（例えば後述する依存関係など）構成要素に関連する無効化情報を格納することができる。無効化情報は、データベースクエリによってデータベースにリンクすることができる。結果は、保存し、１回または複数回使用することができる。クエリに影響を及ぼし得るデータベースへの変更が行われたとき、および／または期限切れになったとき、対応する無効化情報を使用して、例えば通知、イベント、フラグ、保存されている結果の自動無効化、および一致した結果を自動再クエリにより取得することによるメモリの更新などによって保存されている結果を容易に無効にすることができる。

【００２４】

一般の無効化情報は、無効化文字列、サービス情報、およびタイムアウトを含む。無効化文字列は一般に、構成要素識別子、位置（マシン名など）、ポート、アドレス、および／またはキュー名を含み、例えば認証情報、暗号化、およびプロトコル優先順位（ｐｒｏｔｏｃｏｌ　ｐｒｅｆｅｒｅｎｃｅｓ）などをさらに含むことができる。サービス情報は、例えば保存結果の自動削除用フラグ、結果を自動的に更新するためのフラグ、結果に無効であると自動的にタグ付けするためのフラグ、保存されている結果に影響を与えることなく構成要素に通知するためのフラグを含むことができる。タイムアウトは、例えばデータベースの変更、ファイルの変更、時間をベースにすることができる。上記は例であり、限定するものではないことを理解されたい。したがって、本発明の一態様に従って異なる情報をさらに含める、および／または含める情報をより少なくすることができる。

【００２５】

コンテキストバンク１１０に格納されている無効化情報は、クエリマネージャ１２０から使用可能である。クエリマネージャ１２０は、コマンドおよび／またはクエリの実行時に無効化情報を使用することができる。例えばクエリマネージャ１２０は、コンテキストバンク１００から無効化情報を取得し、この無効化情報をクエリに含め、無効化情報をクエリの一部とする、かつ／またはクエリに関連付けることができる。

【００２６】

本発明の一態様では、ページ指示（Ｗｅｂページ内のコメントなど）を使用して、構成要素（依存オブジェクトなど）のインスタンス化、およびその構成要素に関連する無効化情報のコンテキストバンク１１０への格納を開始することができる。コンテキストバンク１１０を使用して無効化情報を格納することによって、クエリマネージャ１２０を呼び出すアプリケーションが、どのように応答をキャッシュするかを知る必要なく、クエリとともに無効化情報を使用するインフラストラクチャを容易に確立できるようになる。

【００２７】

クエリマネージャ１２０は、データベースへのクエリ要求を介して呼び出されると、コンテキストバンク１１０と対話して、その構成要素に関連付けられた無効化情報を取得する。次いでクエリマネージャ１２０は、データベースへのクエリを介して要求を実行し、コンテキストバンク１１０から取得された無効化情報が連続的に、かつ／またはクエリと同時に送信される。次いでクエリ結果を再調整し、使用して、格納すべき結果が生成され

、送信された無効化情報は、構成要素、クエリ、およびクエリが行われたデータベースに関連付けられたままである。

【００２８】

クエリの後、かつ格納すべき結果の生成前に、クエリが行われたデータベースが変更されなかった場合（例えば格納すべき結果がデータベースと一致している場合など）、その結果を使用し、メモリに保存して、さらに使用することができる。実質的に同様の結果を戻し得る要求が続いて送信されるときは、別のクエリを実行する代わりに格納されている結果を使用することができる。したがって、本発明は、パフォーマンス、スループット、処理および速度を低下させる可能性があり、リソース競合になりやすい実質的に同様の結果についてのクエリの実行を軽減する。

【００２９】

結果を保存した後、データベース変更、または期限切れによって保存されている結果が一致しなくなる可能性がある。上記のクエリの結果に影響を及ぼし得るデータベース変更が行われたとき、および／またはタイムアウトが満了したとき、たとえその結果を生成した構成要素がもはや存在していないとしても、クエリとともに送信された無効化情報を使用して構成要素（メモリマネージャ１２０など）に保存されている結果を無効（削除、更新など）にするよう通知することができる。保存されている結果への関連付けを試みる他の構成要素に、保存されている結果が無効であることを示す通知を提供することができる。したがって本発明では、構成要素に通知し、かつ／または保存されている結果を無効にすることによって、一致しない保存結果の使用が軽減される。

【００３０】

図２に移ると、本発明の一態様による無効化メモリ管理システム２００の例を示している。無効化メモリ管理システム２００は、クエリマネージャ１２０およびメモリマネージャ２２０を含む。

【００３１】

上記の機能に加えて、クエリマネージャ１２０は、クエリ結果および／または変更された、かつ／または期限切れになったデータベースに対する無効化メッセージを受信することができる。一般に、無効化情報（例えば上述したような）がクエリとともに送信された後、クエリ結果が戻される。本発明の一態様では、クエリマネージャ１２０は、戻されたクエリ結果を受信し、メモリマネージャ２２０は、クエリ結果から生成された結果の保存、および／または使用を容易にするために使用される。格納すべき結果の生成前にデータベースが変更されたこと、および／または有効期限が切れたことがわかる場合、メモリマネージャ２２０は、その結果をメモリに保存することなく、構成要素にその結果を提供することができる。データが一致しており（データベースの変更が行われないなど）、有効期限が切れていない場合、メモリマネージャ２２０は、その結果を連続的かつ／または同時に構成要素に提供し、かつ／またはメモリに保存することができる。保存されている結果には、実質的に同様のクエリが要求されたときに構成要素からアクセスすることができ、したがってリソースを消費し、パフォーマンスを低下させる実質的に同様のクエリの実行が軽減される。

【００３２】

戻された結果は、生データ、変換データ（例外的な集計データ、概要データ、ピボットテーブル、生データから生成されたオブジェクト、および／または部分または完全Ｗｅｂ応答（ｃｏｍｐｌｅｔｅ　ｗｅｂ　ｒｅｓｐｏｎｓｅ）など）、および／またはクエリ可能データ（そのサブセットを取得するためにその後のクエリによって使用できるデータセット）を含み得ることを理解されたい。さらに、結果をメモリに保存するために様々な技術を使用できることを理解されたい。例えば、第１の手法では、戻されたほぼすべての結果（例えば完全Ｗｅｂ応答）を保存することができる。第２の手法では、戻された結果の一部（例えば部分ページ）をメモリに保存することができる。例えば、その後の要求に共通し得る部分が保存される。別の例では、複数の要求に共通する結果が保存される。その場合、メモリから取得した共通の結果に対するローカルクエリでは、要求に関連するデー

タを戻すことができる。第３の手法では、個人用の結果をクエリおよび他のいくつかの手段によって取得することができ、共通の結果をメモリに保存して、使用することができる。上記の例は、説明の目的で提供したものであり、本発明を限定するものではないことを理解されたい。

【００３３】

無効化メッセージを受信した後、クエリマネージャ１２０は、データベース変更通知を構成要素に容易にルーティングし、かつ／または保存されている結果を無効にすることができる。本発明の一態様では、クエリマネージャ１２０が構成要素に通知し、構成要素は次のアクションを決定する。例えば、構成要素は、メモリマネージャ２２０を使用して保存されている結果を無効にすることができる。あるいは、クエリマネージャ１２０は、メモリマネージャ２２０を使用して、メモリマネージャ２２０は、保存されている結果を無効にし、かつ／または構成要素に通知することができる。保存されている結果が無効になると、クエリを開始した構成要素、および対応する保存はもはや存在しないことを理解されたい。

【００３４】

次に図３を参照すると、本発明の一態様によるデータベース登録／無効化システム３００の例を示している。データベース登録／無効化システム３００は、コンテキストバンク１１０、クエリマネージャ１２０、メモリマネージャ２２０、および任意選択でクライアント３１０、メモリ３２０を含む。複数のメモリマネージャ２２０、クライアント３１０、およびメモリ３２０を使用できるが、簡潔にするために、以下では１つのクライアントインスタンスについて説明することを理解されたい。

【００３５】

上述したように、コンテキストバンク１１０は、保存されている結果に影響を与えることになるデータベース変更が行われたとき、および／または有効期限が切れたときに保存されている結果を無効にするための情報を提供するために使用できる構成要素に関する（例えば上述した）無効化情報を格納することができる。本発明の一態様では、クライアント３１０は、無効化情報、つまり識別子、マシン名、ポート、アドレス、データベース名、一致しない保存結果を自動的に無効にする指示、一致しない結果を自動的に更新する指示、およびタイムアウト（データベース変更、ファイル変更、および時間による無効など）をコンテキストバンク１１０に提供することができる。

【００３６】

データベースに対するクエリを行うためにクライアント３１０がクエリマネージャ１２０を呼び出したとき、クエリマネージャ１２０は、（上述したように）コンテキストバンク１１０から関連する無効化情報を受信し、かつ／または関連する無効化情報をコンテキストバンク１１０から取得する（図示せず）。次いでクエリマネージャ１２０は、クエリを行い、無効化情報を提供する。クエリマネージャ１２０は、本発明の一態様に従って動的および／または静的データベースへのクエリを行うことができる。

【００３７】

クエリマネージャ１２０は、クエリ結果を受け付け、その結果をクライアント３１０に送信することができる。本発明は、同期および非同期メッセージ配信を提供することを理解されたい。同期メッセージ配信では、無効化メッセージの配信および／または一致しない結果の無効化まで、データベースの変更を阻止することができる。同期配信では、変更のコミットと無効化メッセージの受信および／または結果の無効化の間にデータベース変更が確実に行われないようにすることによって、データの不一致が軽減される。非同期メッセージ配信では、無効化メッセージを配信する前、および／または一定しない結果を無効にする前にデータベースの変更をコミットすることができる。非同期配信では、配信待ち時間が軽減される。

【００３８】

クエリ結果を受信した後に、クエリが行われたデータベースは、様々な技術を使用して検査され、クエリ後、格納すべき結果を生成する前に、変更が行われたかどうかを確認す

ることができる。変更が行われた場合、クライアント３１０は、メモリマネージャ２２０を呼び出すことなく結果を使用することができる。変更が行われていない場合、クライアント３１０は、その結果を使用し、メモリマネージャ２２０を使用して、その結果を（ローカル、リモート、高速、ＨＴＴＰ、共有などの）メモリ３２０に格納することができる。実質的に同様の結果を戻し得るその後の要求では、保存されている結果が一致しなくなるまでクエリを行う代わりに、メモリ３２０から保存されている結果を使用することができる。

【００３９】

本発明の別の態様では、結果を保存するかどうかの決定を容易にするために様々な技術が使用される。例えば、結果が複数の要求に共通であり、まれにしか変更されないと決定されると、その結果は保存される。しかし、結果が頻繁に変更され、かつ／または頻繁には使用されず、かつ保存のコストが比較的高いと決定されると、その結果は保存されない。本発明のさらに別の態様では、メモリ問題を軽減するために様々な技術が使用される。例えば、メモリ３２０がいっぱいである場合、結果は、先入れ先出し（ＦＩＦＯ）方式または先入れ後出し（ＦＩＬＯ）方式、最も古い情報が最初に破棄される経時ベース方式（ａｇｅ　ｂａｓｅｄ　ａｐｐｒｏａｃｈ）、最も使用されていない、または最後に使用されてから最も長い情報が最初に破棄される使用量方式（ｕｓａｇｅ　ａｐｐｒｏａｃｈ）、サイズベース方式、およびキーベース方式を使用して保存することができる。別の例では、十分な「空き」メモリを使用できるまで結果が保存されない。他の手法、またはこれらの手法のバリエーションでは、結果の生成に必要な時間量および／またはリソースを考慮することができる。

【００４０】

結果に影響を与えることになるデータベースへの変更が行われると、かつ／または有効期限が切れると、無効化メッセージがシステム３００にディスパッチされる。一般に、無効化メッセージは、無効化情報の少なくとも一部を含む。クエリマネージャ１２０は、無効化メッセージを受信することができる。次いでクエリマネージャ１２０は、クライアント３１０（および／または無効化メッセージを受信するように登録されている他の構成要素）に通知する。次いでクライアント３１０は、メモリマネージャ２２０を呼び出して、保存されている結果を無効にする（例えば削除、消去、上書き、および移動する）ことができる。無効化情報に自動結果更新が指示されている場合、その後のクエリが行われて新しい結果が生成され、これをメモリ３２０に保存し、クライアント３１０のような構成要素によって使用することができる。上記の技術を使用することによって、一致しない結果上に新しい結果を保存することによって一致しない結果の削除を軽減することができる。

【００４１】

無効化登録システム１００、無効化メモリ管理システム２００、およびデータベース登録／無効化システム３００は、さらに構成要素を使用して、データベース（データベーステーブル、データテーブル、テーブルなど）を登録する、データベースへの変更を検出する、変更されたデータベースが登録されているかどうかを決定する、および／または登録されているデータベースが変更されたことを示すメッセージをディスパッチすることができることを理解されたい。

【００４２】

図１～３は、システム１００～３００の構成要素を示すブロック図であるが、コンテキストバンク１１０、クエリマネージャ１２０、メモリマネージャ２２０、クライアント３１０、およびメモリ３２０は、用語が本明細書で定義されているような１つまたは複数のコンピュータ構成要素として実装できることを理解されたい。したがって、システム１００～３００、コンテキストバンク１１０、クエリマネージャ１２０、メモリマネージャ２２０、クライアント３１０、およびメモリ３２０を実装するよう動作可能なコンピュータ実行可能構成要素は、それだけには限定されないが、本発明に従って、ＡＳＩＣ（特定用途向け集積回路）、ＣＤ（ｃｏｍｐａｃｔ　ｄｉｓｋ）、ＤＶＤ（ｄｉｇｉｔａｌ　ｖｉｄｅｏ　ｄｉｓｋ）、ＲＯＭ（Ｒｅａｄ　ｏｎｌｙ　Ｍｅｍｏｒｙ）、フロッピー（登録

商標）ディスク、ハードディスク、ＥＥＰＲＯＭ（Ｅｌｅｃｔｒｏｎｉｃａｌｌｙ Ｅｒａｓａｂｌｅ ａｎｄ Ｐｒｏｇｒａｍｍａｂｌｅ ＲＯＭ）、およびメモリスティックなどのコンピュータ可読媒体に格納することができることを理解されたい。

【００４３】

図４に進むと、本発明の一態様によるＷｅｂベースメモリ無効化システム４００の例を示している。Ｗｅｂベースメモリ無効化システム４００は、Ｗｅｂサーバ４０５、依存構成要素（ｄｅｐｅｎｄｅｎｃｙ ｃｏｍｐｏｎｅｎｔ）４１０、コンテキスト構成要素４２０、クエリマネージャ４３０、サーバ４４０、データベースサーバ４５０、メモリストア４６０、ローカルメモリ４７０、高速メモリ４８０、他のメモリ４９０（リモート高速メモリなど）、アプリケーション４９２、および補助層４９４を含む。

【００４４】

Ｗｅｂサーバ４０５は、ページ指示（上述したようなコメントなど）を含むＷｅｂページにアクセスすることができる。ページ指示を使用して、Ｗｅｂ応答が保存されること、および関連するデータベースが変更されたとき、および／または有効期限が切れたときに保存された応答を無効化すべきであることを指定することができる。Ｗｅｂサーバ４０５は、ページ指示を使用し、依存構成要素４１０（オブジェクト、導出オブジェクトなど）を呼び出し、コンテキスト構成要素４２０への無効化情報の格納を容易にすることができる。

【００４５】

依存構成要素４１０は、例えばグローバル一意識別子、すなわちＧＵＩＤなどの一意の識別子を取得して、その後のクエリに関連付けるべきキャッシュされた応答を識別する。依存構成要素４１０は、一意の識別子および（上述したような）他の様々な無効化情報をコンテキスト構成要素４２０に提供する。一般に、無効化情報は、無効化文字列（例えばＧＵＩＤ、マシン名、ポート、アドレスおよびキュー名、任意選択で認証情報、暗号化、およびプロトコル優先順位）、サービス情報、および（データベースの変更および期限切れなどによる）タイムアウトを含む。しかし、上記は例であり、無効化情報を限定するものではない。

【００４６】

クエリマネージャ４３０（クエリマネージャ１１０など）が呼び出されてデータベースへのクエリを行うとき、クエリマネージャ４３０は、コンテキスト構成要素４２０とインターフェースをとり、関連する無効化情報を取得する。次いでクエリエンジン４２０は、クエリおよび無効化情報をデータベースサーバ４５０内のサーバ４４０（例えば動的および静的）に送信する。データベース４４０に対するクエリが実行されて結果がアプリケーション４９２に戻される。無効化情報は、データベースサーバ４５０に残り、Ｗｅｂサーバ４０５上の依存構成要素４１０と、サーバ４４０によって戻される結果との間の関連を提供する。

【００４７】

クエリ結果を受信した後、Ｗｅｂサーバは、完全応答または部分応答を生成する。Ｗｅｂサーバ４０５は、依存構成要素４１０をポーリングして、応答が生成されている間にデータベースの変更が行われたかどうかを決定し、次いでその結果を保存するかどうかを決定する。その結果を保存することが決定されると、Ｗｅｂサーバ４０５は、メモリストア４６０を使用して、結果をローカルメモリ４７０（キャッシュなど）、高速メモリ４８０（キャッシュなど）、および／または他のメモリ４９０への保存を容易にすることができる。一般に、ローカルメモリ４７０をアプリケーション４９０に関連付け、高速メモリ４８０を補助層４９４（ＨＴＴＰ．ｓｙｓなど）に関連付けることができる。そうでない場合、保存することなく応答を使用し、かつ／または破棄することができる。

【００４８】

保存されているＷｅｂ応答に影響を与えるデータベース変更が行われたとき、および／または有効期限が切れたとき、サーバ４４０は、無効化メッセージを依存構成要素４１０に送信する。次いで依存構成要素４１０は、（例えばイベントを発生させる、フラグを設

定するなどにより）変更を信号で知らせる。メモリストア４６０は、この信号を受信し、応答を無効にする。任意選択で、無効化メッセージが受信されると、自動更新を使用してクエリを実行して、一致した結果を取得し、保存することができる。自動更新は、一致しない結果を上書きし、かつ／または一致した結果を保存する前に一致しない結果を削除することができる。

【００４９】

次に図５に、本発明の一態様によるセキュリティ技術の例を示している。システム５００は、Ｗｅｂサーバ４０５、データベースサーバ４５０、およびＷｅｂサーバ４９６とデータベースサーバ４５０を動作可能に結合するセキュリティ層５１０を含む。

【００５０】

上述したように、クエリエンジン４３０は、コンテキスト構成要素４２０とインターフェースをとり、クエリ要求を受信した後、無効化情報を取得する。次いでクエリエンジン４２０は、Ｗｅｂサーバ４０５からデータベースサーバ４５０内のサーバ４４０にクエリおよび無効化情報を送信する。

【００５１】

上述したように、無効化情報は、グローバル一意識別子、すなわちＧＵＩＤなどの一意の識別子を含むことができる。ＧＵＩＤは、データベースの変更および有効期限切れをシミュレートしようとする悪意のある試みを軽減するために使用することができるセキュリティ機構を提供する。例えば、本発明の一態様では、ランタイム時のデータベースクエリ登録要求が受信されるときにＧＵＩＤを作成し、それによってＧＵＩＤを「推測する」または模倣する機会が軽減される。次いで無効にすべき保存結果の組を識別するためにＧＵＩＤが使用される。

【００５２】

ＧＵＩＤが無効である（例えば無効にすべき保存結果の組を識別しない）と決定されると、無効化要求を無視し、かつ／または例えばＧＵＩＤとともに破棄することができる。別の例では、偽りの無効化要求をシステム管理者に転送して、発信位置を追跡するために使用することができる。

【００５３】

図６は、本発明の一態様による無効化メッセージ転送技術の例を示している。システム６００は、Ｗｅｂサーバ４０５、データベースサーバ４５０、ならびにＷｅｂサーバ４０５およびデータベースサーバ４５０を動作可能に結合する通信インターフェース６１０を含む。

【００５４】

クエリ要求を受信すると、クエリエンジン４３０は、コンテキスト構成要素４２０から無効化情報を取得し、クエリおよび無効化情報をデータベースサーバ４５０に送信する。クエリが実行されて結果がＷｅｂサーバ４０５に送信され、この結果をメモリに保存し、複数回使用することができ、実質的に同様のクエリの実行を軽減することができる。

【００５５】

データベースサーバ４５０から結果が戻されると、無効化情報は、データベースサーバ４５０に残り、Ｗｅｂサーバ４０５内で関連する保存結果を探すためにデータベース変更が行われたときに使用することができる。データベースサーバ４５０内に無効化情報を格納するために様々な機構を使用できること、および無効化情報を変更されたデータベースと照合し、無効化メッセージをディスパッチするために様々な技術を使用できることを理解されたい。

【００５６】

データベースの変更が行われると、無効化メッセージが構築され、通信インターフェース６１０を介してデータベースサーバ４５０からＷｅｂサーバ４９６に送信される。無効化メッセージは、例えば無効化情報の少なくとも一部を含むことができる。

【００５７】

データベースの変更は、コミットおよび／または阻止することができることを理解され

たい。例えば、本発明の一態様で、非同期無効化メッセージ転送モデルを使用することができ、データベース変更は、無効化メッセージが通信インターフェース６１０を介して送信されるかどうか、および／または保存されている結果が無効化されたかどうかに関係なくコミットされる。

【００５８】

本発明の別の態様で、同期無効化メッセージ転送モデルを使用することができ、無効化メッセージが通信インターフェース６１０を介して送信されるまで、および／または保存されている結果が無効化されるまでデータベースの変更が阻止される。非同期モデルは拡張性を提供し、同期モデルは信頼性を提供する。上記は理解しやすくするために提供したものであり、本発明を限定するものではなく、例えば、本発明に従って他の様々なプッシュおよび／またはプル手法など他の技術を使用することもできる。

【００５９】

図７は、本発明の一態様による分散無効化システム７００の例を示している。無効化システム７００は、Ｎを１以上の整数とすると、第１のＷｅｂアプリケーション710_1からＮ番目のＷｅｂアプリケーション710_N、Ｍを１以上の整数とすると、第１のコンテキスト構成要素720_1からＭ番目のコンテキスト構成要素710_M、Ｋを１以上の整数とすると、第１のデータベース730_1からＫ番目のデータベース730_K、および共有メモリ７４０を含む。

【００６０】

第１のＷｅｂアプリケーション710_1からＮ番目のＷｅｂアプリケーション710_NをまとめてＷｅｂアプリケーション７１０、第１のコンテキスト構成要素720_1からＭ番目のコンテキスト構成要素720_Mをまとめてコンテキスト構成要素７２０、第１のデータベース730_1からＫ番目のデータベース730_Kをまとめてデータベース７３０と呼ぶことができる。

【００６１】

Ｗｅｂアプリケーション７１０（オブジェクトインスタンスなど）を使用してデータベースへのクエリを行い、ユーザ要求を満たすことができる。一般に、Ｗｅｂアプリケーション７１０がインスタンス化されると、Ｗｅｂアプリケーション７１０に関連する無効化情報は、アセンブルされ、それぞれのコンテキスト構成要素７２０（コンテキストバンク１１０など）に格納される。上述したように、無効化情報は一般に、一意の識別子、無効化文字列、サービス情報、および有効期限を含むが、無効化情報はそのように限定されるものではない。

【００６２】

クエリ要求が少なくとも１つのＷｅｂアプリケーション７１０で受信されると、対応するコンテキストバンクを使用して関連する無効化情報が取得される。次いでクエリおよび無効化情報を、通信プロトコル（Ｅｔｈｅｒｎｅｔ（登録商標）やＦｉｒｅｗｉｒｅなど）を介してデータベース７３０に送信する。クエリは、ユーザ要求を満たすために１つまたは複数のデータベース７３０を使用できることに注意されたい。

【００６３】

クエリを実行した後、結果を少なくとも１つのＷｅｂアプリケーション７１０に戻すことができる。結果が確実に一致するようにするために、データベース７３０のチェックを行い、クエリの後、しかし格納すべき結果を生成する前に結果に影響を与えることになる変更が行われたかどうかを決定することができる。変更が行われたと決定された場合、その結果を使用し、かつ／または破棄することができる。結果が一致すると決定された場合、その結果を使用し、かつ／または共有メモリ７４０などのメモリに保存することができる。保存されている結果は、１回または複数回使用することが可能である。さらに、１つまたは複数のＷｅｂアプリケーション７１０は、保存されている結果を連続的かつ／または同時に使用することができる。したがって、本発明は、複数のＷｅｂアプリケーション７１０が実質的に同様の結果についてのクエリを行うのをさらに軽減する。

【００６４】

少なくとも１つのデータベース７２０に対してデータベース変更が行われたとき、および／または有効期限が切れたとき、少なくとも１つのデータベース７２０は、無効化メッセージを１つまたは複数のＷｅｂアプリケーション７１０および／または共有メモリ７４０に送信する。（例えば１つの）無効化メッセージは、変更によって影響を受ける複数のＷｅｂアプリケーションに通知するために、複数のＷｅｂアプリケーションアドレス、および／または他の情報（他の無効化情報など）を含むことができることを理解されたい。

【００６５】

本発明の別の態様では、複数の無効化メッセージを送信することができ、無効化メッセージは、Ｗｅｂアプリケーションおよび／または共有メモリ７４０に関連付けられ、そこに送信される。さらに別の態様では、実質的にすべてのＷｅｂアプリケーション７１０および／または共有メモリ７４０が無効化メッセージを受信できるように、無効化メッセージをブロードキャストすることができる。次いでＷｅｂアプリケーション７１０は、無効化メッセージを受け付け、かつ／またはそれに対して作用するかどうかを決定することができる。無効化メッセージを共有メモリ７４０に提供することによって、無効化メッセージをＷｅｂアプリケーションに提供する必要性を軽減できることを理解されたい。例えば、要求を開始したＷｅｂアプリケーションは、保存されている結果を無効にするために存在している必要はない。さらに、保存されている結果を共有した任意のアプリケーションは、保存されている結果を無効にするために存在している必要はない。したがって、本発明は、開始したアプリケーションおよび／または共有した任意のアプリケーションがもはや存在しない場合でさえ、一致しない結果を破棄し、かつ／または共有メモリを解放するための機構を提供する。

【００６６】

無効化メッセージを受信した後、次いで保存されている結果を無効にすることができる。例えば、保存されている結果をメモリから削除し、または不一致としてそれにタグ付けすることができる。任意選択で、自動更新を使用してその後のクエリを実行して一致した結果を取得し、メモリに保存することができる。自動更新はさらに、それらが一致しなくなると更新結果を自動的に無効にするために、その後のクエリとともに無効化情報を再送信することができる。

【００６７】

図８は、本発明の一態様によるＷｅｂサービス無効化システム８００の例を示している。Ｗｅｂサービス無効化システム８００は、クライアント８０５、ファイアウォール８１０、メモリ８２０、Ｎを１以上の整数とすると、第１のＷｅｂサービス８３０_{1}からＮ番目のＷｅｂサービス８３０_{N}、およびデータベース８４０を含む。第１のＷｅｂサービス８３０_{1}からＮ番目のＷｅｂサービス８３０_{N}をまとめてＷｅｂサービス８３０と呼ぶことができる。

【００６８】

クライアント８０５は、ファイアウォール８１０を介してＷｅｂサービス８３０とインターフェースをとる。Ｗｅｂサービス８３０へのアクセスは、ファイアウォール８１０によって制御することができる。例えば、ファイアウォール８１０は、クライアント８０５によるＷｅｂサービス８３０へのアクセスを限定する、かつ／またはアクセスを拒否することができる。さらに、ファイアウォール８１０を使用して、クライアント８０５によるデータベース８４０およびメモリ８２０へのアクセスを限定することができる。Ｗｅｂサーバ８３０、データベース８４０、および／またはメモリ８２０へのクライアントアクセスを相互排他的にすることができることを理解されたい。例えば、クライアント８０５は、データベース８４０および／またはメモリ８２０にアクセスすることなくＷｅｂサーバ８３０にフルアクセスすることができる。しかしクライアント８０５は、Ｗｅｂサーバ８３０、データベース８４０、および／またはメモリ８２０にフルアクセスできること、またはアクセスできないことがある。

【００６９】

さらにファイアウォール８１０を使用して、データベース８４０および／またはメモリ

８２０に保存されている結果へのＷｅｂサービスアクセスを限定することができる。本発明の一態様では、１つまたは複数のＷｅｂサービス８３０は、（例えばクライアント８０５によって開始された）クエリ、（例えば上述した）無効化情報、およびルーティング情報など他の情報をデータベース８４０に送信することができる。１つまたは複数のＷｅｂサービス８３０が使用許可（ｃｌｅａｒａｎｃｅ）を有している場合、送信された情報をデータベース８４０に伝達することができる。本発明の別の態様では、１つまたは複数のＷｅｂサービス８３０は、ファイアウォール８１０を介して情報を伝達することなく、クエリ、無効化情報、および他の情報をデータベース８４０に送信することができる。

【００７０】

クエリが実行された後、データベース８４０は、クエリ結果を戻す。これは、格納すべき結果の生成に使用される。本発明の一態様では、この結果がファイアウォール８１０に戻され、ファイアウォール８１０は、ある機構（メモリマネージャ１２０など）を使用してその結果をメモリ８２０に保存する。本発明の別の態様では、（例えばファイアウォール８１０を通る、および／または回避することによって）結果をＷｅｂサービス８３０に戻すことができる。次いでＷｅｂサービス８３０は、この結果をメモリ８２０に格納することができ、そこでこの結果はメモリ８２０へのアクセス権を有するＷｅｂサービス８３０から使用でき、この結果をクライアント８０５に提供することができる。

【００７１】

したがって保存されている結果には、メモリ８２０の使用許可を有しているＷｅｂサービス８３０からアクセスすることができる。例えば、Ｗｅｂサービス８３０_{N}は、メモリ８２０内に保存されている結果と実質的に同じ、かつ／またはそのサブセットである結果を戻すクエリを送信することができる。Ｗｅｂサービス８３０_{N}は、メモリ８２０にアクセスすることができる場合、クエリを実行する代わりに、保存されている結果を使用することができる。Ｗｅｂサービス８３０_{N}がメモリ８２０にアクセスできない場合、クエリが実行される。

【００７２】

メモリ８４０に保存されている結果に影響を及ぼすデータベース変更が行われたとき、および／または有効期限が切れたとき、データベース８４０は、無効化メッセージをファイアウォール８１０に送信することができる。無効化メッセージが使用許可を有している場合、メッセージを使用して、メモリ８２０内の一致しない結果を無効にすることができる。本発明の一態様では、通知をさらにＷｅｂサービス８３０に送信して、Ｗｅｂサービス８３０に結果が無効になったことを通知することができる。別の態様では、通知をＷｅｂサービス８３０に送信する代わりに、Ｗｅｂサービス８３０がメモリ８２０を定期的にポーリングして、例えば依然として結果が一致しているかどうかを決定することができる。別の例では、Ｗｅｂサービス８３０がクエリおよび無効化情報を送信すると、同様の結果がメモリ８２０に保存されているか、また同様の結果が一致しているかどうかを決定するためにメモリ８２０で検索が行われる。

【００７３】

自動更新を使用して、一致した結果についての再クエリを自動的に行い、次のデータベース変更が行われたとき、および／またはデータベースのデータ（結果）に対応する期限が切れたときに無効にするための無効化情報を再送信することができる。

【００７４】

図９～１２は、本発明によるいくつかの方法を示している。説明を簡単にするために、これらの方法を、一連の動作として示し、記載している。本発明は、例示の動作によって、および／または動作の順序によって限定されるものではなく、例えば、動作を様々な順序で、かつ／または同時に、また本明細書に提示、記載していない他の動作とともに行うことができることを理解されたい。さらに、本発明による方法の実施にすべての動作例が必要であるとは限らない。さらに、ある方法を代わりに相互に関連のある一連の状態またはイベントとして表すことができる（状態図など）ことを当分野の技術者であれば理解されよう。

【００７５】
図９に進むと、自動無効化方法９００の例を示している。参照符号９１０で、あるインスタンスがインスタンス化される（オブジェクト、導出オブジェクト、クライアント、アプリケーション、クライアントアプリケーション、および依存インスタンスなど）。９２０で、インスタンスに関連する無効化情報がコンテキストに格納される。上述したように、無効化情報は一般に、無効化文字列、サービス情報、およびタイムアウトを含む。複数のインスタンスをインスタンス化することができ、また複数のインスタンスは、無効化情報を同様のコンテキストに格納することができることを理解されたい。
【００７６】
参照符号９３０で、方法は、クエリ実行要求の受信を待つ。要求を受信すると、コンテキストに格納されており、インスタンスと関連する無効化情報が取り出される。９４０で、クエリおよび無効化情報が例えばデータベースが存在する動的および／または静的サーバに送信される。
【００７７】
次に図１０を参照すると、本発明の一態様による図９の続きを示している。参照符号１０１０に進むと、方法９００は、クエリの結果を待つ。結果を受信し、処理した後、１０２０で、クエリに使用されるデータベースが検査され、クエリとキャッシュ可能な結果の生成との間に変更が行われたかどうかを確認する。これによって、確実に結果が一致するようになる。
【００７８】
クエリと結果の受信との間に変更が行われなかった場合、１０３０でその結果がメモリに格納され、１０４０でインスタンスに提供される。保存されている結果をその後使用して、例えば同様の結果を戻し得るクエリの実行を軽減することができる。１０２０で、クエリと格納すべき結果の生成の間に変更が行われたと決定された場合、１０４０で、その結果は一般にインスタンスに提供され、保存されない。しかし、インスタンスは、一致しない結果を格納し、使用することを決定することができる。結果が一致しないことを他のインスタンスに通知するために様々な技術を使用することができる。例えば、インスタンスが一致しない結果の使用を試みるときはいつでもイベントを生成させるようにフラグを設定することができる。
【００７９】
次に図１１に、本発明の一態様によるデータ変更ベース無効化方法１１００の例を示している。１１１０に進むと、データベース変更が行われ、無効化メッセージが送信され、受信される。無効化メッセージは一般に、無効化情報の一部を含む。例えば、一意の識別子を含めて、無効にすべき一致しない結果の組の識別を容易にすることができる。
【００８０】
データベース変更および送受信を同期することができ、無効化メッセージが受信され、かつ／またはそれに対して行動するときにデータベース変更がコミットされることを理解されたい。別の例では、非同期技術を使用して、送信が行われた、および／またはメッセージを受信したかどうかに関係なくデータベースの変更がコミットされる。
【００８１】
１１２０で、保存されている結果が無効になる。結果を無効にするために様々な方法を使用できることを理解されたい。例えば、結果をメモリから削除する、無効としてタグ付けする、かつ／または上書きすることができる。１１３０で、自動更新を行い、現在の一致した結果を得るためにクエリを行い、その後のデータベースの変更のために無効化メッセージを再登録することができる。
【００８２】
次に図１２を参照すると、本発明の一態様による時間ベース無効化方法の別の例を示している。１２１０に進むと、無効化情報に含められている期間が満了する。その後、無効化メッセージが送信され、受信される。無効化メッセージは、無効化情報の一部を含むことができる。例えば、一意の識別子を含めて、無効にすべき一致しない結果の組の識別を

容易にすることができる。１２２０で、（例えば上記と同じように）保存されている結果が無効になる。１２１０で、自動更新を行い、現在の一致した結果を得るためにクエリを行い、その後の有効期限のために無効化メッセージを再登録することができる。

【００８３】

本発明の様々な態様に関する状況をさらに提供するために、図１３および以下の説明は、本発明の様々な態様を実施できる適した動作環境１３１０の簡単な概略説明を提供するためのものである。本発明は、１つまたは複数のコンピュータまたは他の装置が実行するプログラムモジュールなどのコンピュータ実行可能命令の一般的な状況で説明するが、本発明は、他のプログラムモジュールとの組合せで、および／またはハードウェアおよび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を当分野の技術者であれば理解されよう。しかし、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、または特定のデータ型を実装するルーチン、プログラム、オブジェクト、構成要素、データ構造などを含む。動作環境１３１０は、適した動作環境の一例にすぎず、本発明の使用または機能の範囲に関する限定を示唆するものではない。本発明との使用に適した他のよく知られているコンピュータシステム、環境、および／または構成には、それだけには限定されないが、パーソナルコンピュータ、ハンドヘルドまたはラップトップ装置、マルチプロセッサシステム、マイクロプロセッサベースのシステム、プログラム可能家庭用電化製品、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、上記の任意のシステムまたは装置を含む分散コンピューティング環境などがある。

【００８４】

図１３を参照すると、本発明の様々な態様を実施する環境１３１０の例は、コンピュータ１３１２を含んでいる。コンピュータ１３１２は、処理ユニット１３１４、システムメモリ１３１６、およびシステムバス１３１８を含む。システムバス１３１８は、それだけには限定されないが、システムメモリ１３１６などのシステム構成要素を処理ユニット１３１４に結合する。処理ユニット１３１４は、使用可能な様々なプロセッサのうちのどんなものでもよい。デュアルマイクロプロセッサおよび他のマルチプロセッサアーキテクチャも処理ユニット１３１４として使用することができる。

【００８５】

システムバス１３１８は、使用可能な様々なバスアーキテクチャのうちの任意のものを使用するメモリバスまたはメモリコントローラ、周辺バスまたは外部バス、および／またはローカルバスを含むいくつかのタイプのバス構造のうちどんなものでもよい。こうしたアーキテクチャには、それだけには限定されないが、８ビットバスのＩＳＡ（Ｉｎｄｕｓｔｒｉａｌ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ）、ＭＳＡ（Ｍｉｃｒｏ Ｃｈａｎｎｅｌ Ａｒｃｈｉｔｅｃｔｕｒｅ）、拡張ＩＳＡ（Ｅｘｔｅｎｄｅｄ ＩＳＡ）、ＩＤＥ（Ｉｎｔｅｌｌｉｇｅｎｔ Ｄｒｉｖｅ Ｅｌｅｃｔｒｏｎｉｃｓ）、ＶＥＳＡローカルバス（ＶＬＢ）、ＰＣＩ（Ｐｒｅｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ）、ＵＳＢ（Ｕｎｉｖｅｒｓａｌ Ｓｅｒｉａｌ Ｂｕｓ）、ＡＧＰ（Ａｄｖａｎｃｅｄ Ｇｒａｐｈｉｃｓ Ｐｏｒｔ）、ＰＣＭＣＩＡ（Ｐｅｒｓｏｎａｌ Ｃｏｍｐｕｔｅｒ Ｍｅｍｏｒｙ Ｃａｒｄ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ａｓｓｏｃｉａｔｉｏｎ）バス、およびＳＣＳＩ（Ｓｍａｌｌ Ｃｏｍｐｕｔｅｒ Ｓｙｓｔｅｍｓ Ｉｎｔｅｒｆａｃｅ）などがある。

【００８６】

システムメモリ１３１６は、揮発性メモリ１３２０および不揮発性メモリ１３２２を含む。基本入出力システム（ＢＩＯＳ）は、例えば起動中など、コンピュータ１３１２内の要素間での情報を転送するための基本ルーチンを含み、不揮発性メモリ１３２２に格納されている。不揮発性メモリ１３２２には、それだけには限定されないが一例として、ＲＯＭ（ｒｅａｄ ｏｎｌｙ ｍｅｍｏｒｙ）、ＰＲＯＭ（ｐｒｏｇｒａｍｍａｂｌｅ ＲＯＭ）、ＥＰＲＯＭ（ｅｌｅｃｔｒｉｃａｌｌｙ ｐｒｏｇｒａｍｍａｂｌｅ ＲＯＭ）、ＥＥＰＲＯＭ（ｅｌｅｃｔｒｉｃａｌｌｙ ｅｒａｓａｂｌｅ ＲＯＭ）、フラッシュメモリなどがある。揮発性メモリ１３２０には、ランダムアクセスメモリ（ＲＡＭ）などが

あり、これは外部キャッシュメモリとして働く。ＲＡＭは、それだけには限定されないが一例として、ＳＲＡＭ（ｓｙｎｃｈｒｏｎｏｕｓ　ＲＡＭ）、ＤＲＡＭ（Ｄｙｎａｍｉｃ　ＲＡＭ）、ＳＤＲＡＭ（Ｓｙｎｃｈｒｏｎｏｕｓ　ＤＲＡＭ）、ＤＤＲ　ＳＤＲＡＭ（ｄｏｕｂｌｅ　ｒａｔｅ　ＳＤＲＡＭ）、ＥＳＤＲＡＭ（ｅｎｈａｎｃｅｄ　ＳＤＲＡＭ）、ＳＬＤＲＡＭ（Ｓｙｎｃｈｌｉｎｋ　ＤＲＡＭ）、およびＤＲＲＡＭ（ｄｉｒｅｃｔ　Ｒａｍｂｕｓ　ＲＡＭ）など多くの形態で使用可能である。

【００８７】

コンピュータ１３１２は、リムーバブル／非リムーバブル、揮発性／不揮発性コンピュータ記憶媒体を含むこともできる。図１３は、例えばディスク記憶装置１３２４を示す。ディスク記憶装置１３２４には、それだけには限定されないが、磁気ディスクドライブ、フロッピー（登録商標）ディスクドライブ、テープドライブ、Ｊａｚドライブ、Ｚｉｐドライブ、ＬＳ－１００ドライブ、フラッシュメモリカード、メモリスティックなどの装置がある。さらに、ディスク記憶装置１３２４は、記憶媒体を別個に、あるいは、それだけには限定されないが、コンパクトディスクＲＯＭ装置（ＣＤ－ＲＯＭ）、ＣＤレコーダブルドライブ（ＣＤ－Ｒ　Ｄｒｉｖｅ）、ＣＤリライタブルドライブ（ＣＤ－ＲＷ　Ｄｒｉｖｅ）、またはデジタル多用途ディスクＲＯＭドライブ（ＤＶＤ－ＲＯＭ）などの光ディスクドライブなど他の記憶媒体と組み合わせて含めることができる。ディスク記憶装置１３２４のシステムバス１３１８への接続を容易にするために、一般に、インターフェース１３２６などリムーバブルまたは非リムーバブルインターフェースが使用される。

【００８８】

図１３は、ユーザと適した動作環境１３１０に記載した基本的なコンピュータリソースとの間で媒介として働くソフトウェアを説明していることを理解されたい。こうしたソフトウェアには、オペレーティングシステム１３２８などがある。オペレーティングシステム１３２８は、ディスク記憶装置１３２４に格納することができ、コンピュータシステム１３１２のリソースを制御し、割り当てるよう働く。システムアプリケーション１３３０は、システムメモリ１３１６またはディスク記憶装置１３２４のいずれかに格納されているプログラムモジュール１３３２およびプログラムデータ１３３４を介してオペレーティングシステム１３２８によるリソースの管理を利用する。本発明は様々なオペレーティングシステム、またはオペレーティングシステムの組合せで実施できることを理解されたい。

【００８９】

ユーザは、入力装置１３３６を介してコマンドまたは情報をコンピュータ１３１２に入力する。入力装置１３３６には、それだけには限定されないが、マウスなどのポインティング装置、トラックボール、スタイラス、タッチパッド、キーボード、マイクロフォン、ジョイスティック、ゲームパッド、衛星パラボラアンテナ、スキャナ、ＴＶチューナカード、デジタルカメラ、デジタルビデオカメラ、Ｗｅｂカメラなどがある。これらおよび他の入力装置は、インターフェースポート１３３８を経由してシステムバス１３１８によって処理ユニット１３１４に接続される。インターフェースポート１３３８には、例えば、シリアルポート、パラレルポート、ゲームポート、ユニバーサルシリアルバス（ＵＳＢ）などがある。出力装置１３４０は、入力装置１３３６と同じタイプの何らかのポートを使用する。したがって、例えばＵＳＢポートを使用して、コンピュータ１３１２への入力を提供し、コンピュータ１３１２から出力装置１３４０に情報を出力することができる。出力アダプタ１３４２は、出力装置１３４０の中でも一部の出力装置１３４０にはモニタ、スピーカー、プリンタなど特殊なアダプタを必要とするものがあることを示すために提供されている。出力アダプタ１３４２には、それだけには限定されないが一例として、出力装置１３４０とシステムバス１３１８の間の接続手段を提供するビデオカードおよびサウンドカードなどがある。リモートコンピュータ１３４４など、他の装置および／または装置のシステムは、入力および出力の機能を提供することに注意されたい。

【００９０】

コンピュータ１３１２は、リモートコンピュータ１３４４など１つまたは複数のリモー

トコンピュータへの論理接続を使用してネットワーク式環境で動作することができる。リモートコンピュータ１３４４は、パーソナルコンピュータ、サーバ、ルーター、ネットワークＰＣ、ワークステーション、マイクロプロセッサベースの装置、ピア装置、または他の一般のネットワークノードなどでよく、一般にコンピュータ１３１２に関連して記載した多くのまたはすべての要素を含む。簡潔にするために、リモートコンピュータ１３４４とともにメモリ記憶装置１３４６のみを示している。リモートコンピュータ１３４４は、ネットワークインターフェース１３４８を介してコンピュータ１３１２に論理的に接続され、次いで通信接続１３５０を介して物理的に接続される。ネットワークインターフェース１３４８は、ローカルエリアネットワーク（ＬＡＮ）および広域エリアネットワーク（ＷＡＮ）などの通信ネットワークを含む。ＬＡＮ技術は、ＦＤＤＩ（Ｆｉｂｅｒ Ｄｉｓｔｒｉｂｕｔｅｄ Ｄａｔａ Ｉｎｔｅｒｆａｃｅ）、ＣＤＤＩ（Ｃｏｐｐｅｒ Ｄｉｓｔｒｉｂｕｔｅｄ Ｄａｔａ Ｉｎｔｅｒｆａｃｅ）、Ｅｔｈｅｒｎｅｔ（登録商標）／ＩＥＥＥ８０２．３、トークンリング／ＩＥＥＥ８０２．５などがある。ＷＡＮ技術には、それだけには限定されないが、ポイントツーポイントリンク、サービス総合デジタル網（ＩＳＤＮ）およびそのバリエーションなどの回線交換ネットワーク、パケット交換ネットワーク、デジタル加入者回線（ＤＳＬ）などがある。

【００９１】

通信接続１３５０は、ネットワークインターフェース１３４８をバス１３１８に接続するために使用されるハードウェア／ソフトウェアを指す。通信接続１３５０は、明瞭にするためにコンピュータ１３１２内に示しているが、コンピュータ１３１２の外部にあってもよい。ネットワークインターフェース１３４８への接続に必要なハードウェア／ソフトウェアには、一例にすぎないが、通常の電話用のモデム、ケーブルモデム、ＤＳＬモデムなどのモデム、ＩＳＤＮアダプタ、Ｅｔｈｅｒｎｅｔ（登録商標）カードなど内部技術、外部技術がある。

【００９２】

上記で説明してきたことは、本発明の例を含む。当然、本発明を説明するために構成要素または方法の予想されるすべての組合せについて説明することは不可能であるが、本発明の他の多くの組合せおよび置換えが可能であることを当分野の技術者であれば理解できよう。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こうしたすべての代替形態、修正形態、および変形形態を含むものとする。さらに、「含む」という用語が詳細な説明または特許請求の範囲で使用されている限り、こうした用語は、請求項で移行語として使用されるときに「備える」が解釈されるように、「備える」という用語を包含しているものとする。

【図面の簡単な説明】

【００９３】

【図１】本発明の一態様による無効化登録システムの例を示す図である。
【図２】本発明の一態様による無効化メモリ管理システムの例を示す図である。
【図３】本発明の一態様によるデータベース登録／無効化システムの例を示す図である。
【図４】本発明の一態様によるＷｅｂベースメモリ無効化システムの例を示す図である。
【図５】本発明の一態様によるセキュリティ技術の例を示す図である。
【図６】本発明の一態様による無効化メッセージ転送技術の例を示す図である。
【図７】本発明の一態様による分散無効化システムの例を示す図である。
【図８】本発明の一態様によるＷｅｂサービス無効化システムの例を示す図である。
【図９】本発明の一態様による自動無効化方法の例を示す図である。
【図１０】本発明の一態様による図９の続きを示す図である。
【図１１】本発明の一態様による変更ベース無効化方法の例を示す図である。
【図１２】本発明の一態様による時間ベース無効化方法の例を示す図である。
【図１３】本発明の一態様によるオペレーティングシステムの例を示す図である。

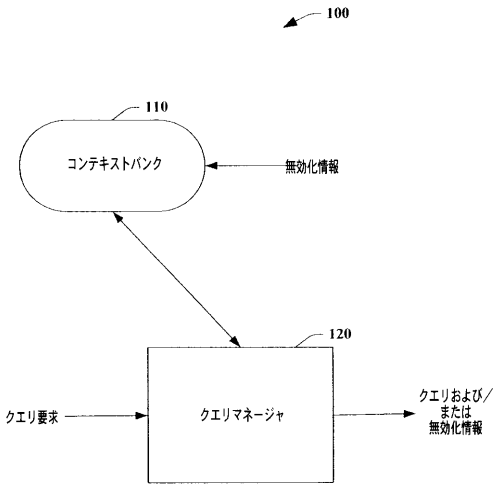
【符号の説明】

【００９４】

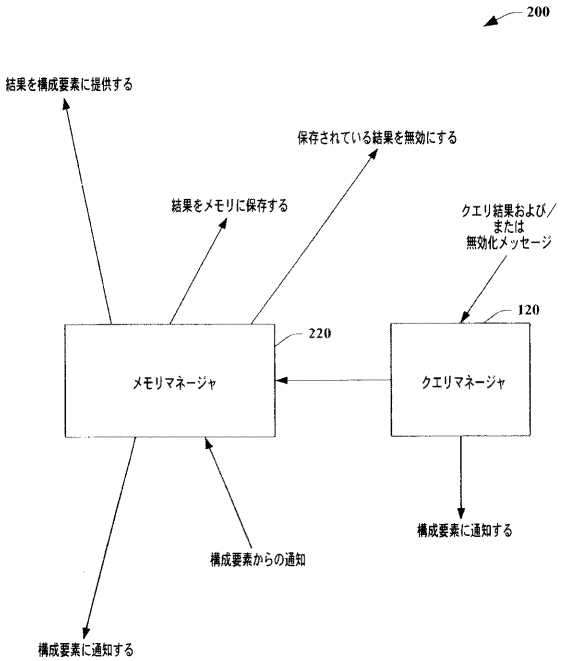
１００　無効化登録システム
１１０　コンテキストバンク
１２０　クエリマネージャ
２００　無効化メモリ管理システム
２２０　メモリマネージャ
３００　データベース登録／無効化システム
３１０　クライアント
３２０　メモリ
４００　Ｗｅｂベースメモリ無効化システム
４０５　Ｗｅｂサーバ
４１０　依存構成要素
４２０　コンテキスト構成要素
４３０　クエリマネージャ
４４０　サーバ
４５０　データベースサーバ
４６０　メモリストア
４７０　ローカルメモリ
４８０　高速メモリ
４９０　他のメモリ
４９２　アプリケーション
４９４　補助層
４９６　Ｗｅｂサーバ
５００　システム
５１０　セキュリティ層
６００　システム
６１０　通信インターフェース
７００　分散無効化システム
７１０_{1}～７１０_{N}　第１のＷｅｂアプリケーション～Ｎ番目のＷｅｂアプリケーション
７２０_{1}～７２０_{M}　第１のコンテキスト構成要素～Ｍ番目のコンテキスト構成要素
７３０_{1}～７３０_{K}　第１のデータベース～Ｋ番目のデータベース
７４０　共有メモリ
８００　Ｗｅｂサービス無効化システム
８０５　クライアント
８１０　ファイアウォール
８２０　メモリ
８３０_{1}～８３０_{N}　第１のＷｅｂサービス～Ｎ番目のＷｅｂサービス
８４０　データベース
９００　自動無効化方法
１１００　データ変更ベース無効化方法
１３１０　動作環境
１３１２　コンピュータ
１３１４　処理ユニット
１３１６　システムメモリ
１３１８　システムバス
１３２０　揮発性メモリ
１３２２　不揮発性メモリ
１３２４　ディスク記憶装置
１３２６　インターフェース
１３２８　オペレーティングシステム

１３３０　システムアプリケーション
１３３２　プログラムモジュール
１３３４　プログラムデータ
１３３６　入力装置
１３３８　インターフェースポート
１３４０　出力装置
１３４２　出力アダプタ
１３４４　リモートコンピュータ
１３４６　メモリ記憶装置
１３４８　ネットワークインターフェース
１３５０　通信接続

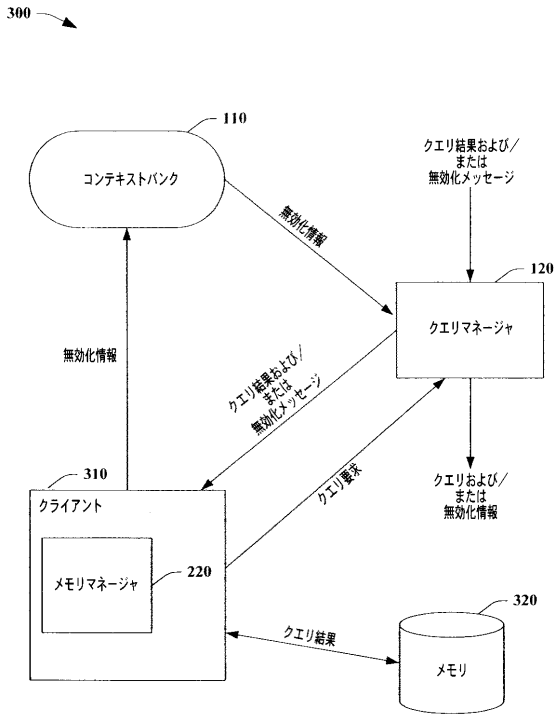
【図１】



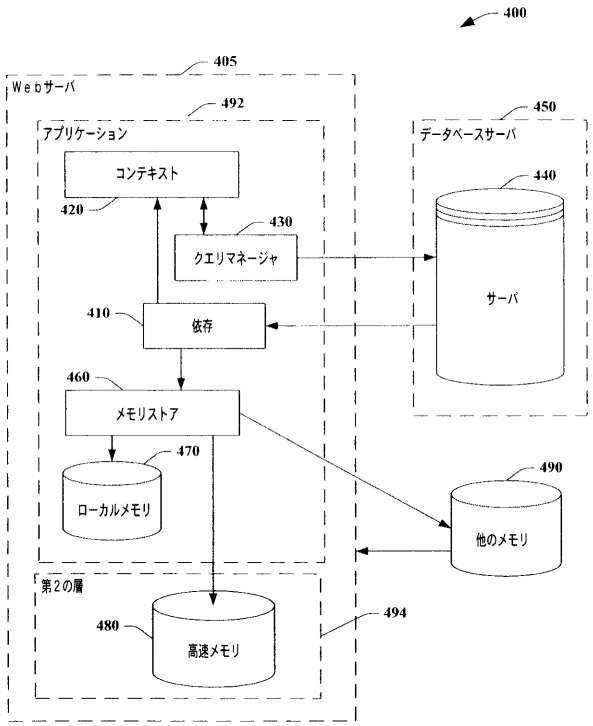
【図２】



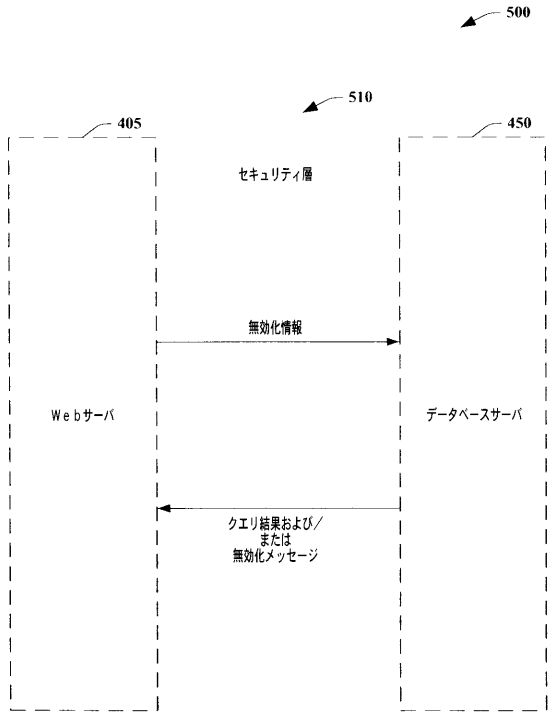
【図３】


300
110
コンテキストバンク
無効化情報
クエリ結果および／
または
無効化メッセージ
120
クエリマネージャ
無効化情報
クエリ結果および／
または
無効化メッセージ
310
クライアント
メモリマネージャ
220
クエリ要求
クエリおよび／
または
無効化情報
クエリ結果
320
メモリ

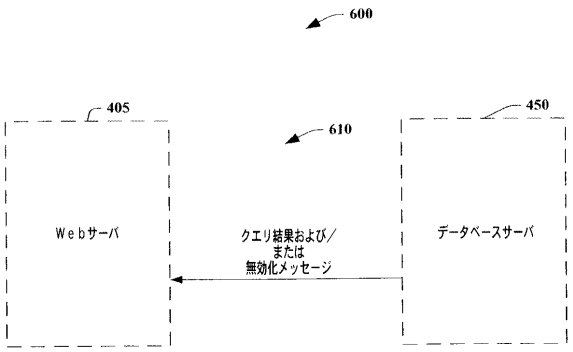
【図４】


400
405
Webサーバ
492
アプリケーション
コンテキスト
420
430
クエリマネージャ
410
依存
460
メモリストア
470
ローカルメモリ
第2の層
480
高速メモリ
494
450
データベースサーバ
440
サーバ
490
他のメモリ

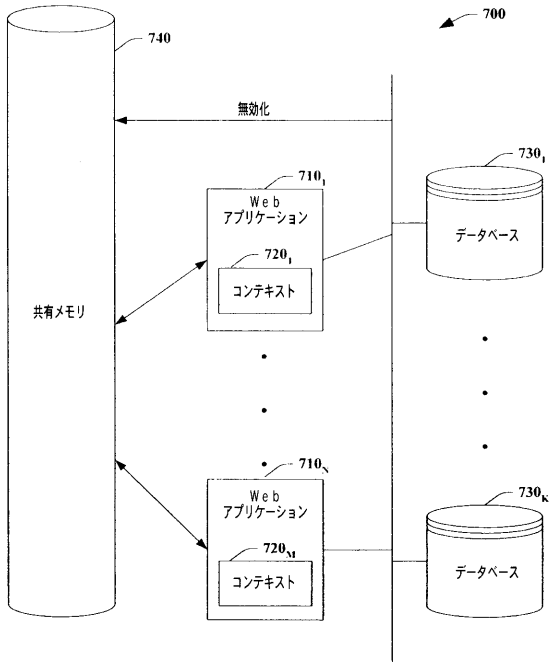
【図５】


500
510
405
Webサーバ
セキュリティ層
450
データベースサーバ
無効化情報
クエリ結果および／
または
無効化メッセージ

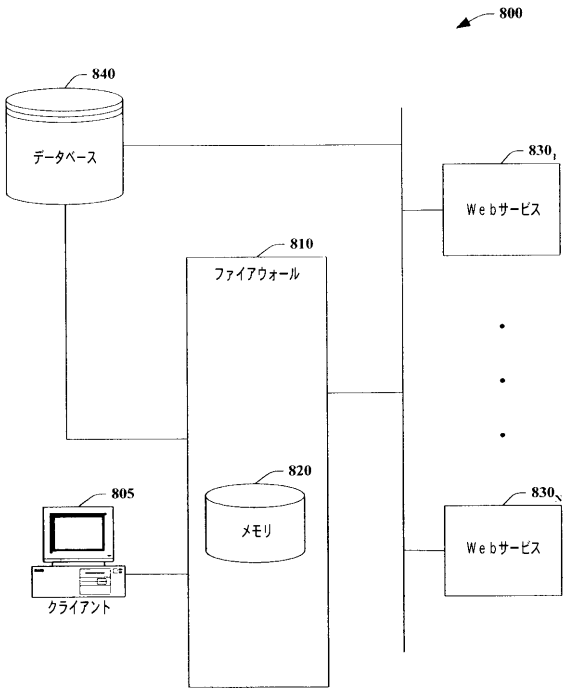
【図６】


600
405
Webサーバ
610
クエリ結果および／
または
無効化メッセージ
450
データベースサーバ

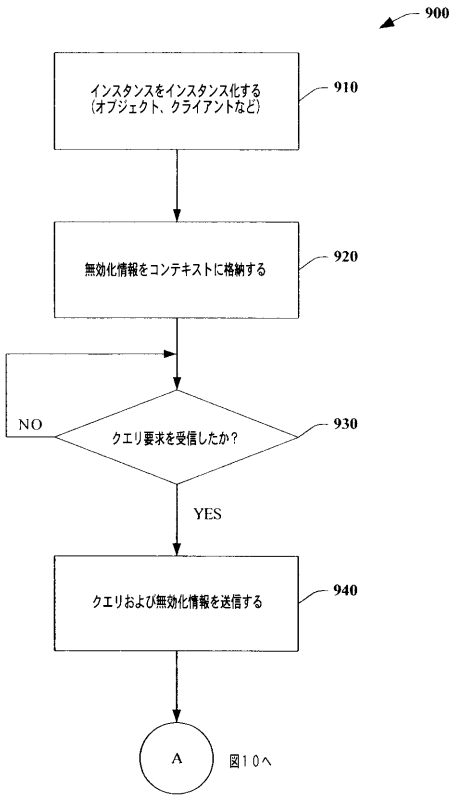
【図７】


700
740
無効化
710₁
Web
アプリケーション
720₁
コンテキスト
730₁
データベース
共有メモリ
710N
Web
アプリケーション
720M
コンテキスト
730K
データベース

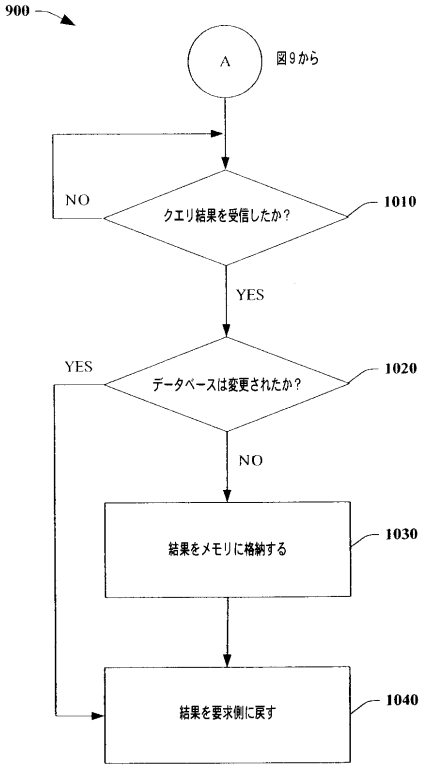
【図８】


800
840
データベース
830₁
Webサービス
810
ファイアウォール
820
メモリ
805
クライアント
830N
Webサービス

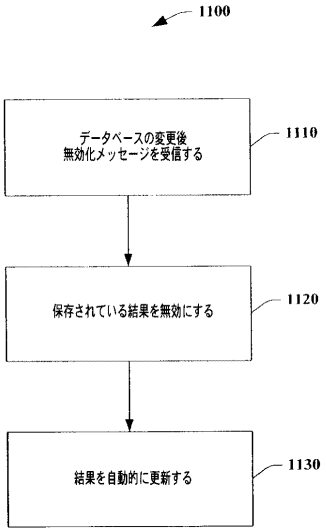
【図９】


900
インスタンスをインスタンス化する
(オブジェクト、クライアントなど)
910
無効化情報をコンテキストに格納する
920
NO
クエリ要求を受信したか?
930
YES
クエリおよび無効化情報を送信する
940
A
図10へ

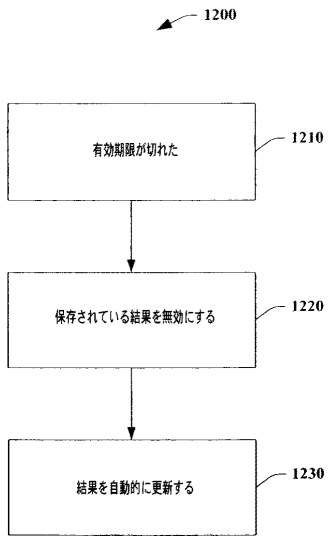
【図１０】


900
A
図9から
NO
クエリ結果を受信したか?
1010
YES
YES
データベースは変更されたか?
1020
NO
結果をメモリに格納する
1030
結果を要求側に戻す
1040

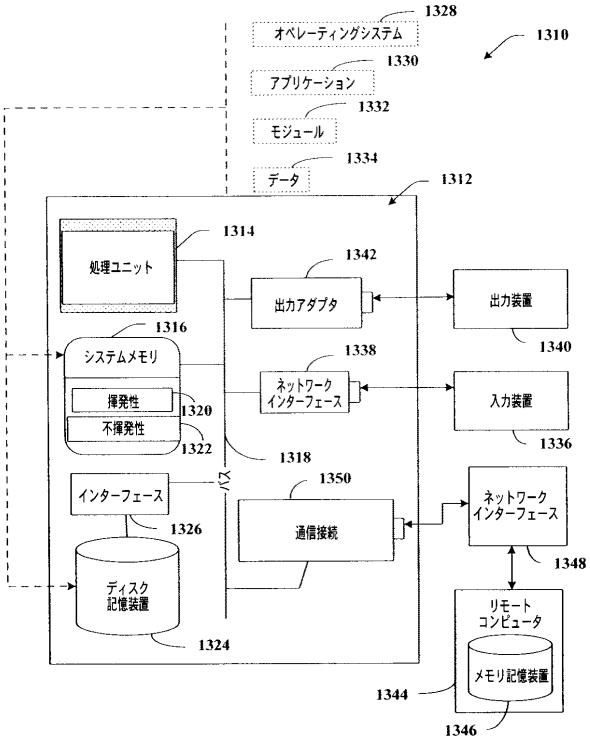
【図１１】


1100
データベースの変更後
無効化メッセージを受信する
1110
保存されている結果を無効にする
1120
結果を自動的に更新する
1130

【図１２】


1200
有効期限が切れた
1210
保存されている結果を無効にする
1220
結果を自動的に更新する
1230

【図１３】


1328
オペレーティングシステム
1310
1330
アプリケーション
1332
モジュール
1334
データ
1312
1314
処理ユニット
1342
出力アダプタ
出力装置
1340
1316
システムメモリ
1338
ネットワーク
インターフェース
入力装置
揮発性
1320
不揮発性
1322
1336
1318
バス
1350
インターフェース
通信接続
ネットワーク
インターフェース
1348
1326
ディスク
記憶装置
リモート
コンピュータ
メモリ記憶装置
1324
1344
1346

フロントページの続き

(72)発明者 ロバート エム．ハワード
アメリカ合衆国 ９８０６５ ワシントン州 スノコルミー サウスイースト カーティス ドライブ ３４９１０
(72)発明者 クリスチャン クレイナーマン
アメリカ合衆国 ９８００７ ワシントン州 ベルビュー ノースイースト ３４ ストリート １４６４５ ナンバーシー２
(72)発明者 パトリック ユークァン ング
アメリカ合衆国 ９８０５２ ワシントン州 レッドモンド ノースイースト ４２ コート １６８５３
(72)発明者 ジョン エフ．ノス
アメリカ合衆国 ９８０２７ ワシントン州 イサコア シエラ コート サウスウエスト ３２３５
(72)発明者 アダム ダブリュ．スミス
アメリカ合衆国 ９８０５３ ワシントン州 レッドモンド ノースイースト ８７ プレイス ２１８１１
(72)発明者 オレン トラトナー
アメリカ合衆国 ９８０３３ ワシントン州 カークランド ノースイースト １０６ レーン １１７３１
(72)発明者 フロリアン エム．ワース
アメリカ合衆国 ９８１０４ ワシントン州 シアトル ファースト アベニュー サウス １１１ ナンバー２０９

Ｆターム(参考) 5B082 FA12 GB02 HA02 HA08

【外国語明細書】

Title: SYSTEMS AND METHODS FOR CACHING AND INVALIDATING DATABASE RESULTS AND DERIVED OBJECTS

TECHNICAL FIELD

The present invention relates generally to databases, and more particularly to systems and methods for saving results generated from database query(s) for subsequent utilization, and invalidating the saved results when they become inconsistent with the database.

BACKGROUND OF THE INVENTION

Computing and networking technologies have transformed many important aspects of everyday life. Computers have become a household staple instead of a luxury, educational tool and/or entertainment center, and provide users with a tool to manage and forecast finances, control household operations like heating, cooling, lighting and security, and store records and images in a permanent and reliable medium. Networking technologies like the Internet provide users with virtually unlimited access to remote systems, information and associated applications.

As computing and networking technologies become robust, secure, and reliable, more consumers, wholesalers, retailers, entrepreneurs, educational institutions and the like are shifting paradigms and employing the Internet to perform business instead of the traditional means. For example, many businesses and consumers are providing web sites and/or on-line services. For example, today a consumer can access his/her account *via* the Internet and perform a growing number of available transactions such as balance inquiries, funds transfers and bill payment.

Typically, an Internet session includes a user interfacing with a client application (*e.g.*, a web server) to interact with a database server that stores information in a database that is accessible to the client application. For example, a stock market web site can provide the user with tools for retrieving stock quotes and purchasing stock. The user can type in a stock symbol and request a stock quote by performing a mouse click to activate a query. The client application then queries a database(s) with stock information and returns the stock quote. Likewise, the user can purchase the stock *via* providing suitable

information. wherein submitting the order can initiate a database query to return current price information and order status.

As more users take advantage of such services. more bandwidth is consumed which can reduce performance and/or speed since bandwidth is a limited resource. Another shortcoming is the limited available data transfer rates relative to the quantity of data available. For example. requests that retrieve large amounts of data (*e.g.*, distributed across various servers) can be time intensive and performance limiting. even when bandwidth is available.

The foregoing is compounded by users performing frequent and similar requests. For example. a user desiring current stock quotes may perform frequent requests to obtain updated quotes. Thus. the user frequently consumes bandwidth to retrieve substantially similar information. In addition. other users can concurrently perform requests that retrieve substantially similar information. The user(s) is further limited in that the consistency of the retrieved data cannot be guaranteed because the returned results can become invalid after a subsequent database table change.

SUMMARY OF THE INVENTION

The following presents a simplified summary of the invention in order to provide a basic understanding of some aspects of the invention. This summary is not an extensive overview of the invention. It is not intended to identify key/critical elements of the invention or to delineate the scope of the invention. Its sole purpose is to present some concepts of the invention in a simplified form as a prelude to the more detailed description that is presented later.

The present invention relates to systems and methods for invalidating saved results generated from database queries when the results become invalid due to a database change and/or due to an expiration period lapse. Thus, the present invention provides a mechanism to query a database, save results generated from the database query(s), and utilize the saved results until a database change and/or expiration period invalidate the results.

Generally, databases store information that is retrievable *via* querying the database. A user can interface with an application, for example, to obtain at least a

portion of the information stored within the database. For example, a user may employ a web service and submit a request that is associated with information stored on a server. The request will initiate one or more queries of a relevant database(s), and then the results of the one or more queries can be used to generate a response to be returned from the web service, displayed, consumed and/or further employed.

In many instances, a database can become a common source of resource contention. For example, more than one user can transmit a similar request that concurrently queries the database and returns similar results. Such concurrent querying for similar results can increase database load, reduce bandwidth, diminish performance and consume resources, for example. In another example, one or more users can frequently perform requests that return similar results. Likewise, system performance can be reduced *via* the database load, for example. Under the foregoing, the database represents a state that is shared amongst a plurality of user requests and/or users, and shared state contention can limit system performance and scalability.

A typically employed technique to reduce the number of queries comprises saving frequently utilized data (*e.g.*, in the form of raw data, transformed data, and data in a queryable form) externally from the database, and utilizing the saved results instead of performing additional queries. However, the consistency of the saved data is compromised because a subsequent database change can occur to render the saved data inconsistent with the database without apprising the user(s).

The systems and methods of the present invention reduce database load (*e.g.*, reduce query(s)) for substantially similar information), increase scalability, reduce repetitive processing, and ensure data consistency. The systems and methods include storing invalidation information in a context bank, wherein the invalidation information can be retrieved and transmitted with a query in order to associate the query with a query requester and register the query with a database. The results (*e.g.*, raw data, transformed data, data in a queryable form, a partial response and a full response) generated from the query can then be saved in an accessible storage medium external to the database, and employed one or more times, and by various users (*e.g.*, web servers), thus mitigating querying for substantially similar information.

If a database change occurs that would affect the stored results, an invalidation message can be sent to facilitate invalidating the saved results to mitigate a user from employing inconsistent data. The invalidation information transmitted with the query is typically utilized to construct the invalidation message. In addition, an expiration period can be defined such that the saved results will be invalidated after the expiration period lapses, even if a database change has not occurred. After invalidating the saved results, the results can be automatically discarded and/or automatically refreshed, for example.

Furthermore, the systems and methods of the present invention can be employed in a web environment, wherein further reduction in latency and usage of web server resources can be achieved when the web response generated from a database query(s) can be stored, in whole or in part, and returned in response to subsequent requests. For example, a developer can request, *via* embedding an annotation(s) within a web page, that a response generated based on a database query(s) be cached until the results of the query(s) change. The annotation(s) invokes the web server to associate a database dependency with the response to be cached and to store the invalidation information associated with the dependency, and/or the dependency itself, in a context bank. The stored invalidation information is automatically retrieved and transmitted with a subsequent query(s) to a database. As noted above, the invalidation information can be employed in an invalidation message when a change occurs to the database that renders previously queried results inconsistent and/or when an expiration period lapses.

Storing and utilizing the invalidation information in the context bank facilitates instantiating the infrastructure for associating the invalidation information with the query and saving (*e.g.*, caching) the query response. Thus, employing the stored invalidation information mitigates requiring the web server (and/or other component(s) capable of processing the query request) to have knowledge about the generation of the response. Further, employing the stored invalidation information mitigates the component generating the response from having to know how the response is going to be utilized by the web server.

According to the subject invention, the systems and methods described herein employ components to store invalidation information, to execute queries, to save results based on the queries, to receive invalidation messages and/or to invalidate saved

information. Additionally, various data transfer techniques for asynchronous and synchronous communication and security techniques that mitigate the existence and/or transfer of malicious and fictitious information and messages can be employed. In practice, the systems and methods can be employed in various environments, including client-server and/or SQL based environments.

To the accomplishment of the foregoing and related ends, the invention comprises the features hereinafter fully described and particularly pointed out in the claims. The following description and the annexed drawings set forth in detail certain illustrative aspects and implementations of the invention. These are indicative, however, of but a few of the various ways in which the principles of the invention may be employed. Other objects, advantages and novel features of the invention will become apparent from the following detailed description of the invention when considered in conjunction with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates an exemplary invalidation registration system in accordance with one aspect of the present invention.

Fig. 2 illustrates an exemplary invalidation memory management system in accordance with one aspect of the present invention.

Fig. 3 illustrates an exemplary database registration and invalidation system in accordance with one aspect of the present invention.

Fig. 4 illustrates an exemplary web based memory invalidation system in accordance with one aspect of the present invention.

Fig. 5 illustrates exemplary security techniques in accordance with one aspect of the present invention.

Fig. 6 illustrates exemplary invalidation message transfer techniques in accordance with one aspect of the present invention.

Fig. 7 illustrates an exemplary distributed invalidation system in accordance with one aspect of the present invention.

Fig. 8 illustrates an exemplary web servicing invalidation system in accordance with one aspect of the present invention.

Fig. 9 illustrates an exemplary automatic invalidation methodology in accordance with one aspect of the present invention.

Fig. 10 is a continuation of Fig. 9 in accordance with one aspect of the present invention.

Fig. 11 illustrates an exemplary change based invalidation methodology in accordance with one aspect of the present invention.

Fig. 12 illustrates an exemplary time based invalidation methodology in accordance with one aspect of the present invention.

Fig. 13 illustrates an exemplary operating system in accordance with one aspect of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It may be evident, however, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to facilitate describing the present invention.

As used in this application, the term "component" is intended to refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution. For example, a component may be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a server and the server can be a computer component. One or more components may reside within a process and/or thread of execution and a component may be localized on one computer and/or distributed between two or more computers. A "thread" is the entity within a process that the operating system kernel schedules for execution. Each thread has an associated "context" which is the volatile data associated with the execution of the thread. A thread's context includes the contents of system

registers and the virtual address belonging to the thread's process. Thus, the actual data comprising a thread's context varies as it executes.

The present invention relates to systems and methods to store results generated from database query(s) (*e.g.*, dynamic and static) in memory wherein the results can be retrieved and utilized, and invalidated when they become inconsistent (*e.g.*, database changes and expiration period lapses). The systems and methods mitigate system load and resource contention *via* reducing querying for substantially similar (*e.g.*, redundant) information. In addition, the systems and methods mitigate employing inconsistent saved results. Thus, the systems and methods provide a mechanism to query a database for information, store results in memory such as cache, employ the saved results one or more times when substantially similar results are desired, automatically invalidate the saved results when they become inconsistent, and optionally refresh the memory *via* automatically re-querying and saving.

Referring to Fig. 1, an exemplary invalidation registration system 100 in accordance with an aspect of the present invention is illustrated. The invalidation registration system 100 comprises a context bank 110 and a query manager 120.

The context bank 110 can be employed to provide information to facilitate invalidating saved results when a database change occurs that would affect the saved results and/or when an expiration period lapses. For example, the context bank 110 can be utilized to store invalidation information associated with a component(s) (*e.g.*, a dependency, as described below). The invalidation information can be linked to a database *via* a database query. The results can be saved and utilized one or more times. When a change occurs to the database that would affect the query and/or the time duration expires, the corresponding invalidation information can be utilized to facilitate invalidating the saved results, for example *via* a notification, an event, a flag, an automatic invalidation of the saved results, and an automatic re-query to obtain and refresh memory with consistent results.

Typical invalidation information includes an invalidation string, service information and a time out. The invalidation string generally comprises a component identifier, a location (*e.g.*, machine name), a port, an address and/or a queue name, and can further include authentication information, encryption and protocol preferences, for

example. The service information can include flags for automatic saved results removal. automatically refreshing results, automatically tagging results as invalid. and notifying the component without affecting the saved results, for example. The time out can be based on a database change, a file change, and time duration. for example. It is to be appreciated that the foregoing is illustrative and not limitative. Therefore. additional. different and/or less information can be included in accordance with an aspect of the present invention.

Invalidation information stored in the context bank 110 is available to the query manager 120. The query manager 120 can utilize the invalidation information when executing a command and/or query. For example, the query manager 120 can obtain invalidation information from the context bank 100, and include the invalidation information with a query, wherein the invalidation information can be part of the query and/or associated with the query.

In one aspect of the present invention. a page directive (*e.g.*, an annotation(s) within a web page) can be employed to initiate the instantiation of a component (*e.g.*, a dependency object) as well as the storage of invalidation information associated with the component in the context bank 110. Employing the context bank 110 to store invalidation information facilitates establishing the infrastructure to employ the invalidation information with a query without the application invoking the query manager 120 having to know how the response may be cached.

When the query manager 120 is invoked *via* a request to query a database(s), the query manager 120 interacts with the context bank 110 to retrieve the invalidation information associated with the component. Then the query manager 120 executes the request *via* querying the database(s), wherein the invalidation information retrieved from the context bank 110 is transmitted serially and/or concurrently with the query. The query results are then retuned and utilized to generate the result to be stored, and the transmitted invalidation information remains associated with the component, the query and the queried database.

If the queried database did not change subsequent the query and prior to generating the results to be stored (*e.g.*, if the results to be stored are consistent with the database), then the results can be utilized and saved in memory for further employment.

When a request that would return substantially similar results is subsequently submitted, the stored results can be utilized instead of performing another query. Thus, the present invention mitigates performing queries for substantially similar results that can reduce performance, throughput, processing and speed, and that are susceptible to resource contention.

After saving the results, a database change or lapse of time can render the saved results inconsistent. When a database change occurs that would affect the results of the aforementioned query and/or a time out expires, the invalidation information transmitted with the query can be employed to notify a component (*e.g.*, the memory manager 120) to invalidate (*e.g.*, delete and refresh) the saved results, even though the component that generated the result may no longer exist. Other components attempting to associate with the saved results can be provided with a notification indicating the saved results are invalid. Thus, the present invention mitigates employing inconsistent saved results *via* informing components and/or invalidating saved results.

Moving to Fig. 2, an exemplary invalidation memory management system 200 in accordance with an aspect of the present invention is illustrated. The invalidation memory management system 200 comprises the query manager 120 and a memory manager 220.

In addition to the functionality noted *supra*, the query manager 120 can receive query results and/or invalidation messages for databases that change and/or expire. Generally, after the invalidation information (*e.g.*, as described above) is transmitted with a query, the query results are returned. In one aspect of the present invention, the query manager 120 receives the returned query results, and the memory manager 220 is employed to facilitate saving and/or utilizing results generated from the query results. If it is discerned that the database changed and/or expiration period lapsed prior to generating the results to be stored, the memory manager 220 can provide the results to the component without saving the results to memory. If the data is consistent (*e.g.*, no database change) and the expiration period has not expired, the memory manager 220 can provide the results to the component and/or save to memory, serially and/or concurrently. Saved results can be accessed by the component(s) when a substantially similar query is

requested; thus mitigating performing substantially similar queries, which consume resources and reduce performance.

It is to be appreciated that the returned results can include raw data, transformed data (*e.g.*, denormalized, aggregations, summaries, pivot tables, objects generated from the raw data, and/or partial or complete web responses), and/or queryable data (*e.g.*, data sets that can be utilized *via* subsequent queries to obtain a subset thereof). Furthermore, it is to be appreciated that various techniques can be employed to save results to memory. For example, in a first approach, substantially all of the returned results (*e.g.*, a complete web response) can be saved. In a second approach, a portion of the returned results (*e.g.*, a partial page) is saved to memory. For example, the portion that may be common to a subsequent request(s) is saved. In another example, the results that are common to multiple requests are saved. Then, a local query against the common results retrieved from memory can return the data associated with a request. In a third approach, personalized results can be obtained through a query or some other means and common results can be saved to memory and utilized. It is to be appreciated that the exemplary illustrations depicted above are provided for explanatory purposes and do not limit the invention.

After receiving an invalidation message, the query manager 120 can facilitate routing a database change notification to the component(s) and/or invalidate saved results. In one aspect of the present invention, the query manager 120 notifies the component(s), and the component decides the next action. For example, the component can employ the memory manager 220 to invalidate the saved results. Alternatively, the query manager 120 employs the memory manger 220, wherein the memory manager 220 can invalidate the saved results and/or notify the component. It is to be appreciated that the component that initiated the query and the corresponding save may no longer exist when the saved results are invalidated.

Referring now to Fig. 3, an exemplary database registration and invalidation system 300 is illustrated in accordance with an aspect of the present invention. The database registration and invalidation system 300 comprises the context bank 110, the query manager 120, a memory manager 220, and optionally a client 310, and a memory 320. It is to be appreciated that more than one memory manager 220, client 310 and

memory 320 can be employed: however, for brevity a client instance is described in the following.

As noted *supra*, the context bank 110 can store invalidation information (*e.g.*, as described above) for components that can be employed to provide information for invalidating saved results when a database change occurs that would affect the saved results and/or when an expiration period lapses. In one aspect of the present invention, the client 310 can provide invalidation information - an identifier, a machine name, a port, an address, a database name, an indication to automatically invalidate inconsistent saved results, an indication to automatically refresh inconsistent results, and a time out (*e.g.*, invalidate on a database change, a file change and a duration of time) – to the context bank 110.

When the client 310 invokes the query manager 120 to query a database, the query manager 120 receives associated invalidation information from the context bank 110 (as depicted) and/or retrieves associated invalidation information from the context bank 110 (not shown). The query manager 120 can then perform the query and provide the invalidation information. It is to be appreciated that the query manager 120 can query a dynamic and/or static database in accordance with an aspect of the invention.

The query manager 120 can accept the query results, and transmit the results to the client 310. It is to be appreciated that the present invention affords synchronous and asynchronous message delivery. In synchronous message delivery, a database change can be blocked until delivery of the invalidation message and/or invalidation of the inconsistent results. Synchronous delivery mitigates data inconsistency *via* ensuring a database change does not occur between committing the change and receiving the invalidation message and/or invalidating the results. In asynchronous message delivery, a database change can be committed prior to delivering the invalidation message and/or prior to invalidating the inconstant results. Asynchronous delivery mitigates delivery latency.

After receiving the query results, the queried database(s) can then be probed, utilizing various techniques, to ascertain whether a change occurred after the query but prior to generating the results to be stored. If a change occurred, the client 310 can utilize the results without invoking the memory manager 220. If no change occurred, the client

310 can utilize the results and employ the memory manager 220 to store the results in the memory 320 (*e.g.*, local, remote, high-speed, HTTP and shared). Subsequent requests that would return substantially similar results can utilize the saved results from memory 320 instead of performing a query until the saved results become inconsistent.

In another aspect of the invention, various techniques are employed to facilitate determining whether to save results. If, for example, it is determined that the results are common to a plurality of requests and change infrequently, then the results are saved. However, if it is determined that the results change frequently and/or are not frequently utilized , and the cost of saving is relatively expensive, then the results are not saved. In yet another aspect of the present invention, various techniques are employed to mitigate memory issues. For example, if the memory 320 is full, then the results can be saved employing a First In First Out (FIFO) or First In Last Out (FILO) approach, an age based approach wherein the oldest information is discarded first, a usage approach wherein the least used, or longest since last used, information is discarded first, a size based approach and a key based approach. In another example, results are not saved until enough "free" memory is available. Other approaches, or variations of these approaches, may consider the amount of time and/or resources required to generate the result.

Once a change occurs to the database that would affect the results and/or the expiration period lapses, an invalidation message is dispatched to the system 300. Typically, the invalidation message comprises at least a portion of the invalidation information. The query manager 120 can receive invalidation messages. The query manager 120 then notifies the client 310 (and/or any other component registered to receive an invalidation message). The client 310 can then invoke the memory manager 220 to invalidate (*e.g.*, remove, erase, write over and move) saved results. If automatic refresh results were indicated in the invalidation information, a subsequent query is performed to generate new results that can be saved in memory 320 and utilized by components like client 310. Employing the foregoing technique can mitigate removing inconsistent results *via* saving the new results over the inconsistent results.

It is to be appreciated that the invalidation registration system 100, the invalidation memory management system 200, and the database registration and invalidation system 300 can further employ components to register databases (*e.g.*,

database tables, data tables and tables), to detect changes to databases, to determine if changed databases are registered, and/or to dispatch messages indicating that a registered database changed.

While Figs. 1-3 are block diagrams illustrating components for the systems 100-300, it is to be appreciated that the context bank 110, the query manager 120, the memory manager 220, the client 310, and the memory 320 can be implemented as one or more computer components, as that term is defined herein. Thus, it is to be appreciated that computer executable components operable to implement the system 100-300, the context bank 110, the query manager 120, the memory manager 220, the client 310, and the memory 320 can be stored on computer readable media including, but not limited to, an ASIC (application specific integrated circuit), CD (compact disc), DVD (digital video disk), ROM (read only memory), floppy disk, hard disk, EEPROM (electrically erasable programmable read only memory) and memory stick in accordance with the present invention.

Proceeding to Fig. 4, an exemplary web based memory invalidation system 400 in accordance with an aspect of the present invention is illustrated. The web based memory invalidation system 400 comprises a web server 405, dependency component 410, a context component 420, a query manager 430, a server 440, a database server 450, a memory store 460, a local memory 470, a high-speed memory 480, other memory 490 (*e.g.*, remote high speed memory), an application 492, and an ancillary layer 494.

The web sever 405 can access a web page(s) that includes a page directive (*e.g.*, an annotation(s), as described *supra*). The page directive can be utilized to specify that the web response will be saved, and that the saved response should be invalidated when an associated database changes and/or when an expiration period lapses. The web server 405 can utilize the page directive, invoke the dependency component 410 (*e.g.*, an object or derived object), and facilitate storage of invalidation information in the context component 420.

The dependency component 410 obtains a unique identifier, for example a globally unique identifier, or GUID, to identify the cached response to be associated with subsequent queries. The dependency component 410 provides the unique identifier and various other invalidation information (*e.g.*, as described *supra*) to the context component

420. Generally, invalidation information includes an invalidation string (*e.g.*, the GUID, a machine name, a port, an address and a queue name, and optionally authentication information, encryption and protocol preferences), service information and a time out (*e.g.*, on database change and on time expiration). However, the foregoing is illustrative and does not limit the invalidation information.

When the query manager 430 (*e.g.*, query manager 110) is invoked to query a database, the query manager 430 interfaces with the context component 420 and obtains associated invalidation information. The query engine 420 then transmits the query and the invalidation information to the server 440 (*e.g.*, dynamic and static) within the database server 450. The query is performed on the database 440 with the results returned to the application 492. The invalidation information remains with the database server 450, and provides the association between the dependency component 410 on the web server 405, and the results returned by the server 440.

After receiving the query results, the web server generates the whole or partial response. The web server 405 can poll the dependency component 410 to determine whether a database change occurred while the response was being generated, and then decide whether to save the results. If it is determined to save the results, then the web server 405 can employ the memory store 460 to facilitate saving the results to the local memory 470 (*e.g.*, cache), the high-speed memory 480 (*e.g.*, cache) and/or other memory 490. Typically, the local memory 470 can be associated with the application 490 and the high-speed memory 480 can be associated with the ancillary layer 494 (*e.g.*, HTTP.sys). Otherwise, the response can be utilized and/or discarded without saving.

When a database change occurs that will affect the saved web response, and/or when an expiration period lapses, the server 440 transmits an invalidation message to the dependency component 410. The dependency component 410 then signals the change (*e.g.*, *via* raising an event and setting a flag). The memory store 460 receives this signal and invalidates the response. Optionally, when an invalidation message is received, an automatic refresh can be employed to perform a query to obtain and save consistent results. The automatic refresh can write over inconsistent results and/or remove inconsistent results prior to saving the consistent results.

Next at Fig. 5. exemplary security techniques in accordance with an aspect of the present invention are illustrated. A system 500 comprises the web server 405, the database server 450 and a security layer 510 that operatively couples the web server 496 and the database server 450.

As noted above. the query engine 430 interfaces with the context component 420 to obtain invalidation information after receiving a query request. Then. the query engine 420 sends the query and the invalidation information from the web server 405 to the server 440 within the database server 450.

The invalidation information. as described previously. can include a unique identifier such as a globally unique identifier. or GUID. The GUID affords a security mechanism that can be employed to mitigate malicious attempts to simulate database changes and time period expirations. For example. in one aspect of the present invention the GUID is created at runtime when a database query registration request is received, thereby mitigating the opportunity to "guess." or mimic the GUID. The GUID is then used to identify the set of saved results to be invalidated.

If it is determined that the GUID is invalid (*e.g.*. it does not identify a set of saved results to be invalidated). the invalidation request can be ignored and/or discarded along with the GUID. for example. In another example. the fictitious invalidation request can be forwarded to the system administrator and utilized to trace the origination location.

Fig. 6 illustrates exemplary invalidation message transfer techniques in accordance with an aspect of the present invention. The system 600 comprises the web server 405. the database server 450 and a communication interface 610 operatively coupling the web server 405 and the database server 450.

After a query request is received, the query engine 430 obtains invalidation information from the context component 420 and sends the query and the invalidation information to the database server 450. The query is executed with the results transmitted to the web server 405, wherein the results can be saved in memory and utilized more than once to mitigate performing substantially similar queries.

After the results are returned from the database server 450, the invalidation information remains with the database server 450. and can be employed when a database change occurs to locate associated saved results in the web server 405. It is to be

appreciated that various mechanisms can be employed to store the invalidation information in the database server 450, and that various techniques can be utilized to match the invalidation information with a changed database and dispatch the invalidation message.

When a database change occurs, an invalidation message is constructed and transmitted over communication interface 610, from the database server 450 to the web server 496. The invalidation message can include at least a portion of the invalidation information, for example.

It is to be appreciated that the database change can be committed and/or held back. For example, in one aspect of the present invention, an asynchronous invalidation message transfer model can be employed, wherein a database change is committed regardless of whether an invalidation message is transmitted over communication interface 610 and/or whether saved results have been invalidated.

In another aspect of the invention, a synchronous invalidation message transfer model can be employed, wherein a database change is blocked until an invalidation message is transmitted over communication interface 610 and/or until saved results have been invalidated. An asynchronous model affords scalability, whereas a synchronous model affords reliability. The foregoing is provided to facilitate understanding and does not limit the invention, for example other techniques such as various other push and/or pull approaches can also be employed in accordance with the invention.

Fig. 7 illustrates an exemplary distributed invalidation system 700 in accordance with an aspect of the present invention. The invalidation system 700 comprises a first web application 710_1 through an Nth web application 710_N, N being an integer greater than or equal to one, a first context component 720_1 through an Mth context component 710_M, M being an integer greater than or equal to one, a first database 730_1 through an Kth database 730_K, K being an integer greater than or equal to one, and a shared memory 740.

The first web application 710_1 through the Nth web application 710_N can be referred to collectively as the web applications 710, the first context component 720_1 through the Mth context component 720_M can be referred to collectively as the context

components 720, and the first database 730_1 through the Kth database 730_K can be referred to collectively as the databases 730.

The web applications 710 (*e.g.*, object instances) can be employed to query databases to fulfill a user request(s). Typically, upon instantiation of the web applications 710 invalidation information associated with the web applications 710 is assembled and stored in respective context components 720 (*e.g.*, context bank 110). Invalidation information generally includes a unique identifier, an invalidation string, service information and an expiration period, as described *supra*; however, the invalidation information is not so limited.

When a query request is received by at least one of the web applications 710, the corresponding context bank is employed to obtain the associated invalidation information. Then, the query and the invalidation information are transmitted *via* a communication protocol (*e.g.*, Ethernet and Firewire) to the databases 730. It is noted that a query can employ one or more of the databases 730 to fulfill a user request.

After the query is performed, the results can be returned to the at least one web applications 710. To ensure that the results are consistent, the databases 730 can be checked to determine if a change occurred that would affect the results subsequent the query but prior to generating the result to be stored. If it is determined that a change occurred, the results can be utilized and/or discarded. If it is determined that the results are consistent, the results can be utilized and/or saved in memory, for example shared memory 740. Saved results are available to be employed one or more times. In addition, one more of the web applications 710 can employ the saved results serially and/or concurrently. Thus, the present invention further mitigates more than one of the web applications 710 performing queries for substantially similar results.

When a database change occurs to at least one of the databases 720 and/or when an expiration period lapses, at least one of the databases 720 transmits an invalidation message to one or more of the web applications 710 and/or the shared memory 740. It is to be appreciated that an (*e.g.*, one) invalidation message can comprise more than one web application address and/or other information (*e.g.*, other invalidation information) in order to inform more than one web application that is affected by the change.

In another aspect of the invention, more than one invalidation message can be transmitted, wherein an invalidation message is associated with and transmitted to a web application and/or the shared memory 740. In yet another aspect, the invalidation message can be broadcast such that substantially all of the web applications 710 and/or the shared memory 740 can receive the invalidation message. The web applications 710 can then decide whether to accept and/or act on the invalidation message. It is to be appreciated that providing the invalidation message to the shared memory 740 mitigates having to provide the invalidation message to a web application. For example, the web application(s) that initiated the request does not have to exist to invalidate the saved results. In addition, any application(s) that shared that saved results do not have to exist in order to invalidate the saved results. Thus, the present invention provides a mechanism to discard inconsistent results and/or free shared memory even when the initiating application and/or any sharing application no longer exists.

After receiving the invalidation message, the saved results can then be invalidated. For example, the saved results can be removed from memory or tagged as inconsistent. Optionally, an automatic refresh can be employed to perform a subsequent query to obtain consistent results to save to the memory. The automatic refresh can further re-send the invalidation information with the subsequent query in order to automatically invalidate the refreshed results they become inconsistent.

Fig. 8 illustrates an exemplary web servicing invalidation system 800 in accordance with an aspect of the present invention. The web servicing invalidation system 800 comprises a client 805, a firewall 810, a memory 820, a first web service 830_1 through an Nth web service 830_N, N being an integer greater than or equal to one, and a database 840. The first web service 830_1 through the Nth web service 830_N can be referred to collectively as the web services 830.

The client 805 interfaces with the web service 830 through the firewall 810. Access to the web services 830 can be controlled by the firewall 810. For example, the firewall 810 can limit access and/or deny access to the web services 830 by the client 805. Additionally, the firewall 810 can be employed to limit access by the client 805 to the database 840 and the memory 820. It is to be appreciated that client access to the web servers 830, the database 840 and/or the memory 820 can be mutually exclusive. For

example, the client 805 can have full access to the web servers 830 without access to the database 840 and/or the memory 820. However, the client 805 could have full or no access to the web servers 830, the database 840 and/or the memory 820.

The firewall 810 can further be employed to limit web service access to the database 840 and/or the results saved in memory 820. In one aspect of the present invention, one or more of the web services 830 can transmit a query (*e.g.*, initiated by the client 805), invalidation information (*e.g.*, as described above), and other information such as routing information to the database 840. If the one or more web services 830 have clearance, the transmitted information can be conveyed to the database 840. In another aspect of the present invention, the one or more of the web services 830 can transmit a query, invalidation information, and other information to the database 840 without conveying the information through the firewall 810.

After the query is performed, the database 840 returns the query results, which are utilized to generate a result to be stored. In one aspect of the invention, the result is returned to the firewall 810, wherein the firewall 810 employs a mechanism (*e.g.*, memory manager 120) to save the result in memory 820. In another aspect of the present invention, the result can be returned to the web services 830 (*e.g.*, *via* through and/or around the firewall 810). The web services 830 can then store the result in the memory 820, where the result can be available to web services 830 with access to the memory 820, and provide the result to the client 805.

The saved results can then be accessible to the web services 830 that have clearance to employ the memory 820. For example, a web service 830_N can send a query that would return results that are substantially similar to and/or a subset of the results saved in memory 820. If the web service 830_N has access to the memory 820, the web service 830_N can utilize the saved results instead of performing a query. If the web service 830_N does not have access to the memory 820, then a query is performed.

When a database change occurs that will affect the results saved in memory 840 and/or when an expiration period lapses, the database 840 can transmit an invalidation message to the firewall 810. If the invalidation message has clearance, the message can be utilized to invalidate inconsistent results in the memory 820. In one aspect of the invention, a notification can additionally be sent to the web services 830 to notify the web

services 830 that the results have been invalidated. In another aspect, instead of transmitting a notification to the web services 830, the web services 830 can periodically poll the memory 820 to determine if the results are still consistent, for example. In another example, when the web services 830 transmit a query and invalidation information, the memory 820 is searched to determine whether similar results are saved in the memory 820 and whether the similar results are consistent.

An automatic refresh can be employed to automatically re-query for consistent results and re-send invalidation information to invalidate when the next database change occurs and/or when a time period corresponding to the database data (results) expires.

Figs. 9-12 illustrate methodologies in accordance with the present invention. For simplicity of explanation, the methodologies are depicted and described as a series of acts. It is to be understood and appreciated that the present invention is not limited by the acts illustrated and/or by the order of acts, for example acts can occur in various orders and/or concurrently, and with other acts not presented and described herein. Furthermore, not all illustrated acts may be required to implement a methodology in accordance with the present invention. In addition, those skilled in the art will understand and appreciate that a methodology could alternatively be represented as a series of interrelated states (*e.g.*, state diagram) or events.

Proceeding to Fig. 9, an exemplary automatic invalidation methodology 900 is illustrated. At reference numeral 910, an instance is instantiated (*e.g.*, an object, a derived object, a client, an application, a client application and a dependency instance). At 920, invalidation information associated with the instance is stored in a context. Invalidation information typically comprises an invalidation string, service information and a time out, as described *supra*. It is to be appreciated that more than one instance can be instantiated, and more than one instance can store invalidation information in a similar context.

At reference numeral 930, the methodology waits to receive a request to perform a query. Once a request is received, the invalidation information stored in the context and associated with the instance is retrieved. Then at 940, the query and the invalidation information are transmitted, for example to the dynamic and/or static server where the database(s) resides.

Referring now to Fig. 10, a continuation of Fig. 9 in accordance with an aspect of the present invention is illustrated. Proceeding to reference numeral 1010, the methodology 900 waits for the results of the query. After the results are received and processed, the database(s) utilized in the query is examined at 1020 to ascertain whether a change occurred between the time of the query and the generation of the cacheable result. This ensures that the result is consistent.

If a change did not occur between the time of the query and the reception of the results, then the results are stored in memory at 1030 and provided to the instance at 1040. The saved results are available for further employment, for example to mitigate performing queries that would return similar results. If at 1020 it is determined that a change did occur between the query and the generation of the results to be stored, then at 1040 the results are typically provided to the instance and not saved. However, the instance can decide to store and utilize inconsistent results. Various techniques can be employed to notify other instances that the results are inconsistent. For example, a flag could be set to raise an event whenever an instance attempts to employ inconsistent results.

Next at Fig. 11, an exemplary data change based invalidation methodology 1100 in accordance with an aspect of the invention is illustrated. Proceeding at 1110, a database change occurs, and an invalidation message is transmitted and received. The invalidation message typically includes a portion of the invalidation information. For example, the unique identifier can be included to facilitate identifying the set of inconsistent results to be invalidated.

It is to be appreciated that the database change and the transmission/reception can be synchronized, wherein the database change is committed when the invalidation message is received and/or acted on. In another example, an asynchronous technique is employed wherein the database change is committed regardless of whether the transmission occurred and/or message was received.

At 1120, the saved results are invalidated. It is to be appreciated that various methods to invalidate the results can be employed. For example, the results can be removed from memory, tagged as invalid and/or overwritten. At 1130, an automatic

refresh can be performed to query for current and consistent results, and re-register for an invalidation message for a subsequent database change.

Referring now to Fig. 12, another exemplary time based invalidation methodology in accordance with an aspect of the invention is illustrated. Proceeding at 1210, the time period included in the invalidation information expires. Subsequently, an invalidation message is transmitted and received. The invalidation message can include a portion of the invalidation information. For example, the unique identifier can be included to facilitate identifying the set of inconsistent results to be invalidated. At 1220, the saved results are invalidated (*e.g.*, similar to above). At 1210, an automatic refresh can be performed to query for current and consistent results, and re-register for an invalidation message for a subsequent expiration.

In order to provide additional context for various aspects of the present invention, Fig. 13 and the following discussion are intended to provide a brief, general description of a suitable operating environment 1310 in which various aspects of the present invention may be implemented. While the invention is described in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices, those skilled in the art will recognize that the invention can also be implemented in combination with other program modules and/or as a combination of hardware and software. Generally, however, program modules include routines, programs, objects, components, data structures, *etc.*, that perform particular tasks or implement particular data types. The operating environment 1310 is only one example of a suitable operating environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Other well known computer systems, environments, and/or configurations that may be suitable for use with the invention include but are not limited to, personal computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include the above systems or devices, and the like.

With reference to Fig. 13, an exemplary environment 1310 for implementing various aspects of the invention includes a computer 1312. The computer 1312 includes a processing unit 1314, a system memory 1316, and a system bus 1318. The system bus

1318 couples system components including, but not limited to, the system memory 1316 to the processing unit 1314. The processing unit 1314 can be any of various available processors. Dual microprocessors and other multiprocessor architectures also can be employed as the processing unit 1314.

The system bus 1318 can be any of several types of bus structure(s) including the memory bus or memory controller, a peripheral bus or external bus, and/or a local bus using any variety of available bus architectures including, but not limited to, an 8-bit bus, Industrial Standard Architecture (ISA), Micro-Channel Architecture (MSA), Extended ISA (EISA), Intelligent Drive Electronics (IDE), VESA Local Bus (VLB), Peripheral Component Interconnect (PCI), Universal Serial Bus (USB), Advanced Graphics Port (AGP), Personal Computer Memory Card International Association bus (PCMCIA), and Small Computer Systems Interface (SCSI).

The system memory 1316 includes volatile memory 1320 and nonvolatile memory 1322. The basic input/output system (BIOS), containing the basic routines to transfer information between elements within the computer 1312, such as during start-up, is stored in nonvolatile memory 1322. By way of illustration, and not limitation, nonvolatile memory 1322 can include read only memory (ROM), programmable ROM (PROM), electrically programmable ROM (EPROM), electrically erasable ROM (EEPROM), or flash memory. Volatile memory 1320 includes random access memory (RAM), which acts as external cache memory. By way of illustration and not limitation, RAM is available in many forms such as synchronous RAM (SRAM), dynamic RAM (DRAM), synchronous DRAM (SDRAM), double data rate SDRAM (DDR SDRAM), enhanced SDRAM (ESDRAM), Synchlink DRAM (SLDRAM), and direct Rambus RAM (DRRAM).

Computer 1312 also includes removable/nonremovable, volatile/nonvolatile computer storage media. Fig. 13 illustrates, for example a disk storage 1324. Disk storage 1324 includes, but is not limited to, devices like a magnetic disk drive, floppy disk drive, tape drive, Jaz drive, Zip drive, LS-100 drive, flash memory card, or memory stick. In addition, disk storage 1324 can include storage media separately or in combination with other storage media including, but not limited to, an optical disk drive such as a compact disk ROM device (CD-ROM), CD recordable drive (CD-R Drive), CD

rewritable drive (CD-RW Drive) or a digital versatile disk ROM drive (DVD-ROM). To facilitate connection of the disk storage devices 1324 to the system bus 1318, a removable or non-removable interface is typically used such as interface 1326.

It is to be appreciated that Fig 13 describes software that acts as an intermediary between users and the basic computer resources described in suitable operating environment 1310. Such software includes an operating system 1328. Operating system 1328, which can be stored on disk storage 1324, acts to control and allocate resources of the computer system 1312. System applications 1330 take advantage of the management of resources by operating system 1328 through program modules 1332 and program data 1334 stored either in system memory 1316 or on disk storage 1324. It is to be appreciated that the present invention can be implemented with various operating systems or combinations of operating systems.

A user enters commands or information into the computer 1312 through input device(s) 1336. Input devices 1336 include, but are not limited to, a pointing device such as a mouse, trackball, stylus, touch pad, keyboard, microphone, joystick, game pad, satellite dish, scanner, TV tuner card, digital camera, digital video camera, web camera, and the like. These and other input devices connect to the processing unit 1314 through the system bus 1318 *via* interface port(s) 1338. Interface port(s) 1338 include, for example, a serial port, a parallel port, a game port, and a universal serial bus (USB). Output device(s) 1340 use some of the same type of ports as input device(s) 1336. Thus, for example, a USB port may be used to provide input to computer 1312 and to output information from computer 1312 to an output device 1340. Output adapter 1342 is provided to illustrate that there are some output devices 1340 like monitors, speakers, and printers among other output devices 1340 that require special adapters. The output adapters 1342 include, by way of illustration and not limitation, video and sound cards that provide a means of connection between the output device 1340 and the system bus 1318. It should be noted that other devices and/or systems of devices provide both input and output capabilities such as remote computer(s) 1344.

Computer 1312 can operate in a networked environment using logical connections to one or more remote computers, such as remote computer(s) 1344. The remote computer(s) 1344 can be a personal computer, a server, a router, a network PC, a

workstation, a microprocessor based appliance, a peer device or other common network node and the like, and typically includes many or all of the elements described relative to computer 1312. For purposes of brevity, only a memory storage device 1346 is illustrated with remote computer(s) 1344. Remote computer(s) 1344 is logically connected to computer 1312 through a network interface 1348 and then physically connected *via* communication connection 1350. Network interface 1348 encompasses communication networks such as local-area networks (LAN) and wide-area networks (WAN). LAN technologies include Fiber Distributed Data Interface (FDDI), Copper Distributed Data Interface (CDDI), Ethernet/IEEE 802.3, Token Ring/IEEE 802.5 and the like. WAN technologies include, but are not limited to, point-to-point links, circuit switching networks like Integrated Services Digital Networks (ISDN) and variations thereon, packet switching networks, and Digital Subscriber Lines (DSL).

Communication connection(s) 1350 refers to the hardware/software employed to connect the network interface 1348 to the bus 1318. While communication connection 1350 is shown for illustrative clarity inside computer 1312, it can also be external to computer 1312. The hardware/software necessary for connection to the network interface 1348 includes, for exemplary purposes only, internal and external technologies such as, modems including regular telephone grade modems, cable modems and DSL modems, ISDN adapters, and Ethernet cards.

What has been described above includes examples of the present invention. It is, of course, not possible to describe every conceivable combination of components or methodologies for purposes of describing the present invention, but one of ordinary skill in the art may recognize that many further combinations and permutations of the present invention are possible. Accordingly, the present invention is intended to embrace all such alterations, modifications and variations that fall within the spirit and scope of the appended claims. Furthermore, to the extent that the term "includes" is used in either the detailed description or the claims, such term is intended to be inclusive in a manner similar to the term "comprising" as "comprising" is interpreted when employed as a transitional word in a claim.

1. A memory invalidation registration system comprising:

a context bank to store invalidation information;

a query manager to submit database queries and associated invalidation information from the context bank to a database; and

a memory to store results based on database queries, wherein the associated invalidation information is maintained by the database.

2. The system of claim 1, the invalidation information comprising at least one of an invalidation string, service information, and an expiration period.

3. The system of claim 2, the invalidation string comprising at least one of a unique identifier, a machine name, a port, an address, and a queue name.

4. The method of claim 3, the invalidation string further comprising at least one of authentication, encryption and protocol information.

5. The system of claim 1, the query manager operative with the context bank in order to obtain invalidation information to submit with the queries.

6. The system of claim 1, the invalidation information associated with a stored result associated with a plurality of queries.

7. The system of claim 1, the query manager receiving invalidation messages after at least one of a database change that would affect the stored results and a time expiration as defined in the invalidation information.

8. The system of claim 7, the database change being blocked until the change is synchronously delivered.

9. The system of claim 7, the database change completing while the change is asynchronously delivered.

10. The system of claim 1, automatically invalidating saved results in the memory after the invalidation message is received.

11. The system of claim 1, automatically re-querying the database to update saved results in memory after the invalidation message is received.

12. The system of claim 3, the unique identifier generated at runtime to define the association between the saved results and the source queries.

13. The system of claim 12, the unique identifier employed to group one or more queries with a saved result.

14. The system of claim 12, the unique identifier employed as a security mechanism to mitigate at least one of fictitious invalidation information and fictitious invalidation messages.

15. The system of claim 1, further comprising a memory manager to coordinate result storage and invalidation.

16. The system of claim 10, the automatic invalidation comprising at least one of removing the inconsistent results from memory, writing over the inconsistent results in memory and tagging the inconsistent results in memory.

17. The system of claim 1, the memory comprising cache memory.

18. The system of claim 1, the memory shared by a plurality of components.

19. The system of claim 1, employed with an SQL based environment.

20. The system of claim 1, further comprising at least one of an annotation and a page directive to request that the results be stored in memory for utilization by subsequent requests and invalidated when the stored results becomes inconsistent with the database.

21. The system of claim 1, the context bank facilitating conveyance of the invalidation information to queries employed to generate the stored results.

22. The system of claim 1, the stored results comprising at least one of raw data, transformed data, denormalized data, aggregated data, summary data, pivot table data, objects generated from the raw data, partial web responses, complete web responses, and queryable data.

23. An automatic memory invalidation system comprising:
a context that stores invalidation information for one or more components;
a query manager to submit database queries and invalidation information; and
one or more memories to store results generated from the database queries.

24. The system of claim 23, the one or more memories comprising local cache memory, ancillary cache memory, and remote high speed memory.

25. The system of claim 23, further comprising a security layer operative to the query engine and the dependency component to mitigate at least one of malicious queries with fictitious invalidation information and fictitious invalidation messages.

26. The system of claim 23, further comprising a mechanism to provide at least one of automatically invalidating saved results when the saved results become inconsistent and notifying components that saved results have become inconsistent.

27. The system of claim 23, further comprising a firewall as a security layer.

28. The system of claim 27, the one or more memories residing in the firewall.

29. The system of claim 27, the firewall facilitating at least one of result storage, result access and result invalidation in the one or more memories.

30. The system of claim 27, the one or more memories concurrently accessible to the one or more components.

31. A method to save results registered to automatically invalidate when the results become inconsistent, comprising:

storing invalidation information in a context;

sending the invalidation information with an associated query; and

saving the results generated from the query in memory.

32. The method of claim 31, further comprising checking the queried database for changes subsequent the query but prior to storing the result in order to ensure consistent results.

33. The method of claim 31, the invalidation information comprising at least one of an invalidation string, service information, and an expiration period.

34. The method of claim 33, the invalidation string comprising at least one of an identifier, a machine name, a port, an address, and a queue name.

35. The method of claim 34, the invalidation string further comprising at least one of authentication, encryption and protocol information.

36. The method of claim 31, the memory being cache memory.

37. A method to automatically invalidate saved results generated from a database query when the results become inconsistent, comprising:

receiving an invalidation message including at least a portion of invalidation information, and

utilizing the at least a portion of invalidation information to invalidate saved results.

38. The method of claim 37, further comprising automatically invalidating the results.

39. The method of claim 37, further comprising automatically re-querying to generate and save consistent results.

40. The method of claim 37, the invalidation message associated with at least one of a database change that would affect the stored results and an expiration period lapse.

41. A data packet transmitted between two or more computer components that facilitate storing results in memory and automatically invalidating the stored results when the stored results become inconsistent comprising:

invalidation information stored in a context that can be retrieved and sent with a query and utilized in an invalidation message to automatically invalidate saved results.

42. A computer readable medium storing computer executable components of a memory invalidation and registration system for saving results and invalidating the results when the results become inconsistent, comprising:

a context to store invalidation information associated with components;

a query manager to obtain invalidation information from the context and send the invalidation information with a requested query; and

a memory manager to facilitate saving results and invalidating inconsistent results.

43. A system for invaliding results saved in memory. comprising:

a means to obtain invalidation information of a component requesting to query a database;

a means to send the invalidation information with a query;

a means to receive an invalidation message: and

a means to invalidate inconsistent results stored in memory.

44. The system of 43. further comprising a means to automatically re-query the database for consistent results.

45. The system of 43. further comprising a means to verify invalidation information and invalidation messages are not fictitious.

1. Abstract

The present invention relates to systems and methods for saving results generated from database queries such as raw data, objects, queryable data sets, full or partial web responses, in memory for current and subsequent utilization and invaliding the saved results when they become inconsistent to mitigate employing inconsistent results. The saved results can be employed by one or more components and are typically utilized when a subsequent query would return substantially similar results. Thus, the system and methods mitigate performing substantially redundant queries that can reduce performance and consume resources. When a database change occurs that affect the consistency of the saved results and/or expiration period lapses, an invalidation message can be transmitted, wherein the message is employed to invalidate the saved results. Thus, the systems and methods mitigate employing inconsistent saved results. In addition, automatic re-querying techniques can be employed to automatically refresh inconsistent results to obtain consistent results.

2. Representative Drawing

FIG. 3

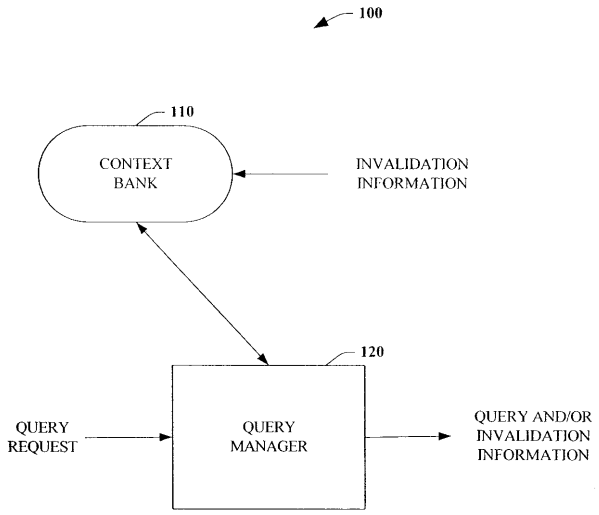


FIG. 1

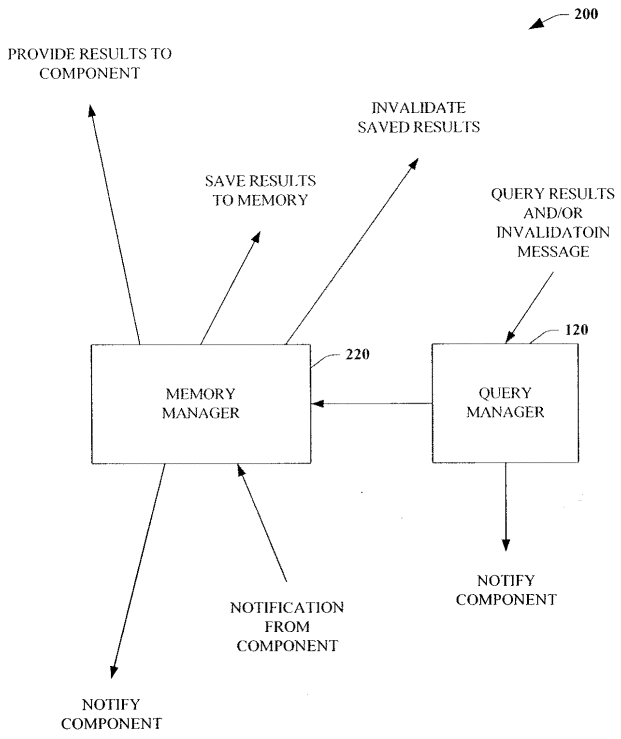


FIG. 2

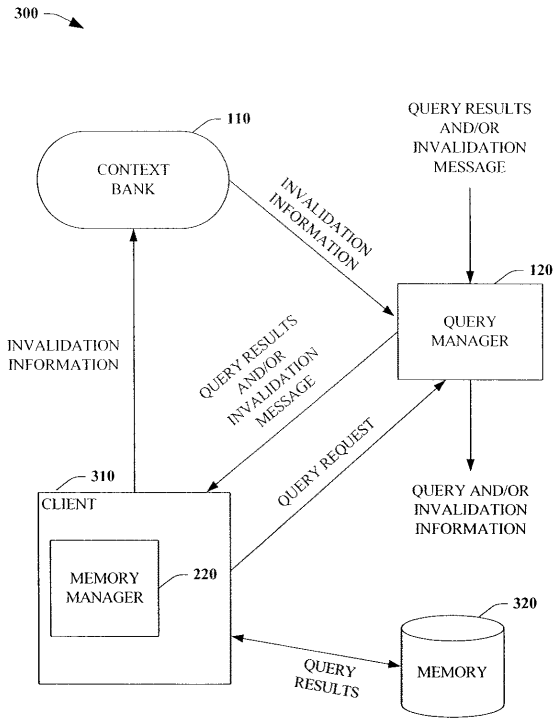


FIG. 3

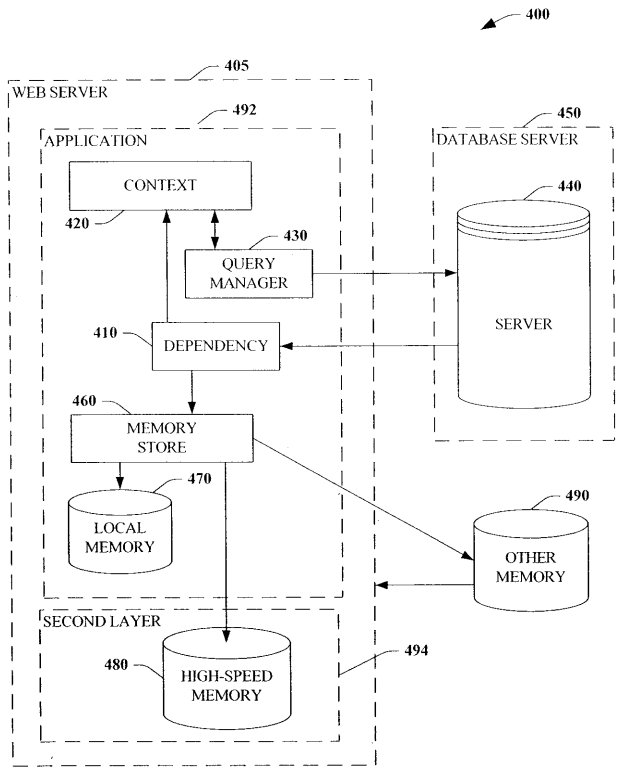


FIG. 4

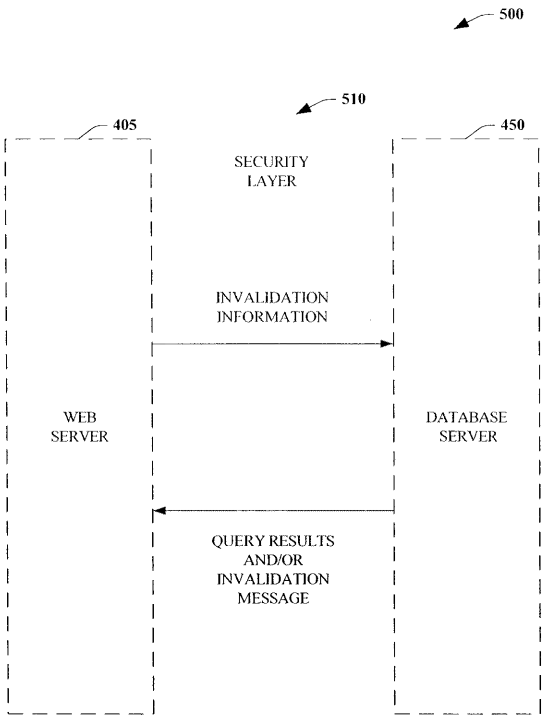


FIG. 5

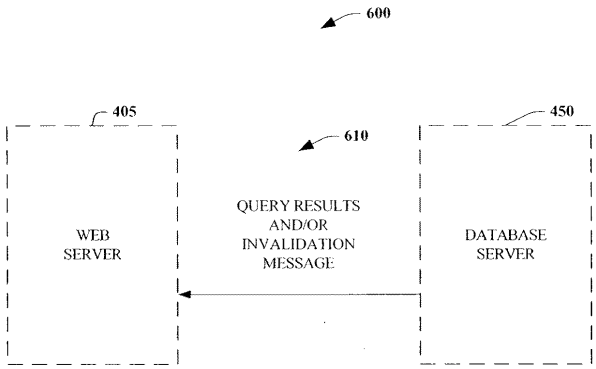


FIG. 6

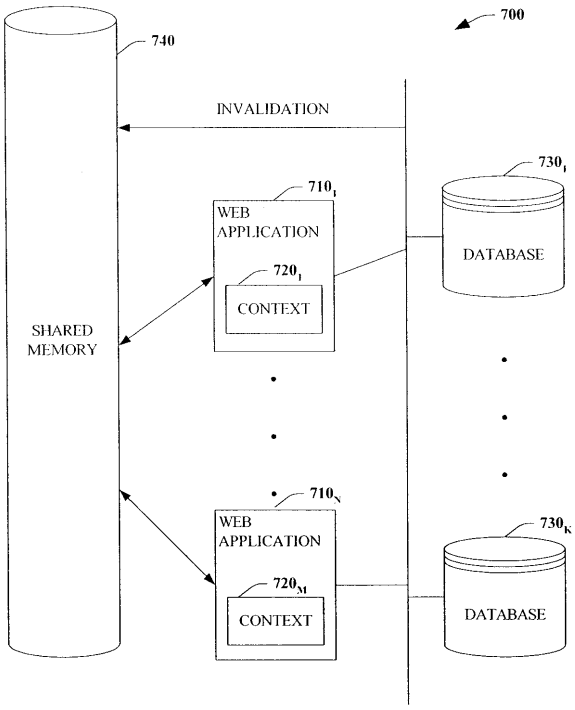


FIG. 7

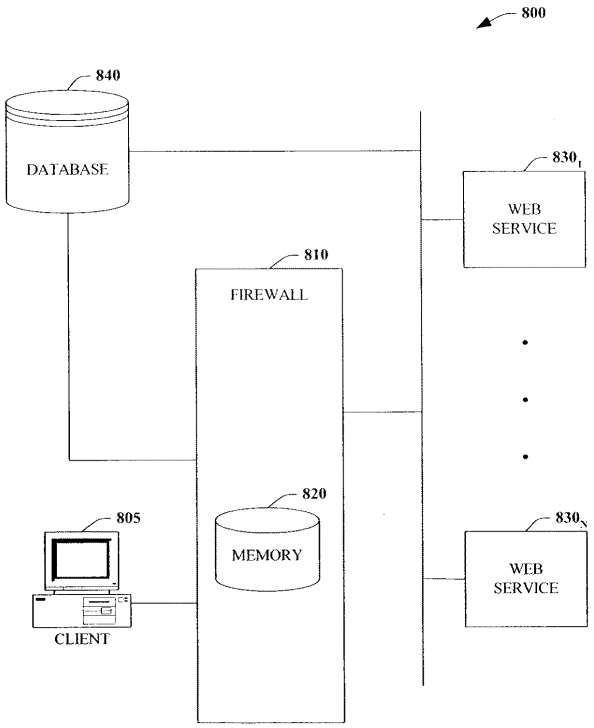


FIG. 8

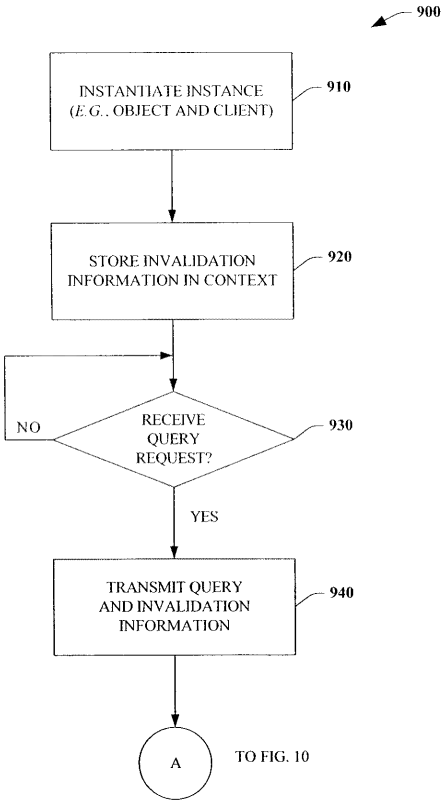


FIG. 9

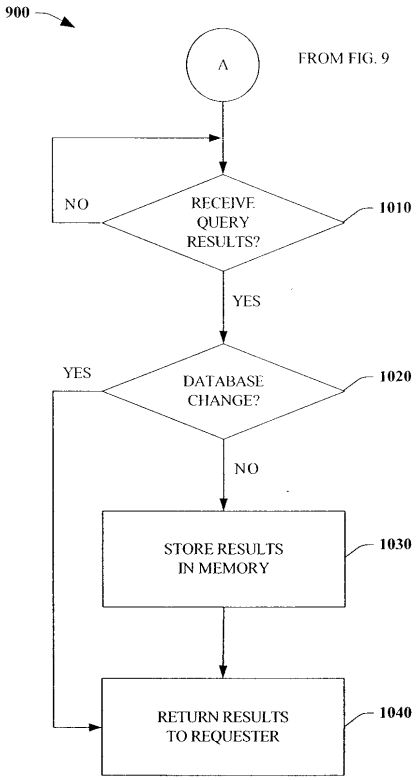


FIG. 10

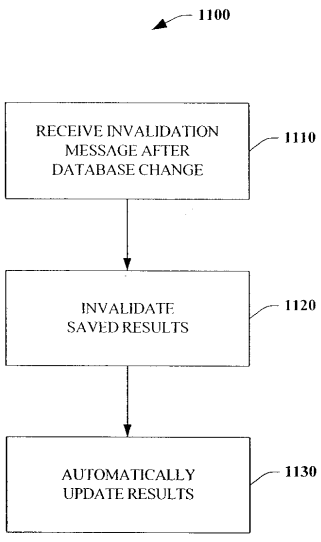


FIG. 11

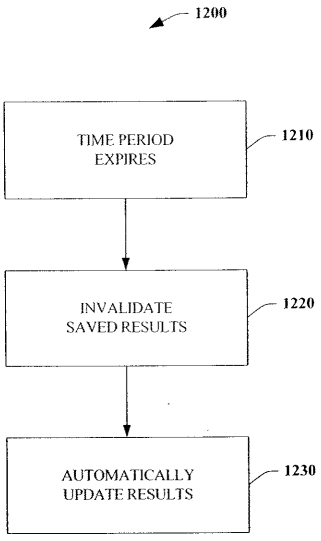


FIG. 12

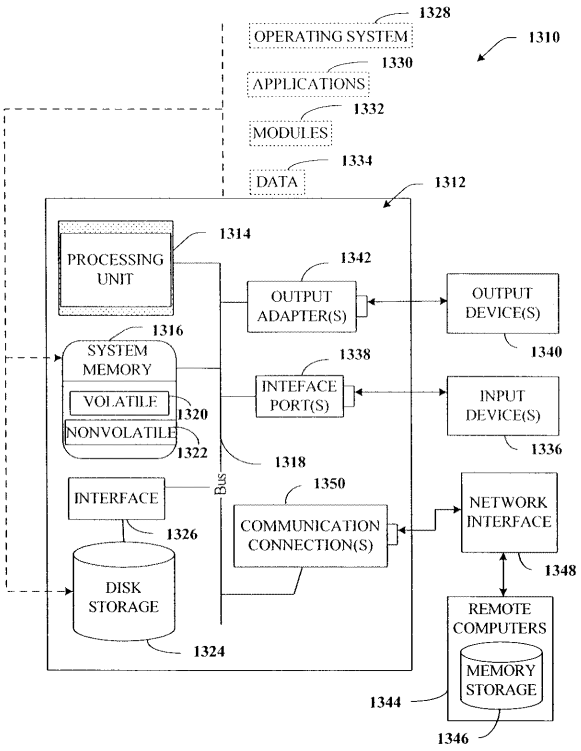

1328
OPERATING SYSTEM
1310
1330
APPLICATIONS
1332
MODULES
1334
DATA
1312
1314
PROCESSING UNIT
1342
OUTPUT ADAPTER(S)
OUTPUT DEVICE(S)
1340
1316
SYSTEM MEMORY
1338
INTEFACE PORT(S)
INPUT DEVICE(S)
1336
VOLATILE
1320
NONVOLATILE
1322
1318
Bus
INTERFACE
1350
COMMUNICATION CONNECTION(S)
NETWORK INTERFACE
1326
1348
DISK STORAGE
REMOTE COMPUTERS
MEMORY STORAGE
1324
1344
1346

FIG. 13