社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－DPS－132 (14)
2007－GN－ 65 (14)
2007－EIP－ 37 (14)
2007／9／20

Web/AP/DB三層モデルにおけるアクセス検査の効率化

山中 真和† 榎本 俊文† 谷口 展郎† 山室 雅司†

† NTT サイバースペース研究所 〒239–0847 神奈川県横須賀市光の丘 1–1
E-mail:
†{yamanaka.masakazu,enomoto.toshifumi,taniguchi.noburou,yamamuro.masashi}@lab.ntt.co.jp

あらまし SOx 法等に対応する内部統制の確立に向けては，複数の業務 AP とそれらがアクセスする DB に対し統合的なアクセス検査を行うことで，業務プロセスの正当性を検証できる．ただし，アクセス要求時に合わせて検査を実施する場合，業務 AP のレスポンス性能低下が問題となる．本論文では，Web/AP/DB 構成システムにおける各アクセス要求を効率的に統合・検査するための方式を提案する．本方式では，検査用データを格納する DB に対し，参照・書込み及び COMMIT 実行回数を削減することで，効率化を図った．プロトタイプを通した性能評価では，本方式の適用によりレスポンスやスループットを約 265 倍と大幅に改善した．検査を実施しない場合と比較しても，5%程度までオーバヘッドを抑制することが出来た．

キーワード マルチスレッド，セキュリティ，データベース，

Runtime Access Inspection for Three Tier System with Efficiency

Masakazu YAMANAKA†, Toshifumi ENOMOTO†, Noburou TANIGUCHI†, and Masashi YAMAMURO†

† NTT Cyber Space Laboratories Hikarinooka 1–1, Yokosuka-shi, 239–0847 Japan
E-mail:
†{yamanaka.masakazu,enomoto.toshifumi,taniguchi.noburou,yamamuro.masashi}@lab.ntt.co.jp

Abstract In multiple business systems, the access inspection of AP/DB servers enables to validate the process in the system, and is helpful to realize internal control. The runtime inspection will need more time to return responses in the business application. This paper proposes the techiques to integrate/inspect the three tier system efficiently. To reduce runtime overheads for inspections, we reduce the cost of read/write and COMMIT operations against DB, keeping data for inspections. Compared with an ordinary approach, our approach shows a highly improved response and throughput. It only needs 5% runtime overheads against programs without the inspection.

Key words Multithreading, Security, Database

1. はじめに

2008 年施行予定の金融商品取引法 (通称日本版 SOX 法) により，企業において内部統制の強化・対策の必要性が高まっている ([1]~[3]). これに伴い，社内で利用される業務アプリケーション (以下，業務 AP) においても，各種ログの記録や監視などによるシステム挙動の動作保証が必須となっている．

文献 [2] では，内部統制の 6 つの基本的要素として，(1) 統制環境，(2) リスクの評価と対応，(3) 統制活動，(4) 情報と伝達，(5) モニタリング，(6)IT への対応が挙げられており，これらは内部統制確立の基礎となっている．これら 6 つの基本的要素における "モニタリング" と "リスクの評価と対応" の実現には，業務 AP の動作

保証を目的とした各種ログの記録や監視が重要である．

”モニタリング”とは，内部統制の有効性を継続的に監視及び評価するプロセスを意味し，業務での各プロセスを証跡として残すことが重要になる．IT システムにおいては，システムログの記録がこれにあたる．

次に，”リスクの評価と対応”は，組織の目標の達成に影響を与える全てのリスクを識別，分析及び評価することによって，当該リスクへの対応を行う一連のプロセスを言う．IT システムにおいては，システムに対する継続的な監視がこれにあたる．

監査実務の観点からは，業務 AP の実行前に，要求されたアクセスの内容を検査すること (以下，”アクセス検査”と表記) が重要である ([4])．業務 AP の実行前に検査を行った場合，検査を経たアクセスのみが実行されるため，不正操作を防止することができる．一方で，記録したアクセスログを利用し，後々，不正操作を検査・発見することは，事前防止を補完する統制行為にすぎない．

業務 AP に対するアクセスを詳細に検査するためには，複数サーバ間に跨ったアクセスや過去に行われたアクセスについての情報を統合して検査する必要がある．例えば，業務 AP で，一般的に良く利用される Web サーバ，アプリケーションサーバ (以下 AP サーバ), DB サーバの構成において，Web サーバへのアクセスのみでなく，AP や DB サーバのアクセスも含めた全アクセスを検査に利用することが考えられる．この時，不正操作の防止を目的に，これらの情報を利用した検査をアクセス要求時に実施する場合，検査による業務 AP のレスポンス低下をいかに抑えるかが課題になる．

本論文では，Web/AP/DB 構成システムにおいて各種アクセス内容の統合・検査をリアルタイムに行うための手法を提案する．検査時に必要となるデータをメモリ上でバッファ・キャッシュすることで，高速化を図る．加えて，検査の実行途上で不要となったデータを順次捨てることで効率化を図る．また，検査データ格納用 DB に対する参照・書き込みの内容に応じて，COMMIT 実行回数を削減する手法について述べる．

論文構成は，以下の通りである．まず，2. 章で，アクセス検査に必要な処理や課題を整理する．3.,4. 及び 5. 章にて，提案システムにおける解決方針及び本検査方式について述べる．6. 章にて，性能評価を行い，最後に，7. 章にてまとめる．

2. アクセス検査における課題

本章では，内部統制の確立に有効となるアクセス検査を実現するための要求事項についてまとめ，それら要求事項を満たす際の課題を整理する．

2.1 検査を実現する上での要求事項

業務アプリケーションとして図 1 で示す Web/AP/DB 構成を考える．この場合，各アクセスに対する検査の要件としては，

- 複数マシンに跨ったアクセスの統合
- 過去に行われたアクセスの情報を利用した検査
- 予防的な検査 (内部統制からの要請)

を全て実現することである．

複数マシンに跨ったアクセスの統合: 一般的に良く利用される Web/AP/DB サーバ構成 (図 1) では，その検査対象が複数システムに跨ってしまう．そのため，良く実施される Web サーバへのアクセス検査にとどまらず，AP や DB サーバへのアクセスを含めた全アクセスに対する検査が必要となる．特に，DB には，個人情報や金銭に関わるデータなどの重要な情報が格納されることが多いため ([5])，DB へのアクセスを含めたアクセス全体を検査することは重要である．

業務 AP が内部で複数システムに跨る例としては，AP や DB サーバを複数利用している場合や，単一の業務プロセスが，内部的には複数の業務 AP から構成されている場合が挙げられる．例えば，図 1 のように旅費に関するシステムや文書決済に関するシステムから，業務プロセスが構成される場合である．より詳細な検査を実現するためには，一つの業務 AP 内部に閉じた検査だけではなく，各業務 AP に対するアクセス内容を全て利用することが重要である．

過去のアクセス情報を利用した検査: アクセス内容を検査する場合，単独のアクセスについての情報のみ利用して検査を実施することが多い．例えば，単純に，アクセス内容に攻撃コマンドが含まれているかどうか検査する場合，そのアクセス内容を利用するだけで良い．

一方で，時系列的に複数の処理が関連する場合，単独のアクセスに限らず関係する過去のアクセスについての情報が検査に必要となる．例えば，”見積”，”申請”，”承認”，”発注”，”検収確認”，”支払い処理”というプロセスを持つ購買業務 AP を想定した場合，個別のアクセス内容だけでは，不十分である．この場合，不正な処理を発見するためには，実行された各プロセスの順序が正しいかどうか (例えば承認を待たず発注を行っていないかどうか) の検査が必要になる．このような検査では，過去に申請が行われたかどうか，つまり，過去に行われたアクセスについての情報が必要になる．

予防的な検査 (内部統制からの要請): 内部統制において，不正な操作のようなリスクをコントロールする方法として，予防的コントロールと発見的コントロールが

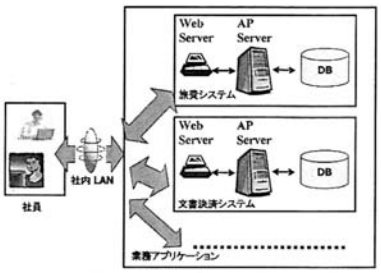

図 1　業務 AP の構成例

存在する．予防的コントロールとは，予め不正な操作といったリスクに対して，それらが発生しないよう整備されたコントロールのことを言う．発見的コントロールとは，不正な操作のような事象が起きたときに，それらを発見することを意図して整備されたコントロールのことを意味する．内部統制においては，予防的コントロールがより重要視される ([4]).

2.1.1　課題の洗い出し

予防的なアクセス検査を実現するためには，Web サーバへのアクセス要求に対して，その内容を検査し終えてからでないとアクセス要求を AP サーバへ届け，実行を完了させてはならない．そのため，検査による業務 AP のレスポンス低下をいかに抑えるかが課題である．

検査対象とするデータである過去に行われたアクセスについての情報は，膨大な量になる．これは，内部統制という観点から，数ヶ月程度の期間におけるアクセス内容を対象とした検査が必要と考えられるためである．検査に必要なデータが膨大になり，全てのデータをメモリ上に保持できず，検査を実施する上で I/O 処理が発生し，ボトルネックを生じやすい．このため，アクセス内容の検査に時間が掛かり，業務 AP 全体のレスポンス低下に繋がってしまう．

3.　提案システム概要

アクセス検査を実現する上で，複数のマシン上に分散したアクセスや過去に発生したアクセスを統合・利用した検査を実現しつつ，検査に伴うアプリケーションのレスポンス低下を最小限に抑えることが重要である．我々は，これを満たす検査システムの実現に向け，検査処理に伴うレスポンス低下を抑止するための手法について述べる．

検査システムを実現する際，既存の業務 AP と独立に，ネットワークを監視するシステム (図 2) を配置することを提案する．検査システムを業務 AP と独立させることで，検査を既存の業務 AP において実現する場合，AP の改変を最小限に抑えることができる．加えて，検

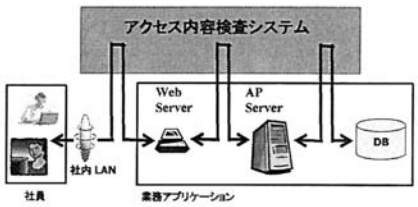

図 2　アクセス検査

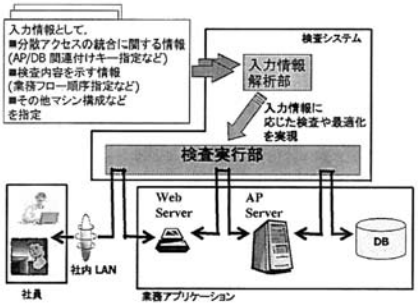

図 3　検査システムイメージ

査に関わる処理を検査サーバ内で集約できるため，AP 内で実装した場合に比べ，実装コストを削減できる．

本論文では，検査システム実現に向け，検査処理に伴うレスポンス低下を抑止するための方式について述べる．

3.1　アクセス検査システム

想定する検査システムでは，検査内容についての指定を受け取り，各業務 AP に応じた検査を実現する (図 3)．これは各業務 AP 毎に業務内容が異なり，検査すべき内容も異なるためである．検査システムが必要とする情報は，検査内容の指定に加えて，分散したアクセス内容の統合方法の指定などである．これらの情報を与えるだけで，各業務 AP に応じた詳細な検査が実現できる．

現在，検査指定を行うための記述については検討中であるため，その概要のみ示す．

3.2　検査システムへの入力情報

分散したアクセスの統合に関する情報

分散したアクセスを統合化するために，

- 各アクセスのフォーマットの定義
- 各アクセスのどの部分が関連付けキーであるか

の記述を想定している．ここでの関連付けキーとは，例えば，4. 章における AP/DB 関連付けキーなどである．さらに，各アクセスの実際の値から，関連付けを行うためのキー値を計算するような場合にも対応することを考えている．

検査内容を示す情報

実際に行う検査の指定には，パターン指定を想定して

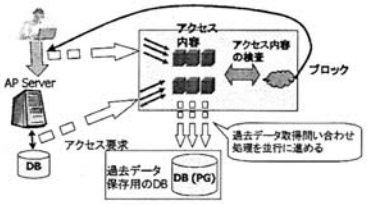

図 4　検査における処理方式内部イメージ

いる．例えば，単純にあるアクセス内容に攻撃コードが含まれるかどうかを判定するような場合は，検査対象とするアクセスフォーマットの定義とその攻撃コードを判定するためのパターン指定を行うことになる．

さらに，業務プロセスにおいて，”申請”，”承認”，”発注”，といった業務を含んでいる場合を考えてみる．この場合，ワークフロー順序を指定することで，検査対象であるアクセス内容を指定ワークフロー順と照らし合わせ，正しいかどうか検査することを考えている．このような検査は，過去に”承認”を意味するアクセスが実行されたかどうかといった過去のアクセス内容を利用して実現される．加えて，ワークフローにおいて，指定部に対して制約条件を指定することで，例えば，過去の見積もり時の金額内容と検収時の金額とが一致するかどうか検査することも可能となる．

最終的に，検査システムの利用に必要なこれらの情報を元に，関連付けキー指定に従い各アクセスの関連付けを行い，ワークフロー順序の指定 (および，制約条件) に合致するかどうかの検査を実施する．加えて，5. 章で述べるアクセス検査に対する高速化手法を適用する．

4. AP/DB アクセスの関連付け

分散したアクセス内容を統合・検査する際，特に，AP と DB に跨るアクセス内容の関連付けは容易ではない．そのため，AP/DB 間でのアクセスを関連付けするために，ある一意なキーを発行し，AP/DB 間で共有／利用するという方法を採用している (詳細 [6])．

5. アクセス検査高速化

5.1 基本方針

本提案方式では，アクセス検査実現のために，メモリ上で保持しきれない過去のアクセスデータを DB を利用して管理する．そのために，検査システム内部で専用の DB (以降過去データ DB と表記) を用意している．合わせて，ボトルネックとなる過去のアクセスデータの参照・書込み (過去データ DB に対する問合せ) を効率化し，高速化を図る．

効率化の基本方針は，問合せの並行実行による効率化を行い，さらに，SQL 実行回数の削減を行う．

5.2 並行問合せ

ボトルネックとなる過去データ DB への問い合わせ処理を並行実行することで，検査の高速化を目指す．実装には，マルチスレッドを利用する．加えて，過去データ DB への過負荷を防ぐため，最大同時接続数を一定数以下に抑えるようスレッドの制御を行う (図 4)．さらに，問合せを並行実行する際に，5. 3. 3 で述べる適切な同期処理を合わせて行う．

5.3 SQL 問合せ回数削減

SQL 実行回数の削減に向けては，

- 問合せ (SELECT) 回数の削減
- アクセス内容データの追記 (INSERT) 回数の削減
- COMMIT 回数削減

を意図して，

- アクセス内容データのバッファリング・キャッシュ
- 不要なアクセスデータの破棄
- COMMIT 実行タイミングの調整

を行う．

5.3.1 アクセスデータのバッファリング・キャッシュ

アクセスが要求された場合，その内容をすぐに過去データ DB に書き込まず，メモリ上でバッファリングしておく．これにより，過去データ DB への書き込みを待たずに，検査を完了することが出来る．加えて，アクセスデータのバッファリングは，キャッシュとしての効果も有する．例えば，直後にその過去データを参照する場合，メモリ上のバッファデータのみ処理すれば良く，過去データ DB へアクセスする必要が無い．これにより，過去データ DB への問合せ回数を削減することが出来，通信回数や負荷を削減することができる．

5.3.2 不要なアクセスデータの破棄

検査によっては，アクセスデータにおける特定の部位に対してのみ検査すれば良いという場合があり得る．また，検査が進むと，アクセス内容のある特定の部位を以後必要としないという場合もあり得る．そのため，検査がどこまで進んだかという進捗状況に応じて，不要なアクセスデータをいち早く破棄し，効率化を図っている．

不要なアクセスデータを破棄するために，検査の進捗状況を内部で状態として管理している．例えば，ワークフローにおいて現在どの業務まで検査済みであるかという状態を管理することになる．さらに，ワークフローの検査のように，過去のアクセスが適切に検査を通ったことが条件となる場合，状態管理を行うことで，それまでの検査結果を状態から知ることが出来るため，再計算を不要に出来る．また，検査がどの時点まで進んだら破棄できるかを判定する方法としては，検査内容の解析や検

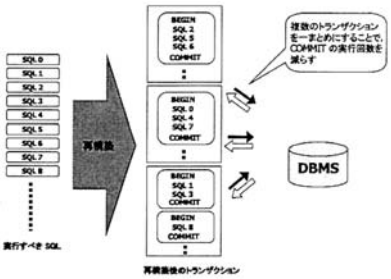

図 5　COMMIT 実行調整のイメージ

査内容の指定時に追加情報としてユーザ側からの指定について検討している．

このようにして，不要なデータを破棄することで，書き込み (INSERT) 回数を減らすことができ，合わせてバッファリングを行うことで，アクセス要求の直後に不要となる場合に限らず，バッファリングされている間に不要となれば，対象データを破棄することができる．

5.3.3　COMMIT 実行タイミングの調整

過去データ DB へ並行にアクセスしているため，頻繁な COMMIT 実行を避けることは容易ではない．これは，Web/AP/DB アクセスを統合して検査に利用する場合，一連の関連するアクセスはほぼ同時に実行されるため，過去データ DB へ並行アクセスすると，COMMIT のタイミングによっては関連付けが出来ず，誤検査が発生してしまうためである．例えば，PostgreSQL などで採用されている多版型同時実行制御においては，あるトランザクション中では，他のトランザクションにより変更された値を参照することができないためにこういった現象が起こり得る．

本手法では，誤検査の発生を避けつつ COMMIT の実行頻度を下げるため，SQL に応じて，実行するトランザクションを動的に構築・管理している (図 5)．基本方針として，更新されたレコードを含め，常に最新の値を参照するため，同一レコードに対する操作 (SQL の実行) を同一トランザクション中に限るように実行している．

単純な例として，あるテーブルのデータに対し，特定のフィールドのみを検索キーとして，SELECT/UPDATE /INSERT を実行する場合を考える．この場合，検索キーの値を管理し，既に実行済みである SQL のキーと今から実行する SQL のキーとを比較することで，同一レコードへのアクセスであるかどうか判定している．加えて，同一レコードに対するアクセスは全て，同一のトランザクション内で実行するように内部で過去データ DB へのコネクションを管理している．特に，過去データ DB へは，アクセス内容を記録したデータが保存されるため，追記・参照のみ実行され，更新されることがなく，そのため，参照時に利用される検索キーの値が変更されるような場合に対処する必要が無い．

一方で，同一レコードに対するアクセスを行う SQL を同一トランザクションとして実行するためには，検索キーがある特定フィールドに限らない場合に課題がある．例えば，ジョインのように複数のテーブルに対してアクセス間合せを行う場合，複数のトランザクションに依存してしまうことがある．このように，実行する SQL が，複数のトランザクションに依存する場合，全コネクションに対して COMMIT を実行させ，それらが完了した後に，その SQL を実行することで，同期化を行っている．これにより，全トランザクションで，それまでの変更結果が参照可能になるため，更新前のデータを利用せずに，最新の値を参照することができる．

6. 評　　価

本提案手法について評価する．評価環境は，Pentium4 Xeon 2.8 GHz x2 (Hyper Threading により x4), 2 Gbytes memory, HDD x1, Red Hat Enterprise Linux 4, PostgreSQL 8.1.4, SUN jdk1.5.0, JDBC3.0 である．

5. 章に述べた方式に基づきプロトタイプ作成した検査システムにより，性能評価を行った (但し検査指定方法およびその解析部などは未実装のため手作業で作成した)．検査内容として，業務フローの順序検査を行うシステムを作成した．具体的には，購入処理システムでの見積，承認，注文，発注，検収，支払い処理という業務フローを想定し，その順序や決済などの権限が守られているかどうか検査を行っている．

評価では，

- **NoCheck**:検査を一切行わずに，業務 AP のみ実行

および，業務 AP に加えて，検査を以下の方法で実行した場合を比較する．

- **Naive**:検査を逐次的に実行し，さらに，過去データ DB に逐一アクセス内容を保存した場合
- **Ours1**:検査を並行に実行
- **Ours2**:検査マシン上でアクセス内容データをバッファリングした場合
- **Ours3**:不要となったアクセス内容を破棄した場合
- **Ours4**:COMMIT 回数の削減手法を適用

加えて，

- **NoFilter**:検査を一切行わないが，ネットワーク構成 (マシン構成) を **Naive**, **Ours** と同様にした場合

業務 AP として，AP/DB 構成を用い，DB の初期

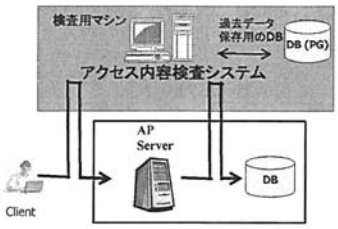

図 6　マシン構成

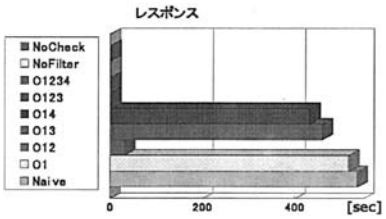

図 7　性能結果 (レスポンス)

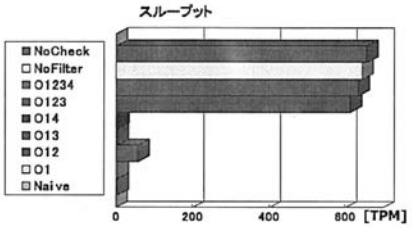

図 8　性能結果 (スループット)

データサイズについては約 900MB とした．また，接続クライアント数を 32 とした．**Naive**, **Ours**, **NoFilter** でのマシン構成は図 6 のようにした．さらに，過去データ DB に PostgreSQL を利用し，初期データサイズとして，こちらも約 600MB に設定した．**NoCheck**, **NoFilter** では，クライアント，AP サーバ，DB サーバのみの構成である．

クライアントマシンから要求があってから，その要求に対する処理を完了するまでの時間をレスポンスとして測定し，さらにスループットを測定した．**Naive**, **Ours** では要求内容の検査を完了した時点で，処理を完了させることになる．**NoCheck** では，検査を一切しないため，単にクライアントが各アクセスを要求してから，処理を完了するまでをレスポンスとして示す．30 分間連続して測定したレスポンスとスループットの平均値を図 7, 8 に示す．

まず，**Naive** に対して，検査や過去データ DB へのアクセスを並行実行 (**Ours1**) し，さらに，バッファリング (**Ours1,2**) を行うことにより，レスポンス，スループット共に 16 倍以上の性能向上を示した．不要となったアクセスデータを破棄 (**Ours1,3**) することで，スループット，レスポンス共に約 20%速度向上した．一方で，不要データの破棄に加えてバッファリングを行った場合 (**Ours1,2,3**)，レスポンス，スループット共に，**Naive** に対してレスポンス約 250 倍，スループットで約 200 倍という大幅な性能向上を達成している．これは，バッファリングしている間にアクセスデータが不要になれば，過去データ DB への保存を行う必要が無くなるという相乗効果を得たためである．また，COMMIT 回数の削減 (**Ours1,4**) により，スループットで 1.6 倍，レスポンスで 1.2 倍の速度向上を示した．最後に，提案手法を全て適用した **Ours1,2,3,4** では，**Naive** と比較した場合，レスポンス約 265 倍，スループットが約 200 倍という大幅に性能向上を達成している．

本手法により，**NoCheck** に対するオーバヘッドを，スループットで 3.5%, レスポンスで 5%まで抑えることに成功した．

7. ま と め

本論文では，内部統制支援を目的とした Web/DB/AP サーバに跨るアクセスや過去に行われたアクセスの統合・検査における高速化方式を述べた．提案方式では，過去のアクセス内容を保持する DB に対して，(1) 並列アクセスによる遅延隠蔽，(2) バッファ・キャッシュによる参照回数削減，(3) 不要な過去データの破棄による効率化，(4) 同期化 (COMMIT) 回数の削減を行うことで，レスポンス，スループットを大幅に改善し，5% 以下の実行時オーバヘッドで検査を実現できた．

本提案手法は，内部統制支援の事例におけるアクセス内容の検査技術に限らず，分散処理アクセスの監視全般に広く利用できると考える．

文　献

[1] トレッドウェイ委員会組織委員会: 内部統制の統合的枠組み (2006).
[2] 金融庁企業会計審議会内部統制部会: 財務報告に係る内部統制の評価及び監査の基準並びに財務報告に係る内部統制の評価及び監査に関する実施基準の設定について (意見書) (2007).
[3] 間形文彦, 濱田貴広, 岡崎聖人, 塩野入理, 金井敦: DBMS における業務処理統制機能の要件と課題に関する考察, 第 35 回電子化知的財産・社会基盤研究会 (2007).
[4] 新日本監査法人: 東京証券取引所の宣誓書と確認書への対応 (2007).
[5] データベース・セキュリティ・コンソーシアム (DBSC) http://www.db-security.org/: データベースセキュリティガイドライン (1.0 版) (2006).
[6] 山中真和, 榎本俊文, 谷口展郎, 山室雅司: ログ検査を目的とした AP/DB ログ関連付け機能の実装と評価, 第 36 回 電子化知的財産・社会基盤研究会 (2007).