(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4777459号
(P4777459)
(45)発行日　平成23年9月21日(2011.9.21)　(24)登録日　平成23年7月8日(2011.7.8)

(51)Int.Cl.　F I
G06F 21/24 (2006.01)　G06F 12/14 520C
G06F 12/14 560B

請求項の数 10　(全 13 頁)

(21)出願番号	特願2009-513354 (P2009-513354)
(86)(22)出願日	平成19年4月23日 (2007.4.23)
(65)公表番号	特表2009-538489 (P2009-538489A)
(43)公表日	平成21年11月5日 (2009.11.5)
(86)国際出願番号	PCT/US2007/067213
(87)国際公開番号	WO2007/140060
(87)国際公開日	平成19年12月6日 (2007.12.6)
審査請求日	平成21年1月26日 (2009.1.26)
(31)優先権主張番号	11/420,657
(32)優先日	平成18年5月26日 (2006.5.26)
(33)優先権主張国	米国(US)

(73)特許権者　500280087
オートデスク，インコーポレイテッド
アメリカ合衆国　カリフォルニア州　９４９０３，サン　ラファエル，マキニス　パークウェイ　１１１
(74)代理人　100094318
弁理士　山田　行一
(74)代理人　100123995
弁理士　野田　雅一
(74)代理人　100107456
弁理士　池田　成人
(72)発明者　チェイス，　マイケル
アメリカ合衆国，　ミシガン州，　マディソン　ハイツ，　カンタベリー　ドライヴ　８０２

最終頁に続く

(54)【発明の名称】コンテンツ管理システム用のセキュリティアーキテクチャ

(57)【特許請求の範囲】
【請求項１】
コンテンツ管理システム用のセキュリティアーキテクチャを提供する方法であって、
アプリケーションストアドプロシージャによって行われたデータベース操作に関わる複数のオブジェクトそれぞれの識別子を監査ログに記録する前記アプリケーションストアドプロシージャを呼び出すステップと、
前記監査ログに記録された前記識別子を使用して、前記アプリケーションストアドプロシージャが前記データベース操作を行うことを許可されたかどうかを判定するアクセス制御チェックを行うように設定されたセキュリティチェックプロシージャを呼び出すステップと、
任意のアクセス違反エラーを戻すステップと
を含む方法。
【請求項２】
前記アプリケーションストアドプロシージャの前の呼び出しに関連付けられた情報を除去することにより、前記監査ログをクリーンアップするステップをさらに含む、請求項１に記載の方法。
【請求項３】
前記監査ログ、前記セキュリティチェックプロシージャ及びＡＣＬテーブルがデータベース管理システムに格納されたフレームワークコードの一部であり、さらに前記監査ログがアプリケーションコードにとって可視的である、請求項１に記載の方法。

【請求項４】
前記アプリケーションストアドプロシージャがデータベース管理システムに格納されたアプリケーションコードの一部である、請求項１に記載の方法。
【請求項５】
任意のアクセス違反エラーを表示するステップをさらに含む、請求項１に記載の方法。
【請求項６】
前記セキュリティチェックプロシージャが、前記監査ログ内の前記識別子を前記複数のオブジェクトの少なくともいくつか用のアクセス制御規則を規定するアクセス制御リスト（ＡＣＬ）内のエントリと比較することにより、前記アプリケーションストアドプロシージャが前記データベース操作を行うことを許可されたかどうかを判定する、請求項１に記載の方法。
【請求項７】
前記コンテンツ管理システムが、コンピュータ援用設計（ＣＡＤ）アプリケーションのユーザ用のＣＡＤコンテンツを格納するために使用される、請求項１に記載の方法。
【請求項８】
アクセス違反エラーに応答して、前記ストアドプロシージャの前記データベース操作を実行する前にあった動作状態にデータベースを戻すステップをさらに含む、請求項１に記載の方法。
【請求項９】
プロセッサによる実行時にコンテンツ管理システム用のセキュリティアーキテクチャを提供するための方法を実行するプログラムを含むコンピュータ可読媒体であって、前記方法が
アプリケーションストアドプロシージャによって行われたデータベース操作に関わる複数のオブジェクトそれぞれの識別子を監査ログに記録する前記アプリケーションストアドプロシージャを呼び出すステップと、
前記監査ログに記録された前記識別子を使用して、前記アプリケーションストアドプロシージャが前記データベース操作を行うことを許可されたかどうかを判定するアクセス制御チェックを行うように設定されたセキュリティチェックプロシージャを呼び出すステップと、
任意のアクセス違反エラーを戻すステップと
を含む、コンピュータ可読媒体。
【請求項１０】
コンテンツ管理システム用のセキュリティアーキテクチャを提供するためのシステムであって、
データベースと、
データベース操作に関わる複数のオブジェクトそれぞれの識別子を監査ログに記録する少なくとも１つの前記データベース操作及びコードを含むアプリケーションストアドプロシージャと、
前記アプリケーションストアドプロシージャが許可されたかどうかを判定するために複数のアクセス制御リスト（ＡＣＬ）テーブルをチェックすることにより前記監査ログのセキュリティチェックを行うセキュリティチェックプロシージャと、
前記アプリケーションストアドプロシージャを呼び出し、前記セキュリティチェックプロシージャを要求し、任意のアクセス違反エラーを戻すディレクタと
を含むシステム。
【発明の詳細な説明】
【発明の分野】
【０００１】
[0001]本発明は一般にコンピュータセキュリティ、データベース管理システム（ＤＢＭＳ）及びコンテンツ管理システム（ＣＭＳ）に関する。より具体的には、本発明はさまざまなアプリケーションプログラムによってアクセスされ、ＣＭＳに格納されるオブジェク

ト用のアクセス規則を強制する方法に関する。
【関連技術の説明】
【０００２】
[0002]一般に、ＣＭＳシステムは、実質的に任意の種類のデジタル情報（例えばファイル、ドキュメント、オブジェクト、テーブルなどのデータ）用の単一の安全なリポジトリを提供するために使用される。例えば、ＣＭＳシステムは、コンピュータ援用設計（ＣＡＤ）システムの複数のユーザによってアクセスされる設計データ、ファイル及び図面などを格納するために使用できる。集中化された場所にコンテンツを格納することにより、ユーザはそのコンテンツを検索、共有、参照及び再利用することができる。ＣＭＳは、マルチユーザシステムにおいてコンテンツの共同作成を編成及び容易にするためのコンピュータソフトウェアシステムである。通常、ＣＭＳはユーザコンテンツをファイルシステムに格納し、ファイルに関する情報をリレーショナルデータベースに格納する。ＣＭＳシステムが複数のユーザに対して広いアクセス可能性を提供できるため、ユーザがどのコンテンツにアクセスできるかを制限することが多くの場合望ましい。例えば、アクセス制限はアクセス制御リスト（ＡＣＬ）を使用して規定することができる。
【０００３】
[0003]ＡＣＬは、ＡＣＬ内で特定されたオブジェクト用のアクセス規則を強制するために使用される。ＡＣＬは、要求を行うユーザの属性（例えば、ユーザの同一性又はグループ内での地位）に基づき、そのユーザが所与のオブジェクトに対して適切なアクセス権を持つかどうかを判定する手段である。ＡＣＬは、プログラム、プロセス又はファイルなどの特定のシステムオブジェクトに対する個人ユーザ又はグループ権限を規定するエントリを有するデータ構造（例えばテーブル）である。特権又は許可は、ユーザがオブジェクトを読み取り、書き込み又は実行可能であるかどうかなどの特定のアクセス権を決定する。したがって、とりわけＡＣＬ内のエントリは、ユーザ又はユーザのグループがオブジェクトを変更又はアクセス又はその両方を行えるかどうかを制御する。
【０００４】
[0004]ＣＭＳにおいて十分なセキュリティを提供するために、ＡＣＬは、リレーショナルデータベース内の１つ又は複数のテーブルの個々の行と関連付けられる必要がある場合がある。これは、ＣＭＳシステムが、ファイルシステム内のオブジェクトを、それぞれＡＣＬ内のエントリとともに参照するためにデータベーステーブルの個々の行を使用できるために行われる。既存のＤＢＭＳシステムは、この種のアクセス制御のためのサポートを提供できない。その結果、必要なセキュリティを提供するための１つのアプローチは、ＣＭＳシステム内のコンテンツに関わる操作を行う際に、ユーザアプリケーションにアクセスチェックを行わせることである。ただし、アクセス制御とコアアプリケーション機能（例えばＣＡＤシステムの提供）は、通常は関連のない問題である。よく設計されたアプリケーションは、アクセス制御をアプリケーション機能から独立してコーディングすることにより、これらの問題の分離を実現するだろう。
【０００５】
[0005]ＣＭＳシステムにおいてアクセス制御を提供するために使用される２つの一般的なアプローチは、いくつかの難点に直面する。提案されたとおり、第１のアプローチでは、アプリケーションプログラマはアクセスチェックをＣＭＳ内のコンテンツにアクセスするアプリケーションの一部として明確にコーディングしてよい。例えば、アプリケーションは、ＣＭＳに格納されたコンテンツに対するアクセス制御機能を実行する構造化照会言語（ＳＱＬ）クエリ又はストアドプロシージャを含んでよい。ストアドプロシージャは、コンパイルされた形式でデータベースに格納される、割り当てられた名前を持つＳＱＬステートメントのセットである。この第１のアプローチは、アクセス制御をアプリケーションロジックに関連させるため、求められる各問題の分離という原則に反する。さらに悪いことに、ＳＱＬ言語で得ることのできる洗練度のレベルがむしろ低いことは、アクセスチェックを行うルーチンがデータベースにアクセスする各ファンクションで重複されるため、重複したコードをすべてのクエリ又はストアドプロシージャ全体にわたって分散させる

ことになる。

【０００６】

[0006]第２のアプローチは、データベースを、データ処理能力を持たない記憶装置として扱うロード－修正－格納アーキテクチャを使用するデータアクセスフレームワークを提供することを含む。この第２のアプローチは、高水準言語及びオブジェクト指向技術の使用を可能にしてアプリケーションプログラム問題をセキュリティ及びアクセス制御から分離する一方、このアプローチには、データベースとの小規模のデータ対話が多量にあることによるパフォーマンス上の問題がある。したがって、このアプローチは十分に利用できない。

【０００７】

[0007]したがって、アクセスチェックがデータベースで実行できるようにすると共に、アクセスチェックコードをアプリケーションコードから分離しておく方法が必要である。時間効率及び費用効率の高い解決策もまた必要である。

【発明の概要】

【０００８】

[0008]本発明の実施形態は、コンテンツ管理システムによって格納されたコンテンツ用のセキュリティ及びアクセス制御機能を提供するために使用される方法、装置及び製品を含む。

【０００９】

[0009]一実施形態は、コンテンツ管理システム用のセキュリティアーキテクチャを提供する方法である。この方法は、ストアドプロシージャによって行われたデータベース操作に関わる複数のオブジェクトそれぞれの識別子を監査ログに記録するアプリケーションストアドプロシージャを呼び出すステップと、監査ログに記録された識別子を使用して、アプリケーションストアドプロシージャがデータベース操作を行うことを許可されたかどうかを判定するためにアクセス制御チェックを行うように設定されたセキュリティチェックプロシージャを呼び出すステップと、任意のアクセス違反エラーを戻すステップとを概して含む。アクセス違反エラーが発生した場合、この方法は、ストアドプロシージャのデータベース操作を実行する前にあった動作状態にデータベースを戻すことをさらに含んでよい。

【００１０】

[0010]本発明の別の実施形態は、プロセッサによる実行時にコンテンツ管理システム用のセキュリティアーキテクチャを提供するための方法を実行するプログラムを含む、コンピュータ可読媒体である。

【００１１】

[0011]さらに別の実施形態は、アプリケーションストアドプロシージャ、セキュリティチェックプロシージャ及びディレクタを含む、コンテンツ管理システム用のセキュリティアーキテクチャを提供するためのシステムである。アプリケーションストアドプロシージャは、データベースアクセス操作に関わる各オブジェクトの識別子を監査ログに挿入するための、１つ又は複数のデータベースアクセス操作及びコードを含む。セキュリティチェックプロシージャは、アプリケーションストアドプロシージャが許可されたかどうかを判定するためにＡＣＬテーブルをチェックすることにより、監査ログのセキュリティチェックを行う。ディレクタはアプリケーションストアドプロシージャを呼び出し、セキュリティチェックプロシージャを要求し、任意のアクセス違反エラーを戻す。

【００１２】

[0012]有利なことに、クライアントアプリケーションのビジネスロジックを書く開発者は、セキュリティチェック命令をアプリケーションコードに繰り返し組み込む必要がない。適切なアクセスがあるかをチェックすることはシステムの問題、並びにフレームワーク及びインフラストラクチャの責務であり、アプリケーション開発者はチェックにとらわれることなくアプリケーション機能の実現に集中できる。

【図面の簡単な説明】

【００１３】
【図１】本発明の一実施形態を実施するための例示のシステム環境を示すブロック図である。
【図２】本発明の一実施形態による、ストアドプロシージャを実行し、セキュリティチェックを行うための例示の操作の流れを示す図である。
【図３Ａ】本発明の一実施形態による、ストアドプロシージャの実行を成功した場合の図である。
【図３Ｂ】本発明の一実施形態による、図３Ａのストアドプロシージャの実行を試み、アクセス違反となった場合の図である。
【好ましい実施形態の詳細な説明】
【００１４】
[0017]本発明の実施形態は、コンテンツ管理システム（ＣＭＳ）によって格納されたコンテンツ用に規定されたアクセス制御制限を強制するための方法、装置及び製品を提供する。したがって、本発明の実施形態は、コンピュータ援用設計（ＣＡＤ）アプリケーションなど、ＣＭＳによって格納されたデータ、ファイル又はオブジェクトにアクセスするアプリケーション用のＣＭＳにおいてセキュリティを提供する。ただし、本発明の実施形態はＣＡＤアプリケーション又はＣＡＤ環境に限定されるのではなく、むしろ本発明の実施形態は、ＣＭＳ内で任意の種類のコンテンツを格納する任意のアプリケーションに対してセキュリティ及びデータアクセス制御を提供するようになされてよい。
【００１５】
[0018]一実施形態は、２つの層が対話する最小限の「フックポイント」を使用してセキュリティ層をアプリケーション層から分離することによって問題を解決する。この「フックポイント」は、データベースに格納された監査ログテーブルであってよい。監査ログテーブルは、他の情報の中でとりわけ、データベース内の記録（例えば、ＣＡＤシステム内のファイル又はファイルのバージョン）及びユーザセッションを識別する固有のデータベース識別子（ＩＤ）を含む。実行する上で、アプリケーションがセキュリティ及びアクセス制御問題で検査されるべきトランザクションを要求する場合、アプリケーションはそのトランザクションに関わる任意のオブジェクトのＩＤを監査ログテーブルに記録する。システムは次にこのテーブルからＩＤを取り出し、セキュリティチェックを行う。このように、監査ログはセキュリティ問題とアプリケーション問題を効率的に分離し、アプリケーション開発者は過度にセキュリティの心配をすることなくアプリケーションコードを書くことができる。
【００１６】
[0019]この実施形態では、アプリケーション開発者が提供したストアドプロシージャは監査ログに情報を追加する。アプリケーション開発者はデータベースアクセスコードをストアドプロシージャ内に配置し、これらは後に実行するためにデータベースに格納される。開発者はこれらのストアドプロシージャを開発ツールのライブラリ及びアプリケーションプログラムインタフェース（ＡＰＩ）を使用して作成してよい。ライブラリ及びＡＰＩは開発者（例えば、第三者開発者又はアプリケーションプログラマから所与のＣＭＳシステムの一環の開発者まで）に提供され、提供された開発者がアプリケーションコードの形式でアプリケーション（例えばＣＡＤシステム）へのカスタマイゼーション又は拡張を作成できるようにする。アプリケーションコードの一種がストアドプロシージャである。ストアドプロシージャはデータを読み取り、データを修正し、新しいデータを作成又はファイルのチェックアウト、ファイルのコピー、ファイルの削除などのその他の操作を行うことができる。アプリケーションがＣＭＳシステムによって管理された情報、データ又はファイルに関わる操作を行う場合、ストアドプロシージャは後のセキュリティチェックのために、その操作に関わる各オブジェクトのすべての関連するＩＤ（すなわち、ストアドプロシージャによって何らかの方法で参照、アクセス又は接触されたデータのＩＤ）を監査ログに書き込む。アプリケーション開発者は、アプリケーションにセキュリティ関連の操作を行わせる必要がないため、アクセス制御チェックに（監査ログ内のＩＤの配置を越え

て）関与しない。

【００１７】

[0020]一実施形態では、システムは論理的に、（ただし、必ずしも物理的にではなく）階層間に対話がある階層モデルによって表される。クライアントアプリケーション層は、サーバーアプリケーション層と対話する。次にサーバーアプリケーション層は、データベースアプリケーション層及びサーバーフレームワーク層の両方と対話する。クライアントアプリケーション層はアプリケーションコードを含み、サーバーフレームワーク層はフレームワークコードを含む。ストアドプロシージャを呼び出すために、クライアントアプリケーション層はサーバーアプリケーション層に要求を渡し、サーバーアプリケーション層はその要求をサーバーフレームワーク層に渡す。次に、サーバーフレームワーク層は、ストアドプロシージャを呼び出し、適切なアクセス制御チェックを行い、階層化フレームワークを通じて任意の結果を戻してよい。

【００１８】

[0021]図１は、本発明の本実施形態を実施するための例示のシステム環境１００を示す。システム１００は、ＣＡＤシステムを含む、ＣＭＳを有する任意の種類のシステムであってよい。システム１００では、クライアントアプリケーション１０２がコンテンツ管理システム（ＣＭＳ）１０４と対話して、データを読み取り、データを修正し、新しいデータを作成し、又は、ファイルのチェックアウト、ファイルのコピー、ファイルの削除などのその他の操作を行い、又はＣＭＳ１０４からのコンテンツを使用してクライアントアプリケーション１０２が提供する他の任意の機能を行う。一実施形態では、Ａｕｔｏｄｅｓｋ社から入手できるＡｕｔｏｄｅｓｋ　Ｖａｕｌｔコンテンツ管理システムを使用してよい。

【００１９】

[0022]図１に示す構成要素は、デスクトップコンピュータ、サーバーコンピュータ、ラップトップコンピュータ、タブレットコンピュータなどの既存のコンピュータシステム用に設定されたコンピュータソフトウェアアプリケーションを含む。ただし、本明細書内で記載するアプリケーションは、任意の特定のコンピューティングシステムに限定されず、新しいコンピューティングシステムが入手できるようになった場合にはそのシステムを利用するようになされてよい。

【００２０】

[0023]さらに、システム１００で示す構成要素は、ローカルエリアネットワーク又はインターネットなどの大規模な広域ネットワークを含むコンピュータネットワークを介して通信する分散システムで実行されるソフトウェアアプリケーションであってよい。例えば、グラフィカルユーザインタフェース１０６及びクライアントアプリケーション１０２は、コンピュータネットワーク（例えば、オフィスＬＡＮ、企業イントラネット又はインターネット）を介してＣＭＳシステム１０４と通信するクライアントコンピュータシステムで実行されるソフトウェアプログラムを含んでよい。また、図１に示す構成要素は、ＣＤ－ＲＯＭ、ＤＶＤ－ＲＯＭ、フラッシュメモリモジュール又は他の有形の記憶媒体などのコンピュータ可読媒体に格納されたアプリケーションプログラム（１つ又は複数）として提供されてよい。

【００２１】

[0024]図示されるとおり、クライアントアプリケーション１０２は、グラフィカルユーザインタフェース（ＧＵＩ）１０６、表示装置１０８及び入力装置１１０を含む。例えば、ＣＡＤシステムでは、ＧＵＩ１０６は設計図を表示装置１０８に表示し、入力装置１１０を使用してアクセス可能な、設計図を作成、修正、共有及び維持するための機能を提供してよい。そのような場合、ＣＭＳ１０４は、ＣＡＤシステム１０４のためのコンテンツを管理する。通常、入力装置１０８は、マウスポインティングデバイス及びキーボードを含み、表示装置１０８はＣＲＴモニタ又は液晶ディスプレイである。ＣＭＳシステム１０４に格納されたコンテンツにアクセスするためにクライアントアプリケーション１０２と対話するユーザの例を図３Ａ及び図３Ｂに示す。

【００２２】
[0025]一実施形態では、ＣＭＳシステム１０４は、サーバー１１２及びデータベース管理システム（ＤＢＭＳ）１１４を含む。サーバー１１２はクライアントアプリケーション１０２に対してサービス（例えば、ファイルのチェックイン、チェックアウト）を提供し、ＤＢＭＳ１１４は、アプリケーションのデータを管理（例えば、ＣＡＤシステム用の設計図の格納及びストアドプロシージャの格納）する。ＣＭＳ１０４内で、サーバー１１２及びＤＢＭＳ１１４の両方はいくつかのアプリケーションコード１１６、１２０及びいくつかのフレームワークコード１１８、１２２を含む。アプリケーションコード１１６、１２０は、クライアントアプリケーション１０２（例えばＣＡＤシステム）の機能を提供し、フレームワークコード１１８、１２２はＣＭＳ１０４の機能（例えば、ＣＡＤシステム用のコンテンツの安全な管理）を提供する。

【００２３】
[0026]図示のとおり、サーバー１１２はサーバーアプリケーションコード１１６及びサーバーフレームワークコード１１８を含む。クライアントアプリケーション１０２と対話するユーザが操作を要求、例えばＣＭＳ１０４が格納する設計図をチェックアウトする場合、クライアントアプリケーション１０２は、サーバーフレームワークコード１１８に対して、その設計図に関連付けられたオブジェクトをＣＭＳ１０４からチェックアウトすること求める要求を送信する。チェックアウト操作を完了（適切なアクセス制御チェックの実行を含む）すると、サーバーフレームワークコード１１８は、クライアントアプリケーション１０２に対して要求されたオブジェクトを提供し、これによりＧＵＩ１０６はその設計図を表示装置１０８に表示できる。

【００２４】
[0027]ＤＢＭＳ１１４は、アプリケーションコード１２０及びフレームワークコード１２２用の分離したデータベースプロシージャ及びテーブルを含んでよい。アプリケーションデータベースプロシージャ及びテーブル１２０は、アプリケーション関連機能を実行するために使用されるストアドプロシージャ１２４を含む。フレームワークデータベースプロシージャ及びテーブル１２２は、クライアントアプリケーション１０２及びＣＭＳ１０４間の「フックポイント」として使用される監査ログテーブル１２６を含む。ＣＭＳ１０４がストアドプロシージャ１２４を、例えば設計図をチェックアウトするために呼び出す場合、そのストアドプロシージャ１２４は、チェックアウトされている設計図に関連付けられたＩＤを監査ログテーブル１２６に記録する。次に、フレームワークデータベースプロシージャ及びテーブル１２２内のセキュリティチェックプロシージャが、クライアントアプリケーション１０２が要求された設計図をチェックアウトするための適切なアクセス特権を持つことを確実にするために、これらのＩＤを使用してセキュリティチェックを行う。

【００２５】
[0028]さらに概して、クライアントアプリケーション１０２がＣＭＳ１０４によって格納されたオブジェクトに関わるトランザクションを行う場合、サーバーアプリケーションコード１１６が、オブジェクトのＩＤを監査ログテーブル１２６に記録するように設定されてよいストアドプロシージャ１２４を呼び出す。これを行うことにより、オブジェクトのＩＤをデータベースプロシージャ１２２に効果的に「渡し」、データベースプロシージャ１２２がそのトランザクションに関わる各オブジェクトについてのアクセス制御チェックを行えるようにする。アプリケーションコード１１６、１２０は、ストアドプロシージャ１２４の１つを呼び出すようにフレームワークコード１１８、１２２に要求する。ストアドプロシージャ１２４の実行後、フレームワークコード１１８、１２２は、ストアドプロシージャ１２４によって監査ログテーブル１２６に挿入されたＩＤについてのアクセス制御チェックを行うために、セキュリティチェックプロシージャを実行する。このアクセス制御チェックは、ストアドプロシージャ１２４によって行われるトランザクションを行ってよいかどうかを決定する。

【００２６】

[0029]アクセス違反が発生したとシステムが特定した場合、フレームワークコード１１８、１２２はさまざまな方法で応答してよい。例えば、システム１００がデータを修正する許可を拒否する場合、システム１００は、標準ＳＱＬメカニズムを使用してストアドプロシージャ１２４によって行われたトランザクションを取り消してよく、トランザクションの前にあった状態にシステム１００をロールバックできるようにする。いくつかの場合、ストアドプロシージャを実行する前にセキュリティをチェックすることとは対照的に、ストアドプロシージャ１２４を実行し、セキュリティをチェックし、必要な場合にロールバックを行うことが好ましいだろう。例えば、マルチステップトランザクションの各ステップでセキュリティチェックを行うことは非効率である可能性がある。さらに、ストアドプロシージャ１２４を行った後にセキュリティチェックを行うことにより、システム１００はクライアントアプリケーション１０２をセキュリティの詳細から遮蔽し、代わりにシステム１００は、クライアントアプリケーション１０２に対してアクセス違反があった場合にのみ通知する。同様に、フレームワークコード１１８、１２２は、ストアドプロシージャによって行われたトランザクション若しくは操作又はクライアントアプリケーション１０２の操作の詳細を知る必要はない。代わりに、フレームワークコード１１８、１２２は、特定のトランザクションに関わるオブジェクトの監査ログ１２６内のＩＤを評価するだけでよい。このように、システム１００はセキュリティ問題をアプリケーション問題から分離する。

【００２７】

[0030]クライアントアプリケーションに監査ログテーブル１２６へのＩＤの追加の責務を割り当てることは、アクセスチェックとアプリケーション問題とを多少関連させることを意味するものの、この多少の関連は、フレームワークコード１１８、１２２が監査ログテーブル１２６をアクセスチェックできるようにする。ＡＣＬをどのように表し、格納するか及びＡＣＬをどのように、いつチェックするかについての詳細は、フレームワークコード１１８、１２２のみの問題であり、それらの詳細はアプリケーションコード１１６、１２０の問題ではない。したがって、アクセスチェックはアプリケーションコード１１６、１２０の一部にはならない。これにより、開発者は、自らのビジネスロジックにセキュリティを意識したコードを組み込む必要がないため、より迅速にアプリケーションを構築できるようになる。問題を分離することにより、セキュリティコードが集中化され、多くのコードの部分に重複されないため、ソフトウェア保守も単純化する。さらに、システム１００は、アクセス制御データ、ＡＣＬ及びアクセスチェックコードをＤＢＭＳ１１４に格納することによりデータベースを出入りするデータの量を最小限にする。さらに、アクセス制御規則を強制することに加え、監査ログ１２６は、クライアントアプリケーション１０２及びＣＭＳ１０２間で多くのシステム機能に関連する情報を渡すための有用な「フックポイント」を提供し、いずれの場合にもアプリケーション開発者が関係することを必要としない。

【００２８】

[0031]図２は、本発明の一実施形態による、ストアドプロシージャを実行し、セキュリティチェックを行うための例示の操作の流れ２００を示す。図２に示す操作の流れ２００を任意の順序で行うように設定された任意のシステムは、本発明の範囲に含まれることを当業者は理解するであろう。

【００２９】

[0032]この例では、アプリケーションストアドプロシージャ２０４を実行し、セキュリティチェックを行うために次の構成要素が対話する。すなわち、ディレクタ２０２、アプリケーションストアドプロシージャ２０４、セキュリティチェックプロシージャ２０６及び監査ログ２０８である。一実施形態では、ディレクタ２０２は、図１に示すフレームワークコード１１８、１２２の一部である。アプリケーションストアドプロシージャ２０４はアプリケーションコード１１６、１２０の一部であり、データベース１１４に格納される。セキュリティチェックプロシージャ２０６はフレームワークコード１１８、１２２の一部であり、これもまたデータベース１１４に格納される。監査ログ２０８はフレームワ

ークコード１１８、１２２の一部であるものの、アプリケーションコード１１６、１２０に対しアクセス可能である。

【００３０】

[0033]ディレクタ２０２は、既知のディレクタ／ビルダソフトウェア設計パターンの一部である。このパターンは、同一の構築プロセスが異なる表示を作成できるように複合オブジェクト（又はオブジェクトのセット）の構築とオブジェクトの表示とを分離するためのコンピュータソフトウェアコンセプトである。他の実施形態では他のパターン又は構築物を使用してよい。一実施形態では、ディレクタ２０２はフレームワークコード１１８、１２２の一部であり、ビルダ（図示せず）はアプリケーションコード１１６、１２０の一部である。当業者によって理解されるとおり、ディレクタ／ビルダパターンは、次のコンセプトを定義する。すなわち、ビルダ、プロダクト、ディレクタ２０２及び具体物ビルダである。ビルダはオブジェクトの一部（プロダクト）を作成するための抽象的なインタフェースである。プロダクトは構築中の複合オブジェクトを表す。ディレクタ２０２はビルダインタフェースを使用してオブジェクトを構築する。そして具体物ビルダはビルダインタフェースを実行することにより、プロダクトの一部を構築及びアセンブルする。

【００３１】

[0034]ストアドプロシージャのデータアクセスルーチンは、１つ又は複数の異なるタイプのオブジェクトのセットを戻す。ビルダは、ストアドプロシージャによって戻された結果セットの１つから各オブジェクトを構築する。結果セットの取得及び処理、アクセス制御のチェック並びにオブジェクトの構築は概してディレクタ２０２の責務である。したがって、ディレクタ２０２は、結果の構築並びに結果セット取得のプロセスの両方を管理する。一実施形態では、開発者がデータアクセスルーチンをコーディングする際、開発者はディレクタ２０２をインスタンス化してそれにストアドプロシージャのパラメータを提供し、１つ又は複数のビルダをインスタンス化してそれらをディレクタ２０２に追加し、次いでディレクタ２０２を実行する。トランザクションが複数の結果セットを含み得る場合、開発者は、１つのビルダオブジェクトが結果セットの１つを処理する責務を負う状態でビルダ（又はビルダ群）をディレクタ２０２に追加する。

【００３２】

[0035]上述のとおり、監査ログ２０８は、セキュリティ問題及びアプリケーション問題を分離する「フックポイント」を提供する。ストアドプロシージャ２０４のそれぞれは、ストアドプロシージャ２０４のそれぞれの操作によって影響される各オブジェクト（結果として生じる任意のオブジェクトを含む）のＩＤを監査ログ２０８に記録する。ストアドプロシージャ２０４は、オブジェクトＩＤをセッションＩＤ（又はスレッドＩＤ）とともに監査ログ２０９に記録してよい。セッションＩＤは、システムが異なるトランザクション間を区別できるようにする。セッションＩＤは、監査ログ２０８にオブジェクトＩＤを記録するユーザ又はプロセスの同一性に基づいてよい。一実施形態では、システム１００は他の属性を監査ログ２０８に追加してよい。一実施形態では、所与のトランザクションに関連して、監査ログ全体が一度にアクセスチェックされる。さらに概して言うと、当業者は多くの場合、パフォーマンスオーバーヘッドを最小限にする方法でシステム１００及び監査ログ２０８を実現するであろう。

【００３３】

[0036]図２に示すとおり、ステップ２１０でディレクタ２０２は、アプリケーションストアドプロシージャ２０４に加えて、ストアドプロシージャ２０４用に必要な任意の引き数を呼び出す要求を受け取る。例えば、クライアントアプリケーション１０２は、ディレクタ２０２に対してＣＭＳ１０４が管理するオブジェクトをチェックアウトするプロシージャを呼び出すことを要求できる。ステップ２１２では、ディレクタ２０２は、アプリケーションストアドプロシージャ２０４の前の実行からの任意の情報を除去することにより監査ログ２０８をクリーンアップする。図に示すとおり、ディレクタ２０２は操作の流れ２００の最初で監査ログ２０８をクリーンアップする。ただし代替として、ディレクタ２０２はトランザクションが完了したときに監査ログ２０８をクリーンアップしてよい。

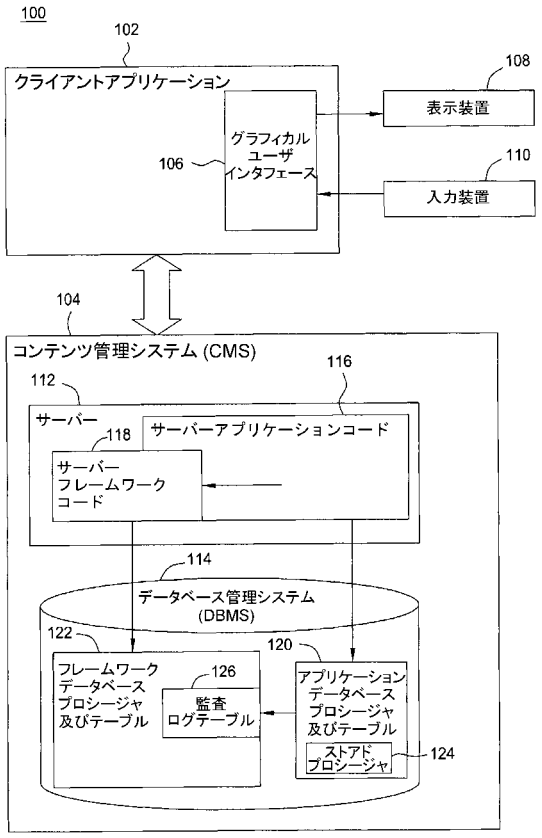
【００３４】
[0037]ステップ２１４で、ディレクタ２０２は要求されたトランザクションの実行を開始し、ステップ２１６でアプリケーションストアドプロシージャ２０４を呼び出す。ステップ２１８で、アプリケーションストアドプロシージャ２１４はトランザクションに関わるオブジェクトのオブジェクトＩＤを監査ログ２０８に記録する。ステップ２２０で、アプリケーションストアドプロシージャ２０４は、ストアドプロシージャ２０４の結果をディレクタ２０２に戻す。
【００３５】
[0038]ステップ２２２で、ディレクタ２０２はセキュリティチェックプロシージャ２０６を呼び出し、監査ログ２０８内のオブジェクトＩＤを使用してアクセス制御チェックを行う。一実施形態では、セキュリティチェックプロシージャ２０６は、ＡＣＬを使用してセキュリティチェックを行う。この実施形態では、アクセス制御リスト（ＡＣＬ）を表すためにデータベーステーブルを使用してよい。セキュリティチェックプロシージャ２０６は、評価されるトランザクションに関わるオブジェクトのユーザ及びＡＣＬに対する役割ベースの許可をチェックする。セキュリティチェックプロシージャ２０６は、オブジェクトのセットの許可を個別にではなく一度に評価してよい。一度に評価することにより、メモリにオブジェクトをロードし、それから戻って許可をロードすることを回避できる。別の実施形態は、オブジェクトとともにＡＣＬをロードすることを回避する。他の実施形態では、ＡＣＬを他の方法で実現できることを当業者は認識するであろう。ステップ２２６では、監査ログ２０８はセキュリティチェックプロシージャ２０６によってクリーンアップされる。ステップ２２８では、アクセス違反が発生した場合、セキュリティチェックプロシージャ２０６はディレクタ２０２に通知する。それに応答して、ステップ２３０でディレクタ２０２はトランザクションを中止し、ステップ２３２でアクセス違反エラーをアプリケーションコード１１６に戻すことができる。アプリケーションコード１１６は、アクセス違反を任意の適切な方法で処理してよい。例えば、アプリケーションコード１１６は、要求されたトランザクションが十分なアクセス権又は特権を欠くために失敗したことをユーザに通知するように設定されてよい。
【００３６】
[0039]図３Ａは、本発明の一実施形態による、ストアドプロシージャの実行を成功した場合を示す。図３Ａは、ストアドプロシージャを実行するための例示のＧＵＩ３００を示す。他の実施形態では、他のインタフェースを使用するか、全くインタフェースを使用しなくてもよい。この図では、ユーザは実行するストアドプロシージャをリスト３０２から選択してよい。ここではユーザは、ハイライトされている「パスによってフォルダを取得」３０４というストアドプロシージャを選択している。このストアドプロシージャは、ユーザがＧＵＩの右上部３０６に入力するディレクトリパスをパラメータとして取る。ここで、ユーザはディレクトリパスとして「Ｃ：＼Ｍｙ　Ｐｒｏｊｅｃｔ」と入力し、次に「実行して結果を表示」ボタン３０８を選択している。結果は、ストアドプロシージャの実行が成功したことを示すステータス「フォルダ取得済み」３１２とともに表示３１０されている。
【００３７】
[0040]図３Ｂは、本発明の一実施形態による、図３Ａのストアドプロシージャの実行を試み、アクセス違反となった場合を示す。図３Ｂは同じストアドプロシージャの実行を試み、アクセス違反がある場合の結果を示す。この場合、結果は表示されず３１４、ストアドプロシージャの実行が失敗したことを示すステータス「アクセス違反」３１６が表示される。当然、他の実施形態も異なる方法で結果及びステータスを表示してよく、図３Ａ及び図３Ｂは単に１つの可能な方法を示すにすぎない。
【００３８】
[0041]開示した本発明の例示の実施形態は、コンテンツ管理システム用のセキュリティアーキテクチャを提供するための方法、装置及び製品を含む。一実施形態では、セキュリティフレームワークはデータベース内のファイル管理システム内のフォルダ及びフォルダ

のコンテンツに対してアクセス制御を適用する。さまざまな実施形態は、異なる細分性のレベルで、例えばデータベース内の個別のオブジェクトに対してアクセス制御を行う。いくつかの実施形態は、多くの異なるアクセス制御モデルを可能にする柔軟性を提供する。それらの実施形態では、セキュリティチェックがアプリケーション層に透過的であるようにセキュリティチェックをフレームワーク層の責務にする。実施形態は、オブジェクトレベルのアクセスチェックのパフォーマンスインパクトを低減する監査ログを含む。実施形態では、セキュリティシステムの重複及び可視性を最小化することによりセキュリティ問題をアプリケーション問題（例えばビジネスロジック）から分離する。有利なことに、ビジネスロジックを書く開発者は繰り返しセキュリティチェック命令をアプリケーションコードに組み込む必要がない。適切なアクセスがあるかをチェックすることはシステムの問題であり、フレームワーク及びインフラストラクチャの責務であり、アプリケーション開発者はチェックにとらわれることなくアプリケーション機能の実現に集中できる。

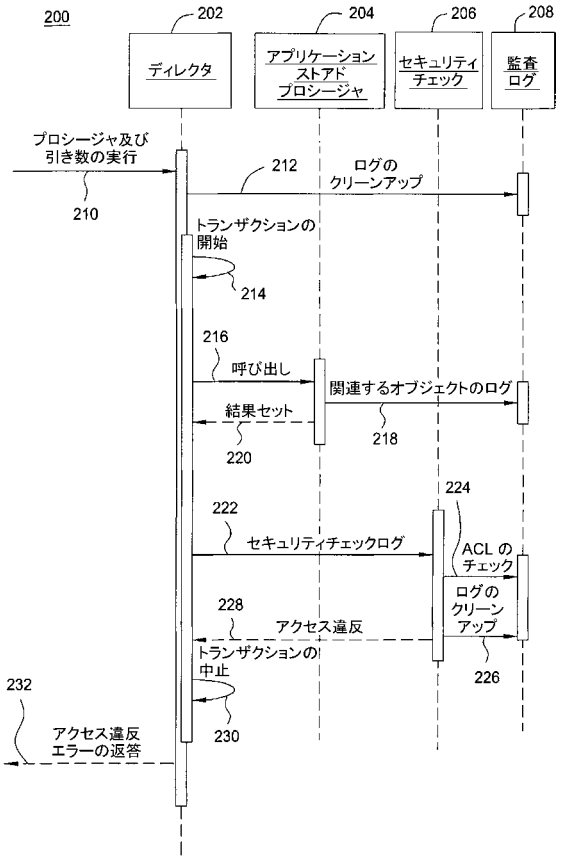
【００３９】

[0042]上述の事項は本発明の実施形態を対象とするものの、本発明の他の及び追加の実施形態が本発明の基本的な範囲から逸脱することなく考案されてよく、本発明の範囲は添付の特許請求の範囲によって決められる。

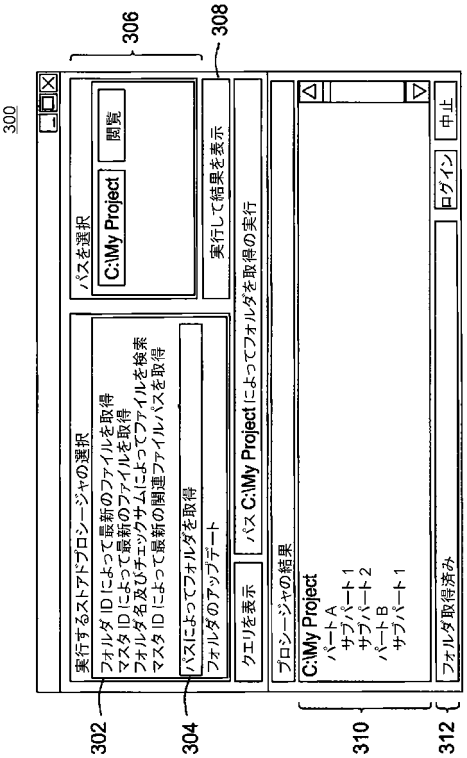
【図１】



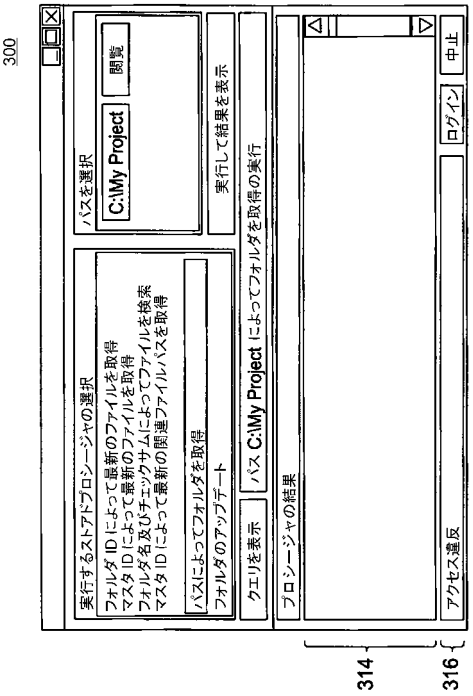
【図２】



【図３Ａ】


300
306
308
実行するストアドプロシージャの選択
フォルダ ID によって最新のファイルを取得
マスタ ID によって最新のファイルを取得
フォルダ名及びチェックサムによってファイルを検索
マスタ ID によって最新の関連ファイルパスを取得
パスによってフォルダを取得
フォルダのアップデート
パスを選択
C:\My Project
閲覧
実行して結果を表示
クエリを表示
パス C:\My Project によってフォルダを取得の実行
プロシージャの結果
C:\My Project
パート A
サブパート 1
サブパート 2
パート B
サブパート 1
ログイン
中止
フォルダ取得済み
302
304
310
312

【図３Ｂ】


300
実行するストアドプロシージャの選択
フォルダ ID によって最新のファイルを取得
マスタ ID によって最新のファイルを取得
フォルダ名及びチェックサムによってファイルを検索
マスタ ID によって最新の関連ファイルパスを取得
パスによってフォルダを取得
フォルダのアップデート
パスを選択
C:\My Project
閲覧
実行して結果を表示
クエリを表示
パス C:\My Project によってフォルダを取得の実行
プロシージャの結果
ログイン
中止
アクセス違反
314
316

フロントページの続き

審査官　和田　財太

(56)参考文献　特表２００７－５１１８２１（ＪＰ，Ａ）
特開２００６－１０７５０４（ＪＰ，Ａ）
米国特許出願公開第２００４／００９７４４１（ＵＳ，Ａ１）
米国特許出願公開第２００４／０２２５９６８（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F　21/24
G06F　 9/54