3層Webクライアント/サーバシステムでのセキュリティ監査ログの実装

大森 泰弘　大塚 知彦　塩谷 理恵

An Implementation for Security Audit Log in 3-Tier Web Client-Server System

Yasuhiro Ohmori　Tomohiko Ohtsuka　Rie Shionoya

個人情報保護法の施行や日本版SOX法への対応に伴い，データアクセスの監査証跡としてのセキュリティ監査ログの実装は企業システムの重要な課題である．特にオンラインシステムなどで一般的に採用されている3層Webクライアント/サーバシステムでのセキュリティ監査ログ実装に当たっては，ユーザーの特定ができない点や，それを解決しても監査ログ収集によるCPUやディスクなどのシステム資源への負荷が大きくなるなどの問題がある．本論文では，解決策として，DB2® UDB V8から追加された「拡張クライアント情報」と呼ばれる特殊レジスターに監査ログに使われるデータ項目（「監査ログ項目」）をセットすることで，プラットフォームを問わずDB2 UDBに対して監査ログ項目を送れることを確認した．また，監査ログの収集，管理という次の課題に対しては，プラットフォーム間で比較した結果，DB2 for z/OS®で一般的に推奨されている会計情報のトレースを，監査ログとして流用する仕組みが最も有効であるという結果を得た．

With the enforcement of the Act on the Protection of Personal Information and the forthcoming the Japanese version of the Sarbanes-Oxley law(J-SOX), requirement against the security for the enterprise system is growing. Among several security requirements, security audit logs are one of most important functions to be gathered for audit trails of data access to fill the security requisites. To implement the online 3-Tier Web Client-Server System using Java based application servers, dealing with product dependent concerns are biggest challenge, which are a default user ID and password and additional CPUs and I/Os to implement security mechanism within a system. In this paper, we address security audit log implementation method to gather and manage security audit log as a part of DB2® UDB function, by putting audit items to new special register, extended client information, which was introduced in DB2 UDB V8. Focusing on zSeries® platform, which by far best solution of all, we address consideration on implementing audit logs along with other platforms.

Key Words & Phrases：セキュリティ, WAS, DB2, 監査ログ, 特殊レジスター, DB2PM, Java
Security, WAS, DB2, audit log, special register, DB2PM, Java

1. はじめに

個人情報保護法の施行や日本版SOX法への対応に伴い，システムに対するセキュリティ機能の要求は高まっており国家レベルの取組になってきている［1］．その中でRDBMS製品へのアクセス履歴としてのセキュリティ監査ログ（以下「監査ログ」と言う）として収集，保存する機能は，セキュリティ要件として重要性が増している［2］．一般的にオンラインシステムなどで広く採用されている，クライアント（ブラウザ），アプリケーション・サーバ，DBサーバからなる3層Webクライアント/サーバシステム［3］での監査ログ実装には様々な課題がある．特に不正が発覚した際の分析に必要な監査証跡としての監査ログの場合，従来のシステムではアプリケーションの識別にDBサーバへの接続に使用される接続ユーザーIDやアプリケーションのプロセス名などが使用できた．しかし，現在広く使われている Java™ベースのアプリケーション・サーバを使用した場合，RDBMS製品にアプリケーション・サーバの接続ユーザーIDで接続することやJava実装上の問題によって，エンドユーザーやアプリケーションを識別することができないという典型的な問題があるが，DB2® UDBやOracle DBといった製品に備わっている監査ログ収集機能だけでは解決できない．こ

提出日：2005年08月30日　再提出日：2006年3月23日

の問題の対応として, クライアントの情報をDBサーバ側に渡す機能がそれぞれの製品で提供されているが, 依然として監査ログとして実装するためには, セキュリティ製品やアプリケーションでの実装で実現している監査ログの収集にかかるDBサーバに対する負荷の増加や, 大量に書き出されたログの管理・分析などの課題が残る. この課題を, 製品機能に限らずデータ項目の定義, アプリケーション設計, 運用設計など必要な範囲をカバーすることで解決を図ったので, 本論文で報告する. また, 既存システムに監査ログを追加で実装する場合, ツールやログ収集によって受けるシステム資源への影響にも十分な注意が必要である. これに対しDBサーバに渡す項目として, 監査ログで使用されるデータ項目(以下「監査ログ項目」と言う)をセットし, 収集することで解決を図り検証した.

本論文では3層Webクライアント/サーバシステムで一般的に使用されているJavaに特化し, オンラインシステムでの監査証跡としての監査ログ要件について述べる. 実装には, DB2 UDB固有の機能でアプリケーションが使用する定義域であるDB2 UDB特殊レジスター(以下「特殊レジスター」と言う)[5,6]にDB2 UDB V8から新しく追加された「拡張クライアント情報」と呼ばれる4つの項目を利用した. この監査ログの実装方法をz/OS®やLinux, Unix®, Windows®(以下「LUW」と言う)の各プラットフォームで検証し, その実現方法や考慮点について述べる. また, 最も有効なDB2 UDB for z/OSのケースを中心に, 各プラットフォームや運用に関する考慮点について整理し比較評価する. なお, 検証にはアプリケーション・サーバとしてWebSphere® Application Server(以下「WAS」と言う)を使用した.

System i5™については, 拡張クライアント情報が現在サポートされていないため, 今回は比較検証の対象外とした.

以下, 2章では本論文で扱う「セキュリティ監査ログ」について明確にし, 監査ログ実装方法の検討および検証項目を定義する. 3章ではWASとDB2 UDBを使用した監査ログ実装の仕組みを紹介し, 4, 5章でz/OS, LUWでの監査ログ実装について述べる. 6章で各プラットフォームでの実装について比較評価し, 最後に結論および今後の展開・展望を述べる.

2. セキュリティ監査ログ実装方法の検討

2.1 セキュリティ監査ログの対象

セキュリティ監査ログは, セキュリティ要件に対応する目的によって意味が異なる. 本論文では一般に「DB監査ログ」[4]と呼ばれる, 「誰がどのデータに対して何をしたか?」を知る目的で収集するログのうち, 以下に該当する監査証跡のために収集するものと定義する.

- Javaベースのアプリケーション・サーバからのDBアクセス
- 正常に認証が行われ, 正常終了した処理

即時対応が必要となる不正アクセスなどのログはオンライン監視として実装されるが, 今回はオンライン監視のできない正常終了時のアクセスを対象とした. また既存システムへの適用を考慮し, ログはDBサーバ側で収集, 一元管理して非同期に分析することを前提とした. ただし本論文での実装方法では, 監査ログ項目をリアルタイムにモニターすることが可能であることを検証している.

2.2 監査ログ実装上の課題

従来DBサーバではパフォーマンス管理やセキュリティ管理上, アプリケーション識別に接続ユーザーIDやプロセス名などを使用してきたが, これらは主にローカルアプリケーションの識別に使用されてきた. Javaアプリケーション・サーバを使用して3層クライアント/サーバシステムを実装した場合, Java Virtual Machine上でバイトコードと呼ばれる中間コードが実行されるというJavaの実装上, DBサーバからはプロセス名がすべて同じに見え, アプリケーションを識別できないという問題がある. また, アプリケーションおよびDBサーバでの負荷軽減を目的とした接続プール機能はアプリケーション・サーバの接続ユーザーIDを使用するため, アプリケーションの識別が困難である.

さらに監査ログ実装上問題となるのが, システムに追加でかかる負荷である. 現在実装されている手法を例にとると, 各RDBMS製品に付随する監査ログ機能を使用する方法, アプリケーションで実装する方法, セキュリティ製品を使用する方法などが挙げられる[3]. RDBMS製品の監査ログ機能は, ログ量と比例して監査ログの書き出し量が増えるため, アクセス増加に伴うDBサーバのパフォーマンス劣化という問題がある. アプリケーションでの実装を行う場合, DBサーバに対する負荷がない分, アプリケーションの開発工数追加やアプリケーションごとに書き出された監査ログをどのようにマージするかなどの課題が残る. このような問題に対応できるものとして, ネットワーク上のすべての情報を取得するパケット・キャプチャ型のセキュリティ製品の採用も考えられるが, セキュリティ要件が高くなり暗号化されたときの対応やパケットからキャプチャした膨大なログからどのように分析に使用していくかが課題となる. それぞれの手法に一長一短があることが監査ログ実装において決定的な実装方法がない理由の一つである.

2.3 監査ログ実装方法および検証項目

クライアントの情報をDBサーバへ送ると言うことは, RDBMS製品としてJavaベースのアプリケーション・サーバ環境でトランザクションごとにSQL処理を識別することを目的としている. よってこの機能を利用して監査ログ実装上の課題の一つであるトランザクション単位のSQL処理とユーザーの識別が可能となり, 本論文のおけるセキュリティ要件に対応する監査ログ実装が可能と言える.

本論文では, DBサーバでの一元管理が最適なソリューションと判断し, 監査ログをDBサーバ側で収集する方法について検証を行った. この時の課題は, 監査ログの収集に起因するDBサーバの負荷を抑えながら十分な情報を収集することである.

以上のことから, 当監査ログ実装に当たっては, 監査ログ運用上の要件である「監査ログのDBサーバ側での収集」と「システム資源への影響の最小化」の観点からプラットフォーム別に検証を行った.

3. WAS-DB2 UDB環境における監査ログ実装

本論文では, WAS-DB2 UDB環境における監査ログ実装(**図1**参照)を以下の3つの設計により実現する.

(1) 特殊レジスターに監査ログ項目をアプリケーションでセットする設計
(2) 特殊レジスターにマッピングさせる監査ログ項目の設計
(3) DB2 UDB側での監査ログ収集・分析する運用設計

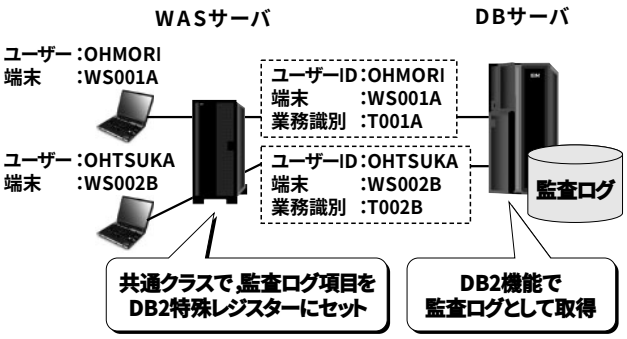

図1. 監査ログ実装の仕組み

(1), (2)部分はアプリケーションで実装する部分であり, プラットフォーム間で共通である. よってプラットフォームごとの検証は(3)について実施する.

3.1 DB2 UDB特殊レジスターの概要

DB2 UDBの固有機能である特殊レジスターは, アプリケーション・プロセスに対して定義する定義域であり, SQLステートメントで参照可能な情報を保管するために使用される. 拡張クライアント情報とは, z/OSやLUWプラットフォームのDB2 UDB V8で新たに追加された, エンドユーザーやアプリケーションの識別を目的とした, 4種類の特殊レジスターの項目(**表1**参照)の総称である.

表1. 拡張クライアント情報の一覧

特殊レジスター	属性	最大長(バイト)
CLIENT_USERID	VARCHAR	255
CLIENT_APPLNAME	VARCHAR	255
CLIENT_WRKSTNAME	VARCHAR	255
CLIENT_ACCTNG	VARCHAR	255

3.2 監査ログ項目の特殊レジスターへのセット

Javaプログラムから監査ログ項目を拡張クライアント情報にセットするには, DB2 Universal JDBC DriverのDB2Connection Classで提供されるMethodを使用する. つまりJavaアプリケーションでは, DB2 Connectionオブジェクトを使用する. なおWAS V6からWSConnection classによるセットがサポートされ, セット方法が容易になった. **図2**は監査ログ項目を以下のようにセットするコーディング例である(JDBCおよびSQLJのいずれでも使用可能)[5].

USERID: OHMORI
WRKSTNAME: WS001A
APPLNAME: T001A

```
WSConnection conn = (WSConnection) ds.getConnection();
Properties props = new properties();
props.setProperty(WSConnection.CLIENT_ID, "OHMORI");
props.setProperty(WSConnection.CLIENT_LOCATION , "WS001A");
props.setProperty(WSConnection.CLIENT_APPLICATION_NAME,
"T001A");
...
conn.setClientInformation(props);
```

図2. 拡張クライアント情報への監査ログ項目セット例

拡張クライアント情報には初期値としてBlankがセットされているが, 各製品で初期値がセットされる仕組みが提供されており, 特に今回使用したWASからの接続に関しては, アプリケーションとDBサーバのインターフェースであるDatasourceごとに定義された値がセットされる.

当監査ログ実装方法では, この事前にセットされた拡張クライアント情報をトランザクションごとに上書きすることが前提となる.

なおWASではDatasourceごとにカスタム・プロパティで拡張クライアント情報に初期値を設定することで, 拡張クライアント情報をすぐに実装する必要がない既存アプリケーションでも, 複数のWASサーバのうち

問題となったアプリケーションがどのWASサーバで稼動していたか識別することが可能である.

図3では, 拡張クライアント情報が実際にセットされるタイミングが描かれている. トランザクション内のどのタイミングでもセットすることは可能であるが, DBサーバへ送信されるのは後続のSQL実行時である. よって拡張クライアント情報のセットによるWASとDB2 UDB間の通信のオーバーヘッドはほとんど発生しない.

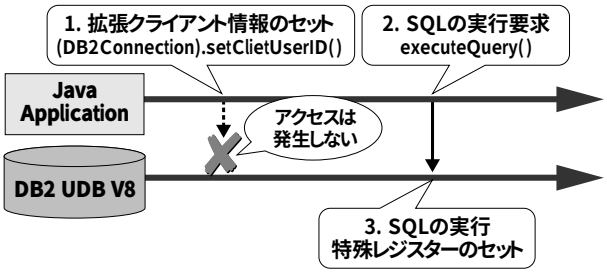

図3. 特殊レジスターのセット処理の流れ

3.3 監査ログ項目のマッピング

監査ログ項目を新規に定義する場合は, **表2**に示すデータ項目を適用する. 既存システムの場合にはセキュリティ制限をかける目的で, 既に定義, 使用されているユーザーID, 業務ID／帳票ID, 端末IDなどのデータ項目をDB2 UDBの拡張クライアント情報の項目に対応させる設計を採用することで監査ログ実装を容易にすることができる.

なお, 監査ログのタイムスタンプは, アプリケーションが特殊レジスターにセットするのではなく, SQLを実行した際にDBサーバ側でセットされるタイムスタンプである.

また, **表2**の監査ログ項目案は拡張クライアント情報を受け取るDBサーバの制約があるため, 可搬性を考え最も制約の厳しいDB2 UDB for z/OSに合わせた長さを採用した.

表2. 監査ログ項目案

特殊レジスターでの識別	属性	意味づけ
TIMESTAMP*注1	Timestamp	SQLの実行Timestamp
USERID	16文字以内	ユーザー識別
WRKSTNAME	18文字以内	端末識別(場所)
APPLNAME	32文字以内	業務, 帳票識別
ACCTNG	255文字以内	原則, 使用しない

注1:TIMESTAMPは特殊レジスターではない

監査ログ項目について定義する際は, 以下の点を考慮して設計を行う.

- **TIMESTAMP**
  SQL実行時のタイムスタンプ.
- **USERID**
  セキュリティ制御に使用されるユーザー識別に使われているデータ項目.
- **WRKSTNAME**
  Webアプリケーションで使用しているPCやロケーションを識別するデータ項目. 既存システムで該当するデータ項目がない場合, 他の追加情報として利用することも可能.
- **APPLNAME**
  業務識別のために, セキュリティ制御の単位になっている業務識別項目をマッピングする. 実行するSQLを識別するIDやレポート番号などが候補となるが, セキュリティ要件が厳しい場合は, もっと細かく制御できるようにコード設計をする.
- **ACCTNG**
  ACCTNGには, 監査ログ項目の3項目以外の情報をセットすることが可能である. ただし, この項目を使用して追加情報をセットする場合は, ネットワーク負荷やログなどのシステム・リソースを考慮し, 原則使用しない.

データ項目に関しては, その妥当性を検証するために, 金融業界でセキュリティに関するガイドラインで示されている「認可されているアクセスについて」の監視項目の5項目[7]に関してカバーされていることを確認した.

3.4 DB2 UDB側での監査ログ収集

当監査ログ実装方法の検証に当たって, プラットフォームの違いが発生するのはDB2 UDBの機能を利用して監査ログを収集する部分である. z/OSと, LUWのそれぞれのケースで監査ログ収集の方法について以降の章で述べる.

4. DB2 UDB for z/OSでの監査ログ収集

4.1 会計トレースを利用した監査ログの収集

DB2 UDB for z/OSで拡張クライアント情報は, 様々な種類のトレースに出力されるが, CLIENT_ACCTNGを除き, アプリケーションに関する情報が収集する会計トレースのヘッダー情報として格納される点に注目した. このことからDB2 UDB for z/OSでは監査ログの収集に当たって当監査ログ実装方法を採用した場合, 監査ログを運用上の変更なしに会計トレースのデータとしてDBサーバ側で収集および保管することが可能といえる.

その理由はDB2 UDB for z/OSでは従来からトレース機能を利用しパフォーマンス・チューニングや問題発生時の問題判別に使用するために, 本番環境であっても最低限のトレースを収集する運用が推奨されているからである. この標準で収集すべきトレースは, 会計トレースおよびDB2 UDB for z/OSの使用状況を

収集する統計トレースである[8]. しかもトレースデータはz/OSの機能の一部であるログを含むシステムの情報を収集する機能であるSystem Management Facility(以下「SMF」と言う)を使用してSMFデータセットに蓄積されるので, この蓄積および利用については, 現行の運用をそのまま維持することが可能である(**図4**参照).

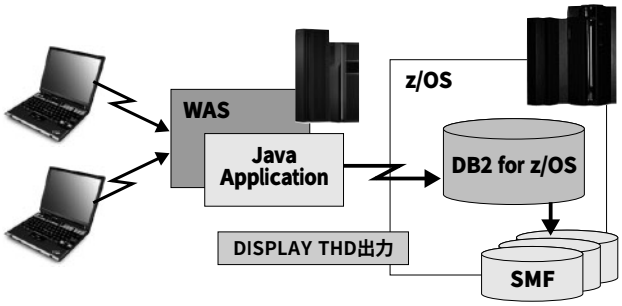

図4. DB2 UDB for z/OSを使用した場合の構成例

また会計トレースではSQLの開始と終了のタイムスタンプが収集されるため, 監査ログのタイムスタンプ項目としても意識することなく値が得られるという効果がある.

なおWAS-DB2 UDB for z/OS環境で監査ログを実装し監査ログ項目を収集するには, DB2 Universal JDBC Driverの使用し, DB2 UDB for z/OSのスレッド・プーリング機能, および会計トレースの集積機能を使用しないことなどが前提条件となる.

またDB2 UDB for z/OSでは, リアルタイムにセットされている情報を出力する方法として-DISPLAY THREAD(*)コマンドがあり, DSNV437Iメッセージにその情報が出力される(**図5**参照).

```
-DB8G DIS THD(*) TYPE(INACTIVE)
DSNV401I  -DB8G DISPLAY THREAD REPORT FOLLOWS -
DSNV424I  -DB8G INACTIVE THREADS - 861
NAME     ST A   REQ ID          AUTHID   PLAN··
SERVER   R2      0 java.exe     ADCDMST  DISTSERV
 V437-WORKSTATION=WS001A, USERID=OHMORI,
        APPLICATION NAME=T001A
```

図5. DB2 for z/OSにおけるコマンド出力例

4.2 監査ログの利用方法

DB2 UDB for z/OSで収集された会計トレースを監査ログとして保管し, セキュリティ監査業務として使用するためには, 監査ログ項目として管理, 利用する手段が必要になる. z/OSの場合, DB2 UDB for z/OSのトレースデータのレポートする手段として, 例えばTivoli OMEGAMON XE for DB2 Performance Expert/DB2 Performance Monitor V3.1(以下「DB2PM」と言う)のようなツールが存在する. 現在最新のDB2PMを使用してDB2PMレポートおよびDB2PM Database(以下, DB2PM DBと呼ぶ)に, 拡張クライアント情報が表示されることを確認した(巻末の**図9**参照). これにより, 新規業務開発をすることなく, 既存のDB2PMレポートを監査業務に使用することが可能であるといえる.

なお, DB2PM DBとはDB2PMのオプション機能であり, トレースデータを表にLOADし蓄積することを可能にするものである. これによってSQLを使用して必要な項目のみ検索・絞り込み・表示することが可能である. 加えてDB2PMでは, 拡張クライアント情報はそれぞれ別の名前にマッピングされていることを確認した(**表3**参照).

表3. DB2PMにおける拡張クライアント情報

特殊レジスター	DB2PMレポート	DB2PM DB列名
CLIENT_USERID	ENDUSER	CLIENT_ENDUSER
CLIENT_APPLNAME	TRANSACT	CLIENT_TRANSACTION
CLIENT_WRKSTNAME	WSNAME	CLIENT_WSNAME
CLIENT_ACCTNG	ACCOUNTING STRING	RQ_STRING

また, CLIENT_ACCTNGをDB2PMレポートで出力するには, システム負荷のかかるRECORDトレースを収集する必要があることを確認した. この点からも, 拡張クライアント情報のACCTGを監査ログ項目に原則使用しないものとする(**表2**参照).

4.3 z/OSでの検証

DB2 UDB for z/OSの場合, ログの収集方法として会計トレースを使用することができることから当監査ログ実装方法は監査ログ運用上の要件を満たす.

監査ログ項目をDBサーバ側で収集する仕組みとして, 既存の会計トレースの仕組みと運用をそのまま流用することができる点と, 既に本番システムで会計トレースが収集中である場合, システム資源に対する負荷の増加を最小化することが可能である.

5. DB2 UDB for LUWでの監査ログ収集

5.1 LUWでの監査ログ収集

今回の検証の結果, DB2 UDB for LUWの現在のバージョン(8.2)では, 拡張クライアント情報をトレースに出力する標準機能が提供されていないことを確認した. 監査ログに該当するような, 常時トレースを書き出す機能としてdb2audit機能があるが, 拡張クライアント情報は収集されないため, 監査ログ収集のためにはアプリケーションで監査ログ項目出力の仕組みを別途開発する必要がある(**図6**参照).

監査ログに該当する情報の出力先としては, ファイルまたは表が考えられるが, システム資源や利用

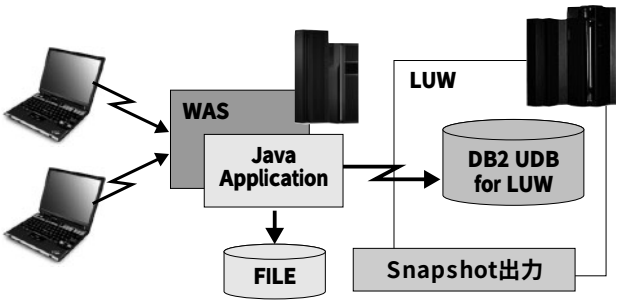

図6.　DB2 UDB for LUWの場合の構成例

の観点で一長一短があり, 選定の判断は難しい.

表の場合は, 軽いSELECT処理に対しても毎回INSERT処理が発生することに起因するCPU負荷の増加と, それに伴うDB2 UDBのログ増加が短所として挙げられる. ファイルの場合, 複数のWASサーバから監査ログ・ファイルを一箇所に収集するための仕組みが必要になる.

なおセットした拡張クライアント情報をDB2 UDB for LUWの機能で収集する方法として, スナップショット・コマンドがある. ただし, スナップショット・コマンドの出力情報はコマンド入力時点の情報であり, 監査ログとしては使用することができない. しかしリアルタイムで「誰が何をしているか」を監視する機能として, DB2 UDB for LUWのケースでも拡張クライアント情報を利用する仕組みは有用である.

DB2 UDB for LUWでは, アプリケーション・スナップショット・モニターで拡張クライアント情報を収集できる. **巻末の図8**はスナップショット表関数を利用した出力例である.

また参考として実行したSQLステートメントを**図7**に示す.

```
SELECT
 CASE WHEN TPMON_CLIENT_USERID IS NULL
     THEN SUBSTR(AUTH_ID,1,18)
     ELSE SUBSTR(TPMON_CLIENT_USERID,1,18)
 END AS CLIENT_USERID,
 CASE WHEN TPMON_CLIENT_APP IS NULL
     THEN SUBSTR(APPL_NAME,1,18)
     ELSE SUBSTR(TPMON_CLIENT_APP,1,18)
 END AS CLIENT_APPNAME,
 SUBSTR(CHAR(AGENT_ID),1,4) AS AGENT_ID,
 CASE WHEN TPMON_CLIENT_WKSTN IS NULL
     THEN SUBSTR(CLIENT_NNAME,1,18)
     ELSE SUBSTR(TPMON_CLIENT_WKSTN,1,18)
 END AS CLIENT_WKSTN,
 SUBSTR(APPL_ID,1,26) AS APPL_ID,
 SUBSTR(DB_NAME,1,8) AS DB_NAME
FROM TABLE(SNAPSHOT_APPL_INFO('SAMPLE', -1)) AS
SNAPSHOT_APPL;
```

図7.　スナップショット表関数の使用例

5.2　LUWでの検証

監査ログ項目をDBサーバ側で収集するためのトレース機能は現時点で提供されていない. 監査ログ項目をユーザーアプリケーションでファイルや表に書き出す仕組みを開発する必要がある点で, 当監査ログ実装方法では対応が不十分である.
また監査ログを表またはファイルに書き出す場合, システム資源への影響は非常に大きいことが予想される[4].

6. 検証結果の比較評価

z/OSとLUWのプラットフォーム間で問う監査ログ実装方法を比較検証した結果, **表4**にあるように基本機能面でz/OSの場合が最適であると結論づけることができた.

表4.　プラットフォーム間の提供機能の違い

	監査ログ収集	監査業務	システム資源への影響
z/OS	○会計トレース	○DB2PM	最小化可能
LUW	△要開発	△要開発	ー

今回の実装方法の検証に当たって, z/OS, LUWの両方で拡張クライアント情報にセットした値をリアルタイムにコマンドで表示できることが確認された.

これによって, WAS-DB2オンライン業務のような3層Webクライアント/サーバ構成であっても, 当実装に準拠し, 拡張クライアント情報に該当する情報をセットしておけば, **表2**にあるような, ユーザーや業務の識別が可能という結果が得られた.

7. おわりに

本論文では, 3層Webクライアント/サーバ環境における監査ログの実装上の課題として, アプリケーションや利用ユーザーの識別の実現だけでなく, ログ収集におけるシステム負荷軽減, 監査ログとしての利用手段といった点に着目した. この課題の解決策として, システム設計として, データ項目としての監査ログ項目, DBサーバへの受け渡しのためのアプリケーション設計, 収集の運用設計をそれぞれ述べその効果を検証した. その結果, 当監査ログ実装に当たって監査ログ運用上の要件としてあげた検証項目の2点である「監査ログのDBサーバ側での収集」と「システム資源への影響の最小化」を満たしたDB2 UDB for z/OSが最も有効であった.

監査ログ実装の監査ログ項目を拡張クライアント情報にアプリケーションでセットする部分は, すべてのSQL実行前に漏れなく実施する必要がある. よって, 実際のプロジェクトでは, Java共通クラスで拡張クライアント情報をセットするフレームワークを当初か

ら採用することが望ましい.

また, 採用したDB2 UDBの新機能は, 3層Webクライアント/サーバシステムにおける問題判別やパフォーマンス・チューニングでの使用を主目的としているので, フレームワークでの吸収は, 運用管理の面からも価値が高く, WAS-DB2 UDBオンライン開発標準として広めたいと考える.

参考文献

[1] 情報セキュリティ対策推進会議決定, 情報セキュリティポリシーに関するガイドライン, http://www.kantei.go.jp/jp/it/security/taisaku/guideline.html, 平成12年7月
[2] 石垣 良信, ITセキュリティー・システム構築へのアプローチ, ProVISION, No. 42, pp.10-15, 2004
[3] 小泉修, 図解でわかるWeb技術のすべて, 日本実業出版社, ISBN4-534-03202-1, 2002
[4] 吉田 晃, 情報漏洩を防ぐDBセキュリティ, 日経システム構築, 日経BP社, 2005年9月号, pp.24-40, 2005年8月
[5] Bart Steegmans et al., DB2 for z/OS and Websphere: The Perfect Couple, IBM Corporation, http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246319.html?Open, 2005
[6] Namik Hrle and Johannes Schuetzner, Finding Out Who Did It, IDUG Solutions Journal, Vol. 11, No. 2, 2004
[7] FISC(The Center for Financial Industry Information Systems), セキュリティガイドラインIII, 情報セキュリティ管理標準, 7.7.2, 2005
[8] Daniel L. Luksetich, DB2 for z/OS Application-Level Performance Reporting and Tuning, z/Journal, Dec. 2004/Jan. 2005, http://www.zjournal.com/Article.asp?ArticleID=982&DepartmentID=32, 2004

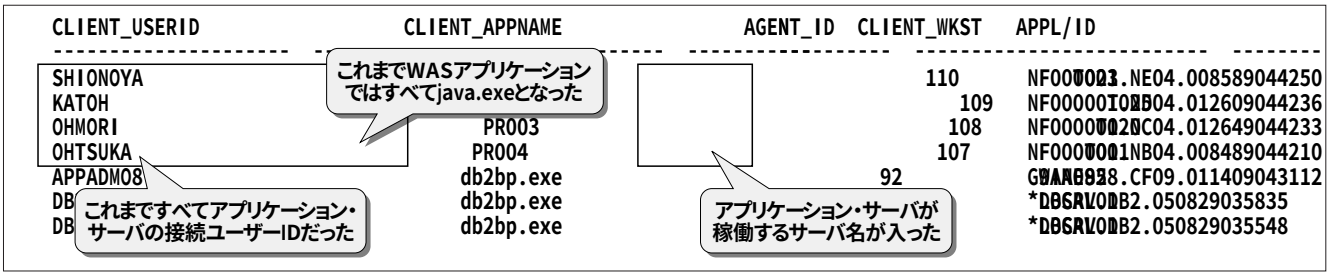

図8. スナップショット表関数を利用した接続アプリケーション・リストの出力例

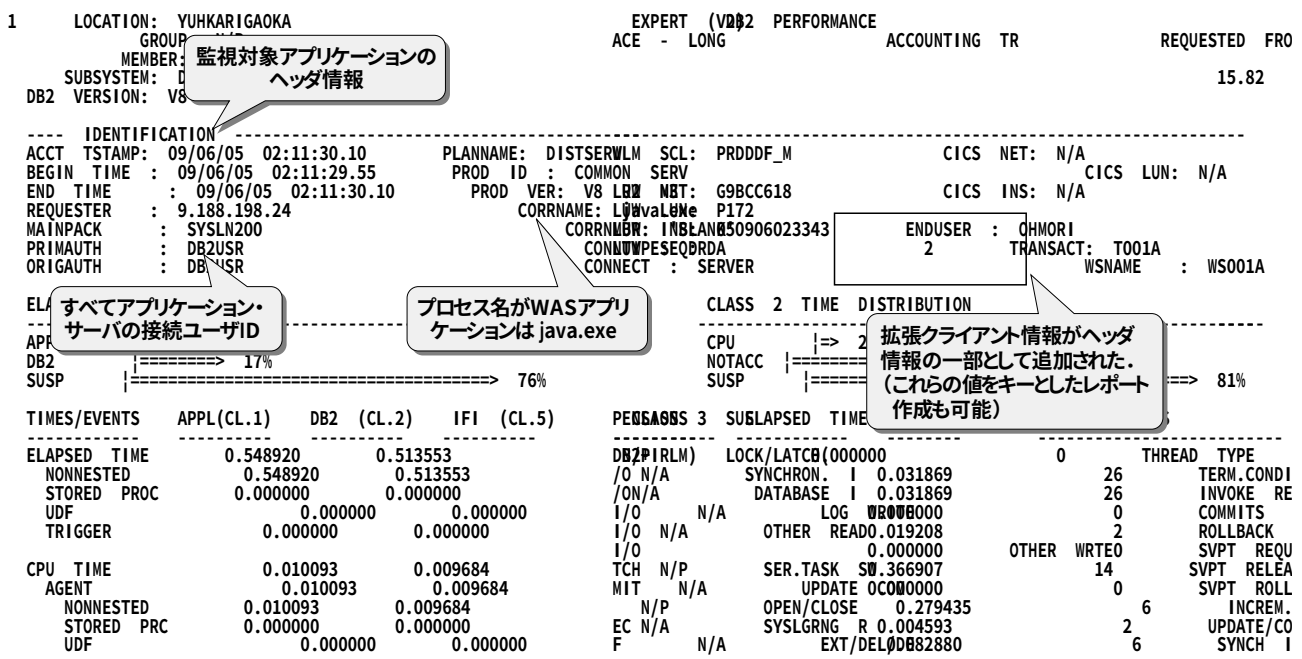

図9. DB2 for z/OS環境におけるDB2PMの出力例

日本アイ・ビー・エム
システムズ・エンジニアリング株式会社
エンタープライズ・ミドルウェア
ITスペシャリスト

大森 泰弘 Yasuhiro Ohmori

［プロフィール］
2001年，日本アイ・ビー・エム システムズ・エンジニアリング入社．DB2 UDB for z/OSの製品やそのDRDA構成について技術支援を担当し，お客様のプロジェクト支援，技術資料の執筆，研修を実施．
ohmoriy@jp.ibm.com

日本アイ・ビー・エム
システムズ・エンジニアリング株式会社
インフラストラクチャー・デザイン担当
ICPシニアITアーキテクト

大塚 知彦 Tomohiko Ohtsuka

［プロフィール］
1986年日本IBMに入社．九州地区で流通小売業（岩田屋様）のお客様担当SEや九州地区の複数プロジェクトにDB2スペシャリストとして従事．2002年からISEに異動し現在はICPシニアITアーキテクトとして，企業のデータモデルの視点から，経営やシステムの最適化を図る活動に取り組んでいる．
TOMOHIKO@jp.ibm.com

日本アイ・ビー・エム
システムズ・エンジニアリング株式会社
インフォメーション・マネジメント
ITスペシャリスト

塩谷 理恵 Rie Shionoya

［プロフィール］
1996年日本IBM入社．i5プラットフォームをベースにしたお客様への技術支援，プロジェクトに参画後，2001年ISEに出向．ISEではDB2 UDBの技術サポートを担当．主にレプリケーション，Information Integratorなどデータベース連携に関するプロジェクトの技術支援や，研修を実施．
e27532@jp.ibm.com