Windowsを前提とした商品先物業界向け基幹業務パッケージ開発

Case Study of Implementation of Mission-critical Application on the Windows Platform for Future Commodity Industy

伊藤　公一郎

要　約　アウトソーシングを前提とした商品先物取引基幹業務パッケージ・システム（以降COMTRADEと略す）は，2002年6月に第一号ユーザの本番を迎え，2002年12月現在，4社で本番稼働している．

COMTRADEは，ES 7000およびWindows 2000 Datacenter Server（以降DCSと略す）上で稼働するWebアプリケーションとして開発された．保守性，障害時の対応を考えソフトウェアを可能な限りMicrosoft社製品で統一し，AP環境の確認手段として稼働ログ，エラーログを設けた．基幹業務のアウトソーシング展開を想定し，1台のES 7000をパーティション分割し，各サーバ層を二重化した．また，LUCINAの提唱する3層構造を利用し，Webサーバは Internet Information Service，APサーバにCOM＋コンポーネント・サービス，DBサーバにはSQL Server 2000を採用した．

パッケージ開発にあたっては，『基幹業務系をWindowsでどう実装するか』という点と『アウトソーシングを想定した複数社環境をどう保持・運用できる形態にするか』というのが大きなテーマとなった．

本稿ではWindowsプラットフォーム上で基幹業務系アプリケーションを構築した事例について報告を行う．

Abstract　COMTRADE, which is a mission critical commodity futures trading package system based on outsourcing, started its first production run in June of 2002. As of December 2002, the system is in production for four user companies.

COMTRADE is a Web enabled application which runs on the ES 7000 and Windows 2000 Datacenter Server platform. Considering for maintainability and counter-measures in case of system failures, we used Microsoft products as much as possible. we added the operation logging and error logging functions for confirming the application operation status.

To enable this mission-critical application for use in outsourcing to multiple customers, we partitioned the ES 7000 system and duplicated each server layers. We adopted the three-tiered architecture design proposed by LUCINA. Internet Information Service is used for the Web server, COM+ component service for the application server, and SQL Server 2000 for the database server respectively.

There existed two major issues in the development of this package "How do we implement a mission critical application on Windows?" and "How can we create a design for managing and operating a multi-customer environment?"

This paper discusses an example of the implementation of a mission critical application on the Windows platform.

1. はじめに

商品先物取引所の取引員各社は，業界を取り巻く変化（手数料自由化対応，取引所システムの更改対応，制度改正等）に対して，より柔軟に，より低コストで対応できる情報システムを

切望していた．

こうしたなか，汎用機に匹敵する信頼性，オープン系の拡張性・可用性を兼ね備えた ES 7000 と Windows 2000 Datacenter Server（Windows 2000 DCS）が発表された．これを契機として，最新技術を取り入れた商品取引の基幹業務システムを ES 7000 上で構築し，大幅なコストダウンを実現するため，パッケージ・システム（以降 COMTRADE と省略）をアウトソーシング事業として提供・展開することとなった．

また更なるコストの削減策として，「1 台の ES 7000 を複数パーティションに分割し，複数社で使用する」サービス方式を採用し，低価格のサービスを実現した．

2000 年 6 月に COMTRADE の開発に着手した当時は ES 7000 上で実現する初の金融基幹業務システムであったため，参考にできる過去事例やミドルソフトがほとんど無く，Web アプリケーションということもあり，金融業務で求められるパフォーマンスや信頼性，可用性がどこまで実現できるか，課題を抱えた形での開発となった．それらの問題を乗り越え，2002 年 6 月に第一号ユーザの本番を迎え，現在 4 社が稼働している．

本稿では，COMTRADE の開発事例を通して，ES 7000 上で Windows ベースの基幹系 Web アプリケーションを構築する際に考慮した点について述べる．

2. システムの概要

2.1 業 務 機 能

本システムは，商品先物取引所の取引員各社で利用され，一般委託者（個人投資家）および会員企業からの売買受託取引（委託取引）と自社のディーラが行う自己取引を取り扱う基幹業務システムである．主な機能は次のとおりである．

1） 注 文 管 理

注文の入力，取消，照会を行う機能．取引員企業は注文をクライアント PC より入力し，アウトソーシング・センタへ送信する．アウトソーシング・センタでは注文の受付を記録し，受付結果をクライアント PC へ返信すると共に，東京工業品取引所へ発注する．

2） 約 定 管 理

注文の約定情報を登録，取消，照会する機能．システム間のインタフェースを持つ東京工業品取引所からは約定電文を受信し，自動的に約定情報を登録する．他取引所からは約定の連絡を受け，約定情報を画面より手入力する．当日だけでなく過日の取扱いも可能．

3） 建 玉 管 理

建玉情報を管理し，照会画面より委託者別，商品別に建玉明細を，また支店別，商品別に建玉数量の確認を行う機能．

4） 入 出 金 管 理

入出金，入出庫の登録，削除，照会を行う機能．当日だけでなく過日の取扱いも可能である．

5） 証拠金過不足管理

変動する商品の値段から売買に必要となる証拠金の計算（値洗い処理）を行い，委託者毎の過不足管理を行う機能．

6） 帳票作成機能

法定帳票や顧客宛て帳票，社内管理資料などを作成する機能．主に夜間バッチ処理にて

作成するが，一部の帳票については画面からオンライン中に作成し，照会することが可能である．

7) 実　績　管　理

営業要員やディーラの実績管理機能として，営業担当者や部署別に，売買件数や金額，手数料収入などを集計，照会する機能．自己売買の実績はディーラ毎に照会が可能である．

8) 営　業　支　援

売買シミュレーション機能により，模擬売買を行い委託者の損益状況のシミュレーションや手数料や証拠金の仮計算を行う機能．

9) 外部インタフェース

東京工業品取引所，外部情報ベンダ（時事通信社），インターネット・ホームトレードサーバとのインタフェースを司る機能．

・東京工業品取引所と注文および約定情報を送受信するインタフェース
・外部情報ベンダ（時事通信社）より商品の価格情報を取り込むインタフェース
・インターネット・ホームトレードサーバと入出金および約定情報を送受信するインタフェース

10) 運　用　管　理

商品属性や会社属性，手数料および証拠金の基本情報を登録，削除，照会する機能．

2.2 システム構成

2.2.1 ハードウェア構成

アウトソーシングセンタに設置した 1 台の ES 7000 を最大八つのパーティションに分け，AP サーバと DB サーバおよび待機用 AP サーバを稼働させる．1 台の ES 7000 につき，取引員企業 1〜4 社の取扱いが可能である（図 1）．

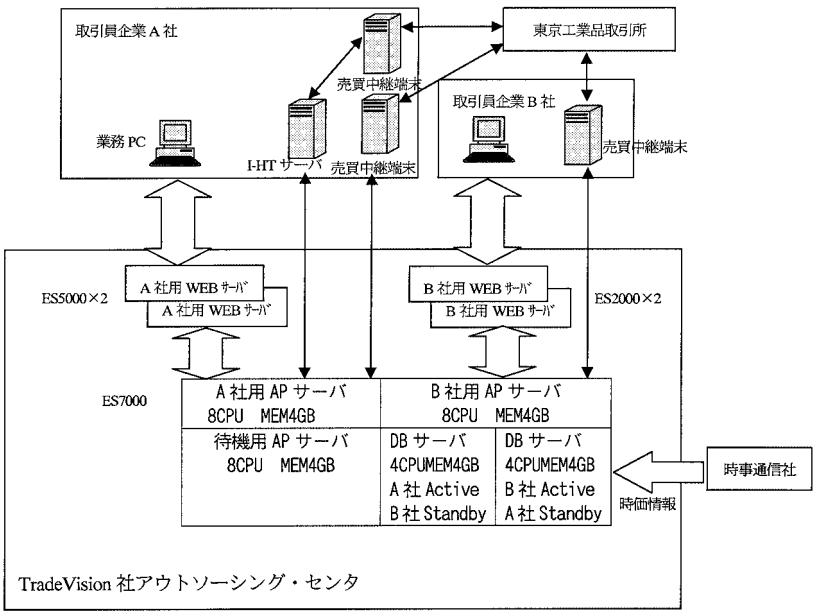

図 1　システム構成（1 台の ES 7000 上で取引員企業 2 社を扱う例）

1） Web サーバ：ES 7000 とは別に 1 社あたり 2 台割り当て，ネットワークロードバランスで負荷分散する．

2） AP サーバ：1 社あたり業務量によって 4〜8 CPU の AP サーバを割り当てると共に，ES 7000 1 筐体当り待機用の AP サーバを 1 パーティション割り当てる．AP サーバ障害時は，手動で待機用の AP サーバへ切り替える（スクリプト一つ実行）．

3） DB サーバ：ES 70001 筐体あたり 2 パーティションを割り当て，MSCS[*1] クラスタリング構成とする．MSCS は，Active-Active の構成で，会社毎にいずれかのサーバをプライマリ・サーバとし，取引員会社毎にインスタンスを分離する．

2.2.2 ソフトウェア構成

本システムは，クライアント側に Web ブラウザのみ搭載することを想定した Web アプリケーションである．また，ES 7000 との親和性や Windows 2000 DCS に対するサポート状況を考慮し，ソフトウェアを可能な限り Microsoft 社製品で統一している（図 2）．

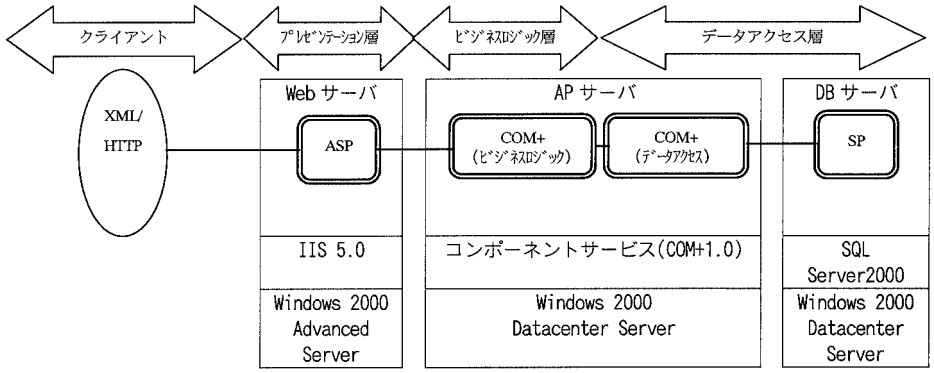

図 2 ソフトウェア構成とアプリケーション構造

2.2.3 アプリケーション構造

アプリケーションの構造は LUCINA[*2] にて推奨する 3 層構造を採用した．

1） プレゼンテーション層（P 層）は ASP[*3] で実装し，クライアント側の入力項目のチェックを JAVA スクリプト，Web サーバ側の入力チェックを VB スクリプトで行っている．P 層では，業務に関するチェックは行わず，入力項目の単体チェックや相関チェックのみ行う．

2） ビジネスロジック層（B 層）は COM＋[*4] コンポーネントでトランザクション処理を実装し，業務ロジックを司る．

3） データアクセス層（D 層）は COM＋コンポーネントから SP[*5] を呼ぶ形式を取っている（図 2）．

4） DB のプログラムインタフェースは OLEDB[*6] を採用した．OLEDB は汎用的ではあるが，開発者の経験や知識にばらつきがあると品質を保てなくなるので，D 層 COM で OLEDB を隠蔽した．値洗い処理などパフォーマンスを重視すべき一部の処理は，ビジネスロジックをデータアクセス層（SP）にて実装することにより，サーバ間のデータ転送量を減らして，処理を高速化した．また，COM＋コンポーネントと SP の引数サイズの

制限は 8 KB なので，これを超える情報を扱う場合は SP 側で計算を行い，結果のみ COM 側へ返す方式としている．

3. アクセス制御

3.1 アクセス制御

COMTRADE で提供する機能について，その利用者の所属や役割などに従い，使用できる機能の範囲やアクセスできるデータの範囲を制限する制御を行っている．COMTRADE の権限機能では，ログイン ID を基にアクセス制御を行っており，次の権限機能を実装している．

1） メニュー表示制御機能

ログインしているユーザに対して使用が許可されている画面のみメニューに表示する．COMTRADE の画面では URL を直接指定できないようにしているため，メニューのリンクからのみ画面へのアクセスを行える．

2） 入力データの処理可否検査機能

画面から入力した情報が，ユーザにとって処理を許可された範囲のものであるかを検査する．この機能は画面上のボタン単位に設定可能である．

3） データ抽出制御

特定の照会画面について，ユーザごとに支店・部・課レベルのデータ抽出条件を指定できる．この機能により，ユーザが許可されていない支店・部署のデータを見ることを防止している．

4） 特殊権限機能

特定の照会画面について，指定した支店や部のデータを除外して表示する．これは，業務上の理由によりある支店や部のデータをユーザに見せたくないときに設定する．

これらの権限設定情報はデータベースのテーブル上に格納している．画面毎・ボタン毎に権限を設定したパターン（支店長，営業担当者，事務担当者など）をいくつか用意し，ユーザ毎にいずれかのパターンを割り当てる方式を取っている（図 3）．

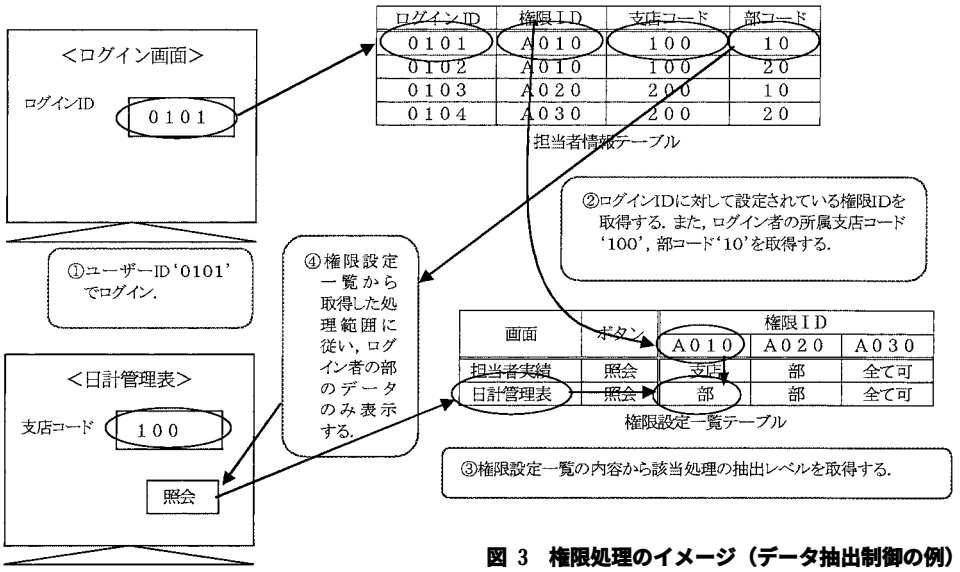

図 3　権限処理のイメージ（データ抽出制御の例）

4. トランザクション管理

4.1 COM＋

トランザクション管理にはCOM＋を用いており，アプリケーション側では特別な考慮はしていない．本システムでは，ビジネスロジック層のコンポーネント（以降B層COMと呼ぶ）でトランザクションを発生させ，データアクセス層はトランザクションに参加する．これを実現するために各B層COMの作成時に，プロパティ設定にてトランザクション・サポートを「必要」と設定している．B層COMは，別サーバ（WEBサーバ）から呼び出されるため，サーバアプリケーションとしてモード設定している．これらの設定により，WEBサーバから呼び出されたB層COMが，独立したトランザクションとして動作する．

データアクセス層COM＋コンポーネント（以降D層COMと呼ぶ）は，プロパティ設定にてトランザクション・サポートを「サポート」と設定することにより，B層COMからの呼び出しに従属し，一つのトランザクションとなる．B層COMから呼び出されるD層COMは同じサーバ上にあるため，ライブラリアプリケーションとしてモード設定している．ASPがプルダウンメニューを表示させる場合，効率の面から直接D層COMを呼び出す形になっているため，ASPから直接呼ばれるD層COMについては，サーバアプリケーションとして設定している．

コンポーネントの各メソッドで処理が正しく終了した場合，ObjectContextオブジェクト（COM＋で用意されている）のSetComplete()メソッドを呼び出すことによりCOM＋がトランザクションのコミットを実行する．トランザクション中に，SetAbort()メソッドを１度でも呼び出すと，COM＋がトランザクションの終了時にロールバックを実行する．

COMTRADEは，WEBサーバとAPサーバが物理的に別サーバとなっており，ASPから直接別サーバ上のCOMを呼び出すことが出来ない．従って，WEBサーバ側にサーバアプリケーション設定されているCOMをプロキシとして設定し，プロキシCOMからAPサーバのCOMを実行している（図4）．

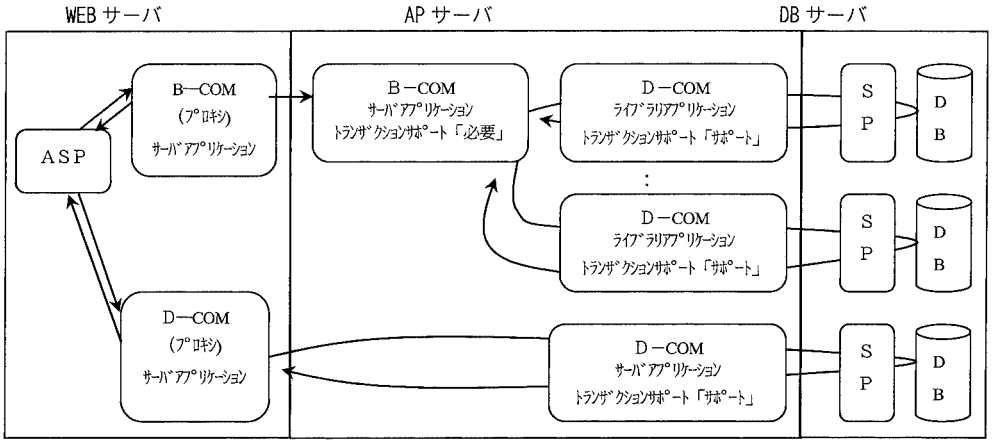

図 4　トランザクション管理

5. レスポンス向上のための考慮

5.1 データベースのアクセス

レスポンス向上という観点から，インデックスの付加，クラスタインデックス（物理的なデ

ータ配列）の見直し，データガーベージ（不要データの削除），機能に特化した SP の作成，データベース接続のプーリングを実施した．

5.1.1 ロック単位の最小化

データアクセス層でのレスポンス向上のためには，キーとなる項目にインデックスを使用するほかに，データのロック単位を最小限にとどめる考慮をしている．金融アプリケーションでは特に注文や約定に関連するテーブルの更新が頻繁に発生するが，COMTRADE ではこれらのテーブルについて行単位のロックを用いている．ロック単位を最小限にしておくことにより，デッドロック回避の効果も期待できる．

5.1.2 デッドロックの回避

COM＋と SQL　Server の組み合わせでは，トランザクション分離レベルの自動変更によって発生するデッドロックの考慮が必要だった．SQL　Server では，トランザクション分離レベルを設定することで他のトランザクションの同時実行操作から読込み操作を分離できる．COM＋のトランザクション・サービスを使用するアプリケーションから SQL　Server へ接続する場合，トランザクション・サービスはトランザクション分離レベルを自動的に最も厳しいレベルである SERIALIZABLE に設定し，SQL　Server のデフォルトである READ　COMMITTED を上書きしてしまう．これは COM＋では比較的軽量で，かつ直列化したトランザクションの使用を想定しているためであり，トランザクションが直列化されていれば，同期処理の整合性について考慮する必要が無いという考え方に基づいている．分離レベル SERIALIZABLE では前のトランザクションがコミットまたはロールバックされるまで，別のトランザクションが修正操作は実行できない[1]．

このとき，同一トランザクション内で SELECT 文を実行したあとに UPDATE/DELETE/INSERT 文を実行すると，デッドロックが発生する．これは SELECT 実行時に参照範囲にロックがかかり，分離レベル SERIALIZABLE が設定されているために次の操作が待ち状態に入ってしまうためである．

COMTRADE では複数の約定レコードを挿入する際に，すでにテーブル上に存在する場合はそのレコードを飛ばして後続のレコードを処理する．この実現方法として，当初は SELECT 文実行により重複を検査し，重複していなければ直後に INSERT 文を実行していた．これはレコードが重複していても SP から正常終了として呼出元 COM へ戻ることにより，COM で正常処理を継続させるためであった．この方式では前述の理由によりデッドロックが発生したため，SELECT 文による重複検査を廃止し，INSERT 文で重複エラーを発生させてエラーコードを返し，呼出元 COM 側では重複エラーコードを判断して後続レコードを処理する方式に変更した[3]．

5.1.3 インデックスの付加

効率改善において，最も効果を挙げたのが，インデックスの付加であった．時間的な制約があったため，全体の 70% を占める 12 の機能を選出し，機能の観点からクラスタインデックスとインデックス追加の試行テストを行い，適正なインデックスを付加した．これらの対応により，処理時間が約 87% 削減された（表 1）．

表1　改善効果

画面	ボタン	処理時間（ミリ秒）		備考
		改善前	改善後	
売買シミュレーション	照会	3,507	691	支店：７１０，委託者：２８２６
	仮計算	308	351	
新規継続登録・変更・削除	照会	887	78	支店：７１０，委託者：２８２６
節別値段照会	照会	1,631	175	取引・所商品：２０５１
売買データモニタ	照会	1,529	150	支店：７１０，委託者：２８２６
委託者預り一覧照会	照会	6,162	564	支店：７１０，委託者：２８２６
値洗請求一覧照会	照会	327,773	41,468	支店：７１０
日々業務実績照会	照会	3,297	1,141	支店：１３０
入出金データモニタ	照会	1,618	176	支店：７１０，委託者：２８２６
委託者別値洗照会	照会	1,578	459	支店：７１０，委託者：２８２６
新規継続実績モニタ	照会	131	87	支店：７１０
委託者別建玉明細照会	照会	1,298	161	支店：７１０，委託者：２８２６

5.1.4　データベース接続プーリング

同時にアクセスが集中する環境で，リクエストごとにデータベースに対するセッションを生成していてはシステム上で頻繁にボトルネックが発生するため，データベースとのセッションをある一定数確保（プーリング）し，多数のユーザ要求をそこに割り振る機能を採用した．これにより，データベースのコネクション回数が減り，処理時間の短縮とシステム資源の有効利用を可能とした．この機能はD層COMの，SP毎に生成されるヘッダファイルに“db.Open With Service Components”を記述することにより実現している．

5.2　画面表示高速化の考慮

Webブラウザは画面表示の処理に時間がかかるので，表示時間を速くするために，いくつかの工夫をしている．

1）　メニューの表示については，フレームを業務画面と分け，画面左側に常時表示している．
　メニューの内容はログイン直後に一度だけ読み込む．フレームを業務画面と分けることにより，画面を切り替えた場合でもメニューを再度読み込む必要が無い．

2）　プルダウン・メニューに表示する項目の内容は業務画面を呼び出す際に毎回サーバから取得しているが，男女など決まったものについてはASPに直書きして表示の高速化を図っている．

3）　売買シミュレーションでは，文字列を連結して出力する処理を，文字列出力を繰り返すように変更することにより処理時間を半分に効率化した．

4）　全データを取り込んでから表示するのではなく，1画面分データを取得したら画面表示することにより，体感待ち時間を少なくした．

6.　帳　　票

商品先物業界では，複写式の帳票をインパクトプリンタで印書する方式が一般的であり，本システムでの実装方法をどうするかが課題となったが，高速インパクトプリンタが高額であり，ES 7000との接続実績も無かったため，レーザプリンタを前提とした帳票作成システムを採用することとした．また，取引で発注あるいは取引所で約定した情報を一方送信でラインプリンタに印書する機能をIA（InquiryAnswer）を前提としたWebシステムでどう実装するかが大きな課題となった．

この課題については帳票を電子ファイル化して，ブラウザ画面から参照・印刷可能とすることで対応した．帳票の電子ファイル化により帳票の改ざん防止という副次効果も得られた．ファイルサイズを小さく抑えることができるという理由から，COMTRADE では帳票をすべて A 4 サイズの PDF 形式にて電子化している．新しく注文や約定が入った，つまり新たな通知レコードが作成されたことを使用者に通知する仕組みについては 6.3 節の自動照会・自動通知，で説明している．

6.1 リアル帳票

① ブラウザからの要求により業務 ASP が業務 COM を呼び出す．
② 業務 COM は SP を用いて DB を参照してファイルのフルパスを業務 ASP に返す．
③ 業務 ASP はリアル帳票専用 ASP を呼び出す．
④ リアル帳票専用 ASP は帳票 PDF ファイルをバイナリモードで読み込み，BinaryWrite する．BinaryWrite することによりデフォルト PDF ファイル名が ASP 名になり，エンドユーザに PDF ファイル名がまったくわからないようにした．PDF ファイル名は現在日時（ミリ秒）をファイル名に含め，重複を避けている（図 5）．

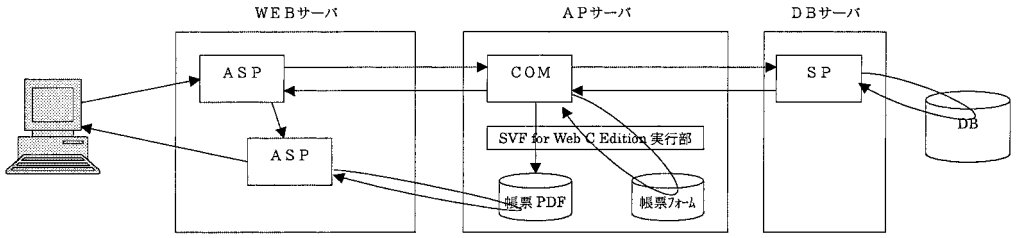

図 5　リアル帳票の流れ

6.2 バッチ帳票

バッチによる帳票作成とブラウザからの参照要求が非同期になることを除いて，基本的な仕組みはリアル帳票と同じである（図 6）．

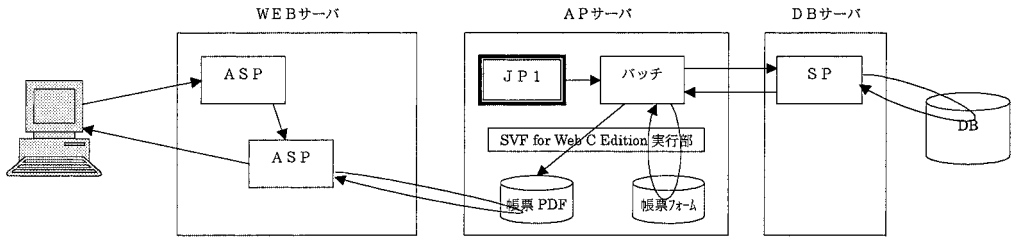

図 6　バッチ帳票の流れ

帳票を電子ファイル化することで，過去分の帳票の参照や検索が容易に行える．帳票は全て A 4 サイズにしており，印刷に既存のレーザプリンタを利用することにより，取引員としては OA 機器の有効活用が図れる効果もある．3 枚複写で，顧客用，保存用，取引員用に出力が必要な帳票に関しては，3 枚を 1 レコードの帳票として PDF ファイルを作成した．

月次や年次で出力する社内管理帳票はページ数が多いので，大量印刷にかかる処理時間の長さが懸念された．汎用機システムでは，取引員企業の高速プリンタで連続用紙に全社分の帳票を一括出力してから各部署に配布していたが，本システムでは各部署で帳票を直接参照，印刷することができるようになり，出力を分散化することで印刷時間は，汎用機時代よりも短縮された．

PDF ファイル作成は，当初自前で帳票作成システムを構築する予定であったが，開発工数，スケジュール，保守性の観点から他社製品を使用することとなった．いくつかの候補の中から，帳票作成の実績，および DCS 上で稼働できる確認（カーネルモードを使用していない）が取れたため，翼システムの Super Visual Formade[*7]C Edition を採用した．

なお，帳票についても 3 章で述べたアクセス制御の対象としており，ユーザは許可された帳票のみ参照できる．

いくつかの画面については，画面イメージを出力したいという要件があった．これらの画面では印書ボタンを設け，ブラウザの印刷機能を利用してイメージを出力する方式とした．この場合，Internet　Explorer の仕様で，情報量が多いと画面の一部しか出力されないという現象が発生する．したがって，この機能は参考情報として紙へ出力したい画面にのみ適用することとし，すべての情報を残す必要のある画面はリアル帳票として別に実装した．

6.3　自動照会・自動通知

前述したように，注文や約定が新しく発生した際指定されたプリンタへの注文・約定情報の印書は，画面照会による方式で代替している．しかしながら商品先物基幹業務は，注文と約定が中核のシステムで，新しく注文が入ったあと，約定した場合に，即時に認識する必要がある．このため画面による照会は，自動照会でなければならず，かつ，新しく注文が入った際や約定した際に音を鳴動させることが前提条件となった．（今までは，プリンタが動作する際の音を聞き，新しく注文や約定が来た事を認識していた）

この機能を実現するにあたって，一定間隔で HTML データをサーバから取得する方式では次に示す問題が発生する．

① 情報を取得する度に画面全体が書き換わるため，ブラウザ画面表示がちらつく
② スクロールしていた場合には，画面表示の度にスクロール位置が先頭に戻ってしまう
③ 毎回全データを取得するため，ネットワークへの負荷が高くなる

COMTRADE では，クライアントが Web サーバからデータを XML[*8] 形式で取得し，動的に画面を描画することで，データの取得と画面の描画を分離した（図 7，8）．

XML ではデータにタグを付加してデータ構造を明確に定義できる．複数の通知内容を XML の子ノードの集合として記述することにより，構造化されたデータとしてクライアント側で扱いやすい形式となった．

クライアント側で更新日時の最大値（XML のルートノードのアトリビュートとして記述）を保持しておき，2 回目以降の検索時には，前回取得時以降に更新されたデータのみ取得する方式とし，ネットワーク負荷軽減を図った．また，画面の変更箇所だけを描画することにより，画面のちらつきを抑えている．テストでは，128 MB メモリ搭載のクライアントで通知を 3,000 件まで保持可能であることを確認済みである．

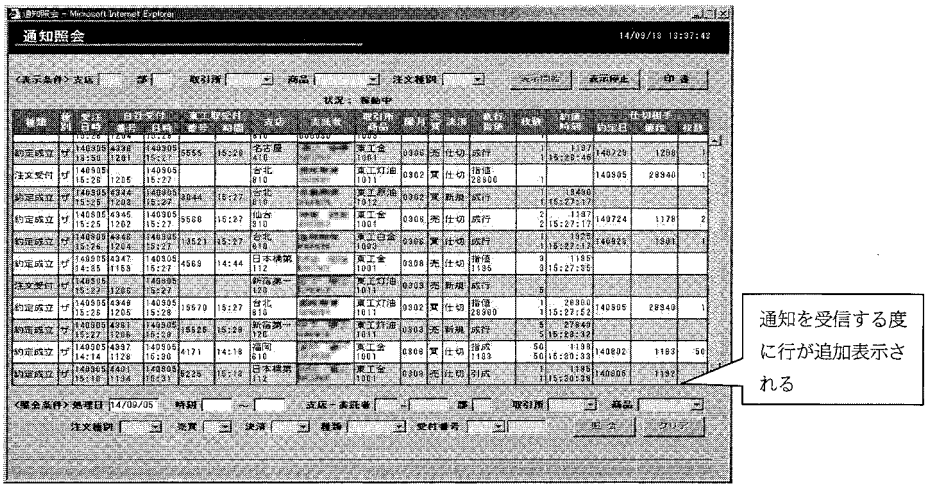

図 7　通知照会画面

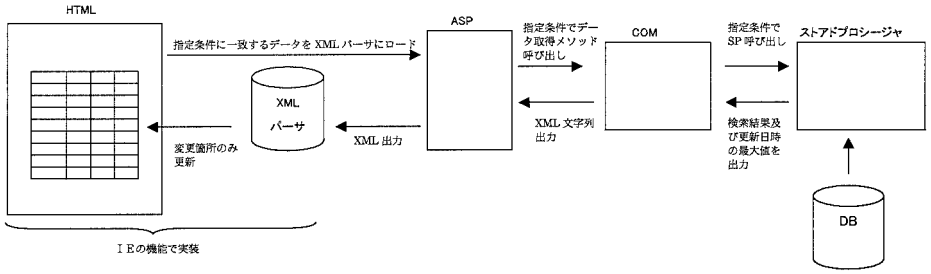

図 8　XML を利用した自動画面更新方式のイメージ

7. お わ り に

現在システムは安定稼働を続けており，短期間の評価ではあるが，今のところ，目標としていた汎用機並みの安定性を実現できている．また，バッチ処理の処理速度向上により，従来のシステムよりも早く業務を終了することができるという評価を得ている．2003 年内に，さらに 4 社の本番が予定されている．

機能面では，東京工業品取引所新システムのインタフェース，大口手数料対応を完了し，今後は T＋1 対応（2003 年 6 月本番予定），東京工業品取引所のオプション取引対応（2003 年秋本番予定），小口手数料対応（2004 年 1 月本番）を実施し，自由化の波，制度変更への対応を行い，より魅力ある商品として育てていく予定である．

＊ 1　MSCS：Microsoft Clustering Service の略．Microsoft 社の提供する共有ディスク型クラスタ．

＊ 2　LUCINA：当社で推奨している，ビジネスアプリケーション向けのコンポーネント指向開発技法．

＊ 3　ASP：Active Server Pages の略．WWW サーバ側で，JavaScript や VisualBasic Script などのスクリプト言語や，各種 ActiveX コンポーネントを動作させるためのフレームワーク．

＊ 4　COM＋：オブジェクト指向の実装モデルである COM に，トランザクションサービスなどいくつかのサービスを統合したもの，またその実行環境を指す．

＊ 5　SP：Stored Procedure の略．SQL 文を格納し，実行できる．

＊6　OLEDB：Object Linking and Embedding DataBase：Microsoft 社によって開発された，データベースの種類によらず統一的な手法でデータベースにアクセスするためのプログラミングインタフェース．
＊7　Super Visual Formade：翼システムが提供する帳票作成ソフトウェア
＊8　XML：eXtensible Markup Language の略．HTML のようなシンプルなフォーマットで文書構造を記述でき，独自にタグを定義できることが特徴のマークアップ言語．1998 年に W3C（World Wide Web Consortium： WWW で使われる技術を標準化する団体）により標準化勧告された．

参考文献

［1］ Dusan Petric 著, トップスタジオ監訳, SQL Server 2000 ビギナーズガイド, PP 338－340，翔泳社，2001
［2］ David S.Platt 著，豊田　孝監訳，COM＋テクノロジガイド，日経 BP ソフトプレス，1999
［3］ 益田　智之著，TechnicalSymposium 2002,ES 7000 上での金融 Web アプリケーション開発における留意点
［4］ Windows 2000 ビジネスアプリケーション構築ガイド
URL http://www.zdnet.co.jp/help/howto/win/win2000/0007complus_vb/

執筆者紹介

伊 藤　公 一 郎（Kouichiro Ito）
1967 年生．1989 年横浜国立大学教育学部卒業．同年日本ユニシス(株)に入社．金融系基幹業務を中心とした開発・保守を担当．1997 年より商品先物取引基幹業務の開発に従事．現在，システムサービス本部金融第一システム統括部システム四部に所属．COMTRADE, HOMETRADE その他のソフトウェア商品の企画・開発を担当．