高性能ＸＭＬ文書データベースサーバの開発

Implementation of a Hi-performance XML Document DBMS

中川路 充 [1)]

Nakakawaji Mitsuru

1) 株式会社 プロキューブ（〒650-0012 兵庫県神戸市中央区北長狭通 4 丁目 9-23 KHK 元町ビル 301 号 E-mail: mitsuru@procube.info ）

ABSTRACT. The framework provided by groupware servers, which consists of the semi-structured document database and the workflow management system, is efficient to implement application system that aids a variable business process. But typically groupware servers lacked performance, robustness and availability to use as enterprise level server. So users are suffering from some system failure or some performance problem. We have developed XML Document DBMS : TagStore which provide performance, robustness and availability as same level as generic RDBMS. That is, TagStore has both advantage of the document storage like groupware server and advantage of RDBMS. Then TagStore is best solution of storage for a workflow system that aids a variable business process of enterprise level organization. This document describes the goal and the technologies of TagStore.

１．背景

今日まで、企業はさまざまなビジネスプロセスを支援するシステムを導入してきたが、これらは今まで事務職がこなしていた定型的な業務をシステム化しているにすぎない。

真のホワイトカラーの生産性向上によるメリットを獲得するためには、たとえば、稟議書、複雑な受発注プロセスのフォロー、政府・自治体の文書の回覧など、非定型業務を支援するシステムを構築する必要がある。

このようなビジネスプロセスはそれぞれの商習慣にあわせた特異なプロセスを含んでいる場合が多い。このため、定型的な枠組みを提供するパッケージソフトウェアを適用すると、そのカスタマイズコストが増大し、かえってコスト高になる場合が多く見受けられる。

一方で、グループウェアサーバなどが提供している半構造化文書とワークフローの仕組みは、その柔軟性から前述のような多様性を持つ業務を支援するシステムを構築する際の有効な手段である。

しかし、既存のグループウェアサーバは、もともと小規模な運用を前提として設計されているため、性能、信頼性、可用性の面でユーザの要求を満たしていないものが多い。一部の企業はホワイトカラーの生産性向上を目的として文書回覧システムの構築に取り組んできたが、開発コストが大きい上に性能、信頼性、可用性の面で要件を満足できず、苦しんでいる。筆者は、実際にサーバのトラブルや性能で苦労しているユーザを多数見てきた。すでに SIer の中にはこのような手間ばかりかかって利益につながりにくいワークフローシステムを敬遠する業者さえ出ている状況である。

２．目的

前項で示したように、現状では、非定型業務を支援するシステムを実装するためのデータベースとしては、グループウェアサーバか RDBMS を利用することになるが、それらはそれぞれ、「帯に短し、たすきに長し」であり、それぞれ短所長所が見られる。

そこで、これらのデータベースの短所を避け、長所を併せ持つデータベースを提供することを目的とし、XML 文書を格納することに特化した DBMS である TagStore を開発した。

３．文書データベースの要件

この章では、非定型業務を支援するシステムを構築する場合に、その基盤となる文書データベースに必要となる機能要件について RDBMS と TagStore を比較して考察する。

（１） 可変長データに対するアクセス性能

一般的に、非定型業務で利用される文書はその大きさがまちまちで、かつ添付ファイルにマルチメディアデータなどを含んでいる場合など、サイズが大きいものが頻繁に出現する。

このような文書を RDBMS のようにゴミ領域を即時回収する方式のデータベースに格納すると、データのフラグメントが発生し、性能が劣化する場合がある。また、このようなフラグメントによる性能劣化は本番稼動後徐々に顕在化するため、システムの性能設計を難しくする。

TagStore では、追記型でデータベースを実装し、ゴミ領

域の回収を非同期にコピー＆コンパクションアルゴリズムにより実行する（図 1 TagStore のゴミ回収方式を参照）。このため、フラグメントの発生が少なく、性能劣化がない。また、領域を循環して再利用する方式を採用することにより、ゴミ回収をオンライン（サービスを停止せずに）で実行でき、24Hours*7Days のサービスを提供する。ゴミ回収処理の実行時はスループットが若干低下するが、これは夜間や休日などの負荷が低い時間帯に実行することにより回避できる。

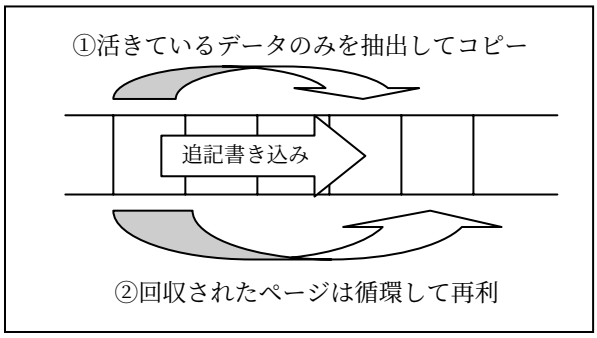

図 1 TagStore のゴミ回収方式

（2） 高度な文書一覧機能

非定型業務を支援するワークフローシステムにおいては、エンドユーザに対する文書の一覧表示が持つ情報量がシステムの使いやすさに大きく影響する。複雑な構造をもつ複数の文書から最適な文書一覧を表示するために以下のような機能が必要である。

- 異なる種類の文書から特定の属性の抽出。たとえば、購入申請書や交通費清算書などが混在している文書一覧において、発注コードと金額を抽出など。この機能により、文書の多様性を吸収することができる。RDBMS のように定型データのみを格納するデータベースでは実現できない。
- ユーザのコンテキストと文書を照合して抽出する機能。たとえば、エンドユーザが回覧先に自分が含まれている文書のみを表示する機能など。
- 派生属性（実際の文書には値が記述されていないが文書の内容から演算で導かれる値）を生成する機能。たとえば、受注伝票内に受注生産品が含まれているものだけを抽出。

TagStore では、文書を XSL にかけることにより、上記のような高度な文書一覧を生成できる。また、問い合わせを XML と相性の良い XSL で記述できることは開発者の負担を軽減する。

さて、RDBMS や ORDBMS では、このような高度な文書一覧を生成する問い合わせも参照時に実行される。通常、文書一覧表示は文書更新に比べて頻繁に実行されるので、このことは性能低下の原因となる。

TagStore では XSL の処理を文書の更新時に実行することにより、性能を確保している。また、ワークフローのようなシステムでは、一つの文書集合に対して、多種の問い合わせが必要となることがないので、更新時に問い合わせを実行しても性能低下の原因となることはない。

４．信頼性と可用性の要件

そもそもグループウェアサーバはそれほど高い信頼性や可用性を要求しない小規模な運用を前提として設計されてきた。しかし、TCO の削減が叫ばれて、サーバー機能が集約されるとともに EAI などのアプリケーション連携が進んだため、文書回覧などのワークフローシステムにも高い信頼性と可用性が必要になった。

この章では TagStore の信頼性と可用性に対応する機能を一般的なグループウェアサーバと比較しながら説明する。

（1） トランザクション処理

ほとんどの既存のグループウェアサーバでは、トランザクション処理をサポートしていない。したがって、メモリ不足やデッドロックなどのトランザクション障害が発生した場合にデータの一貫性を保つのはプログラマの責任となる[1]。プログラマはエラーが発生した時点で全ての更新を元へ戻すコードを書く必要がある。さらにシステム障害発生時には一貫性が確保できないため、データに矛盾が生じ、ワークフローが途中で消失したり、同じものが二つ現れたりといった現象が発生する。

TagStore では、トランザクションをサポートするため、Commit 命令／Abort 命令により、自動的に一貫性が保証される。

また、グループウェアサーバの中には、システム障害発生時の自動復旧機能が無いものや、復旧に長時間かかる（データベース全体をなめまわして矛盾が無いかをチェックするタイプ）ものがある。これらのグループウェアサーバでは、サービス再開までの時間が長くなり、可用性を確保できない。

TagStore はログの内容から障害発生時のトランザクションの内容を復旧し、その一貫性を保証しながらデータベースを復旧する。また、スナップショット機能により、スナップショット採取後のログのみを復旧対処とすることにより、復旧時間を短縮し、可用性を確保する。

この機能により、クラスタ機上での運用でも待機系へのフェールオーバーを速やかに実行することができる。

さらに TagStore では 2 フェーズコミット機能を実装し、分散トランザクションにも対応する。この機能により、RDBMS や MQ などの他の回復可能リソースと一貫性を保つことができる。

（2） オンライン差分バックアップ機能

グループウェアサーバの中にはバックアップ採取のためにサービスを停止する必要があるものがある。さらに、差分バックアップをサポートしていないものは、毎回データベース全体のバックアップを採取しなければならないので、バックアップ採取に長時間かかる。このようなグループウェアは夜間に長時間サービスを停止してバックアップを採取する必要がある。

TagStore ではオンライン差分バックアップ機能を提供しており、無停止でバックアップを採取できる。また、前回のバックアップからの差分情報のみを採取できるため、短時間でバックアップが採取できる。この機能と、前述のオンラインゴミ回収機能と組み合わせて 24Hours*7Days のサービスを提供することができる。

５．更なる性能を求めて

TagStore には、前節までの技術要素以外にも、性能向上のためのいくつかの工夫点がある。

データベースの性能はディスクに対するアクセスの回数に大きく依存する。ディスクはページに区切られて管理されており、複数のページにアクセスするとディスクのシーク（磁気ディスクヘッドの移動）が実行される。このシークはメモリ上の処理に比べると 1000 倍以上の時間がかかるもの[2]で、このシークの回数を減らすことがそのままデータベースの性能を向上させることになる。

TagStore ではデータを小さく格納することにより一つの文書にアクセスする場合にアクセスが必要なページ数を最小化している。

（１） 差分記録＋参照キャッシュ

TagStore では XML 文書を更新したとき、文書全体を保存するのではなく、更新前の文書との変更点のみを保存する。これにより、更新時の書き込みデータ量を小さくし、更新スループット性能を向上させている。

ここで、XML 文書は差分情報のチェーンとして格納されるため、文書の参照処理は、チェーンを拾い集める処理となり、アクセスページ数の増加を招き、コストが高くなる。これを回避するために文書のキャッシュを保持している。文書の最新イメージをキャッシュすることにより、この問題を回避している。

また、この機能を用いて XML 文書の複数のバージョンを効率よく格納できる。

（２） ゼロ・パディング格納方式

TagStore では、文書をデータベースやキャッシュに格納するとき、情報をビット単位につめて格納する。これにより、パディングのための不使用領域を 0 にする。たとえば、5 ビットの数値と 11 ビットの数値を格納した場合でも、アライメントのためのパディングを必要とせず、2 バイトしか使用しない。

（３） 型予測文書

XML 文書は、その形式が決まっている場合がある。DTD や XML Schema により厳密に決められている場合もあるが、そうでなくてもほとんどの場合、アプリケーションのロジックから文書の形式が決まる。

TagStore では XML 文書に対してその形式を記述する型予測文書を付与することができる。同一の形式の文書には同じ型予測文書が結び付けられる。型予測文書は文書を格納するときに文書タグから実データを抽出するのに用いられる。型予測文書に結び付けられている文書は文書タグ情報を保存せず、実データのみが保存される。これにより、ディスク上のデータサイズを最小化し、性能を向上させている。型予測文書はあくまでこの目的で利用されるもので、XML 文書が型予測文書による予測どおりの形式でなくてもエラーにはならない。その場合は、文書タグ付きで保存される。部分的に型予測から外れる場合は、その外れたノードの部分だけがタグ付きで保存され、予測どおりの部分は実データだけが保存される。

通常の RDBMS では、スキーマと異なるデータは格納できないが、TagStore ではあえてスキーマではなく型予測とすることにより、例外データを格納できるようにしている。

６．既存 XML データベース製品の問題点

現在、市場にある XML データベース製品は、その実装方式により、いくつかに分類できる。ここでは、XML 文書が最終的にディスク上で連続領域に配置されるかに着目してそれらを分類し、それらの問題点を TagStore と比較して指摘する。

（１） エレメント分解型

RDBMS や OODBMS 上に XML の 1 エレメントを 1 レコード（OODBMS では 1 オブジェクト）に格納するタイプのもの。現在、主流となっている XML 専用データベース製品は実は OODBMS 上に実装されており、このタイプに属する。

このタイプの製品は XML 文書としての構造を失わずにそのまま格納されるため、実装がシンプルであり、XML としての特性がフルに活かされている。

しかし、一つの XML 文書がディスク上で分散して配置されるので、性能が確保できない。実際に簡単なアプリケーションで TagStore と比較したところ、スループット性能で約 8 倍の差があった（実測値）。この差は、1 文書あたりのノード数が増えるとさらに拡大するものと考えられ、このタイプの実装は実用レベルには不適切であると考えられる。

（２） コンプレックスレコード型

RDBMS 上に 1 文書を 1 レコードとして格納するもの。文書の最上位レベルのエレメントはフィールドに分解されるが、2 レベル以上あるエレメントは XML のままフィールドに保持される。

このタイプの場合、アプリケーションの設計時に「最上位レベルに文書の属性を配置する」、「その属性を構造化してはならない」などの配慮が必要で、XML としての柔軟性を活かせない。そもそもこのような配慮をするのであれば、XML でデータを保持する必然性が無い。さらに XML 文書の構造変更がそのまま RDBMS のスキーマの変更となり、XML の柔軟性のメリットが得られない。

TagStore では XML 文書をそのまま保持することと高度な文書一覧機能を提供することにより、XML のメリットを最大限に活かすことができる。

７．RDBMS との分岐点

さて、「３．文書データベースの要件」で述べたように TagStore は複雑な文書を扱うために最適化されている。ここで文書が複雑であるとは、以下のような性質を持つことである。

- 文書の大きさがまちまちである
- 文書の構造に多様性がある（よく似ているが少し構造がちがう文書がある）

たとえば、売上伝票は単純であるが、出張報告書は少し複雑で稟議書はかなり複雑であると言える。そして、複雑であればあるほど RDBMS よりも TagStore のほうが性能面で有利である。逆に単純なデータばかりを扱うアプリケーションでは RDBMS のほうが有利になる。これを

図にすると「図 2 RDBMS と TagStore の性能分岐点」のようになり、ある一定の複雑度を境に TagStore を使用するべきか RDBMS を使用するべきかが分かれる（ただし、これは理論的な予測であり、実測にもとづいているわけではない）。

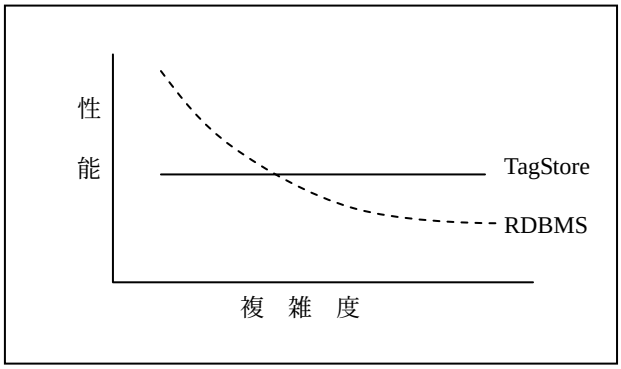

図 2 RDBMS と TagStore の性能分岐点

８．開発状況と開発ロードマップ

IPA の平成 12 年度未踏ソフトウェア創造事業で TagStore のプロトタイプを開発した。これにより、TagStore のコア部分については完成した。このプロトタイプで既存製品との性能比較を行い、実装の有効性を確認した。

しかし、いくつかの重要な機能が未実装である。また、商用製品として販売する事業を計画しており、そのためには運用周りのツールおよびアプリケーション開発のためのリソースを開発する必要がある。

現在、開発のロードマップとして以下のように計画している。

項目	時期	説明
コア部改善	2002 年 3 月	コア部分にまだ性能改善の余地があり、これを改善する
スタンダードβ版	2002 年 6 月	製品化のための第 1 バージョン。分散トランザクション、クラスタ機対応、オンラインバックアップを除く機能を完成させる。パイロットユーザや OEM 提供先に提供。
スタンダード製品版	2002 年 10 月	運用ツール、開発リソースを整備し、製品として販売する
エンタープライズ製品版	2002 年 12 月	分散トランザクション、クラスタ機対応、オンラインバックアップを完成させ、エンタープライズ版として販売開始

９．まとめ

TagStore は、非定型業務を支援するシステムを実装するためのデータベースとして設計された。IPA の平成 12 年度未踏ソフトウェア創造事業では、そのプロトタイプを開発し、その実装の有効性を確認した。この論文では TagStore の実装技術を紹介し、性能、信頼性、可用性の面で既存製品を上回ることを示した。

今後、2002 年末までにこれを商品化し、販売する事業を計画中である。

最終的に TagStore が XML 文書 DBMS のスタンダードとして認知されるよう事業を展開していきたいと考えている。

［1］フィリップ・A・バーンスタイン、エリック・ニューカマー著 大磯 和広 他 訳:トランザクション処理システム入門、p11、日経 BP 社（1998）

［2］宇野 俊夫著：ディスクアレイテクノロジ RAID、p53、エーアイ出版（2000）