



・ Achieve Agility ・

～ 激動する環境の変化に機敏に対応するために～

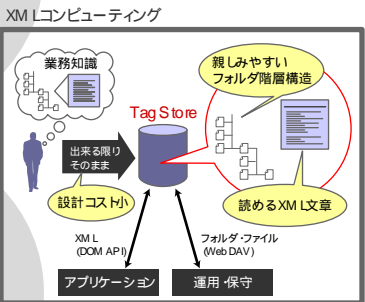
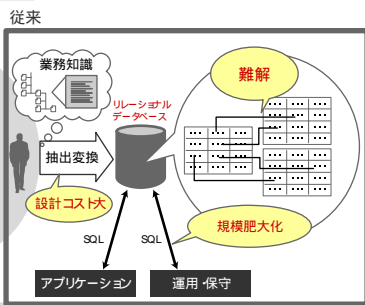
いまや企業の情報システムはその複雑度を増し、とするとその複雑さが企業活動の足かせにさなりつつあります。一方で企業には環境の変化に対応するAgility(機敏性)が要求されています。弊社では、最新のXML技術を使って情報システムの柔軟性を確保し、結果として企業に機敏性をもたらすXMLコンピューティングを提案しております。



XMLコンピューティングとは

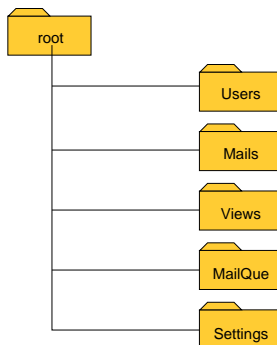
従来の情報システムではリレーショナルデータベースを中核としていたため、すべてのデータを表形式に変換する必要がありました。その結果、ちょっとしたシステムでも数十個のテーブルからなる複雑なデータベースを必要とし、設計した担当でさえその扱いに神経質にならざるを得ませんでした。さらに大規模なシステムでは、テーブルの個数が数百にのぼるものもあります。数万個のテーブルが生成されるメジャーなERP製品もあります。従来の技術では、このような多数のテーブルを複雑に結合してシステムを構成せざるを得ませんでした。様々な技術革新の結果、システムにとってのベストソリューションも変わりつつあります。

弊社では、すべてのデータをXML形式にすることにより、シンプルにデータを扱うことができるXMLコンピューティングを提案しています。



親しみやすいフォルダ階層構造

XMLコンピューティングでは、データをフォルダ階層構造で整理します。つまり、パソコンで文書を整理するときと同じように、目的別や種類別に文書を分けて格納します。たとえば、会議開催通知のフォルダの下に月ごとのフォルダを作成し、そこに会議開催通知の文書をいれるといった具合です。



読めるXML文書

XMLを情報システムに導入することには様々なメリットがありますが、中でも弊社が注目しているのは「XMLが人間にもコンピュータにも読みやすい形式である」という点です。人間の側からはテキストとして「読む」ことができ、コンピュータの側からは構造化されたデータとして「扱う」ことができます。

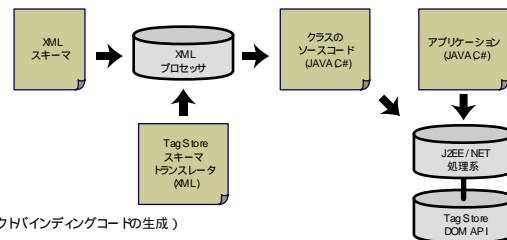
リレーショナルデータベースのテーブルとは異なり、開発者にとっても「読めるXML文書」は直感的に理解しやすいものと言えるでしょう。

```
<会議開催案内>
<会議開催案内 ID=1</会議開催案内 ID>
<利用者 ID=dm-nakata</利用者 ID>
<利用者名 沖田 充</利用者名>
<開催日付>15/08/09</開催日付>
<開始時刻>10:22:00</開始時刻>
<終了時刻>12:22:00</終了時刻>
<主催者 ID=dm-nakata</主催者 ID>
<主催者名 沖田 充</主催者名>
<タイトル>リリース版の内容について</タイトル>
<説明>リリース版の内容について議論します</説明>
<個人情報>
<LoginID=dm-nakata</LoginID>
<氏名 沖田 充</氏名>
<出欠 未回答</出欠>
<日付>15/07/30 10:22:00</日付>
<備考></備考>
</個人情報>
<個人情報>
<LoginID=s-kitano</LoginID>
<氏名 北野 繁樹</氏名>
<出欠 未回答</出欠>
<日付></日付>
<備考></備考>
</個人情報>
<個人情報>
<LoginID=i-suzuki</LoginID>
<氏名 鈴木 一郎</氏名>
<出欠 未回答</出欠>
<日付></日付>
<備考></備考>
</個人情報>
</会議開催案内>
```

(XML 文書の例)

オブジェクト指向との高い親和性

XMLは、コンピュータから見るとデータとして扱うことができます。XMLでは、オブジェクトバインディングと呼ばれる技術により、データをオブジェクトとして扱うことができます。オブジェクトバインディングを用いると、プログラミング時にデータがXMLであることを意識する必要がなく、オブジェクト指向言語のネイティブなオブジェクトとして扱うことができます。



(オブジェクトバインディングコードの生成)

標準化された暗号化、署名

XMLの標準化団体であるW3Cでは、XMLの暗号化や、署名する場合のフォーマットについての標準化も行っています。この標準に従えば、ベンダー独自の署名や暗号化を利用する場合に比べてオープン性が確保でき、データの互換性が高くなります。標準化されている仕様では、部分署名、部分暗号化、リンク先を含めた署名などをサポートしており、XMLの柔軟性を最大限活かしながら署名、暗号化を実装できます。たとえば、稟議書のような文書において「システム管理者はヘッダー情報を読めるが内容は読めない」とか「本文への署名の有効性を失わずにコメントを付与する」などの機能が実装できます。

Nakakawaji Mituru



・ Achieve Agility ・

～ 激動する環境の変化に機敏に対応するために ～



XML コンピューティングで何ができるのか

ここでは XML コンピューティングで何ができるのかについて考えてみます。

非定型データを演算加工する

そもそもリレーショナルデータベースは企業間で行われるトランザクション (= 取り引き) を扱うことを目的として開発されました。そこでは、定型データをコンピュータで演算加工処理することが主な機能であり、その信頼性や可用性が重視されてきました。しかし、一般にリレーショナルデータベースに非定型データを格納すると、ディスク上でのフラグメント化が発生し、運用中に徐々に性能が低下するという問題があります。つまり、リレーショナルデータベースは、文書のようなデータのサイズがまちまちなものを格納するには向いていないツールなのです。

一方でグループウェアサーバは、企業内での知識共有に利用されることを目的として開発されました。そこでは、非定型データを扱うための工夫がいろいろと取り入れられ、いまやそれらの技法は一般的なものと広く普及しています。たとえば、電子メールのヘッダ部分のように、文書の一部をコンピュータでも処理できるように構造化し、人間とコンピュータの両方が処理できるようにした文書形式 (これらは専門用語で半構造化文書と呼ばれています) もグループウェアの技法のうちの一つです。XML はこの半構造化文書をさらに進めたものと言えるでしょう。反面、グループウェアサーバは主に情報を共有することを目的として設計されていたため、情報を演算加工処理しようとする信頼性の面で問題が出ます。一例として、グループウェアサーバでは OS のパニックや電源断などのシステム障害が発生した場合、更新中のデータの一貫性は保証されず、結果として不整合を生じます。また、内部構造 (データ間のリンク関係など) 自身にも不整合が生じる可能性があり、これを解消するためにデータベース修復ツールを実行する必要があるものもあります。この手のツールは、すべてのデータを読み出して矛盾点を解消する操作を行うため、データの量が多いほど時間がかかります。システム障害発生後、修復ツールの実行に時間がかかり、サーバ再開までに膨大な時間を要したという例もあります。これはトランザクションログを参照して高速に回復するリレーショナルデータベースと比べると、ある意味機能が欠落しているとも言えます。言い換えるとグループウェアサーバは、情報共有機能に特化したサーバであり、演算加工処理には対応していないツールであるとも言えます。

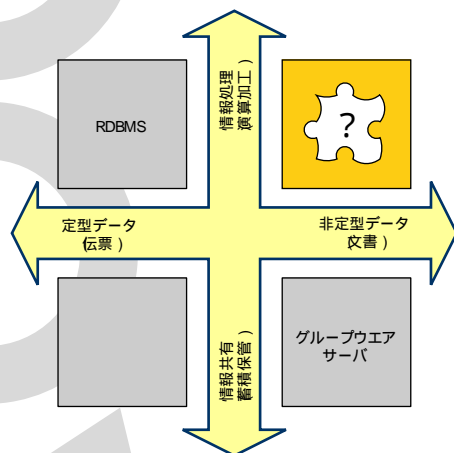
文書トランザクション

前節で述べたとおり、従来のコンピュータシステムでは定型データを交換したり、数値データを演算加工処理したりする数値トランザクションが主な仕事でした。しかし、圧倒的な CPU パワーの向上、ブロードバンドネットワークの普及、ユーザの情報リテラシーの向上などを背景に、非定型データを交換し、知識を演算加工処理する文書トランザクションにシフトしつつあります。

また、一般的には定型データであると思われるものでも、実は非定型データつまり文書として考えたほうが柔軟で高度なアプリケーションを構築できる場合が多々あります。CRM で扱われる顧客データや SCM で扱われる資源データなど、本来は多様性や豊富な表現力が要求されているのに、従来型のシステムで処理するためにしかたなく情報量を縮減させているのが実情ではないでしょうか？

リレーショナルデータベースを中核とした従来のコンピューティング技法では、人間の持っている知識をコンピュータで扱いやすいように変換、縮退してデータを扱うため、業務自身を情報システムに合わせる必要がありました。

XML コンピューティングでは、人間の知識に近い形の文書をトランザクション処理するため、企業の競争力の源泉となる各社固有の業務の仕組みをそのまま情報システムに反映できます。XML コンピューティングでは、業務を情報システムにあわせて変更するのではなく、業務にあわせて情報システムを構築できます。



ここまででリレーショナルデータベースとグループウェアサーバのそれぞれの特性を述べました。これらの特性から、非定型データを演算加工処理できるちょうどよいソリューションが現状では存在しないことがわかります。弊社の提案する XML コンピューティングでは、非定型データを演算加工処理するためのデータベースマネジメントシステムである "TagStore" を中核とした技術基盤を提供します。TagStore は、ワークフローシステムのよう半構造化文書を扱うシステムの信頼性を向上させ、性能面の問題を解消します。

トランザクション処理について

トランザクション処理 (TP: Transaction Processing) とは、データベースに対する操作をトランザクションという単位でまとめることによって、性能と信頼性を向上させる技術の総称です。特に金銭の授受を伴う取り引きをコンピュータ上で管理する目的で開発され、銀行のオンラインシステムなどで古くから利用されてきました。では、文書をトランザクション処理することのメリットは何でしょうか？それはやはり信頼性の確保です。裏返して言うところの耐性の確保です。

例として、サーバ上で稼働している立替交通費清算書のワークフローシステムについて考えてみましょう。申請者の上司が申請書の承認ボタンを押して自動的に経理の係りに転送される場面ではサーバはどのような処理をしているでしょうか？申請書の転送は内部的に見ると上司の「承認待ち一覧」から申請書を削除し、経理係りの「処理待ち一覧」に追加するという処理になります。ここで、二つの「一覧」を同時に更新しなければならない点に注意してください。このロジックが非トランザクション処理で実装されている場合、処理中の障害発生によって、片方だけの「一覧」が更新される可能性があります。たとえば、上司の「承認待ち一覧」からは削除されて経理の係りの「処理待ち一覧」には追加されていないという状態が発生しうるので、つまり申請書がなくなってしまうわけです。

扱うものが文書であってもデータ間の矛盾は許されません。TagStore を使って、ロジックをトランザクション処理で実装することにより、このような矛盾を防ぐことができます。



<?XML-DOCUMENT-DBMS>
<TAG/><STORE/>

<PRODUCT-BY href="http://www.procube.info">
PROCUBE CO.,LTD.</PRODUCT-BY>
<?INSTALL VERSION = " _ _ _ " MEDIUM="CD-ROM">

・ Achieve Agility ・

～ 激動する環境の変化に機敏に対応するために ～



XML コンピューティングに必要なもの

さて、XML コンピューティングを行うために必要なものは何でしょうか？

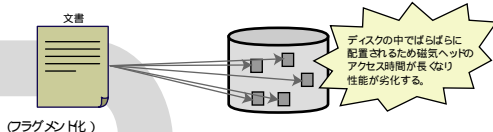
前述したとおり、XML コンピューティングでは、文書トランザクションを扱える必要があります。TagStore では、文書トランザクションを扱えるようにするために従来のリレーショナルデータベースのトランザクション処理機能とグループウェアサーバの文書処理機能を融合させ、高い信頼性と性能を両立させました。

ここでは、その技術要素についてみていきたいと思います。

追記型データベース

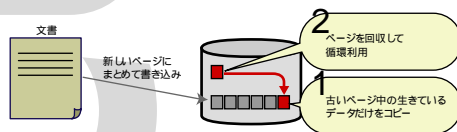
従来のリレーショナルデータベースでは、空き領域を即時に再利用するため、文書のようにサイズがまちまちなデータの更新を繰り返すとディスク上でばらばらに配置されてアクセス性能が劣化するフラグメンテーションという現象が起きます。

たとえば、100 バイトの空き領域が 10 個あるところに 1000 バイトのデータが追加されると、そのデータが 10 個の領域に分割されて格納されます。次に文書を参照すると、この 10 個の領域を拾い集める処理となり、ハードディスクのシーク待ちが多発し、アクセスに時間がかかることとなります。



(フラグメンテーション)

TagStore では、追記型の実装を採用することにより、このフラグメンテーションを回避します。追記型の実装では、データが更新時には新しいページに追記で書き込み、非同期にガーベージコレクションして、散在する空き領域を回収します。



(TagStoreの空き領域回収の仕組み)

トランザクション処理で障害に対応

TagStore では、システム障害発生時もトランザクションの一貫性を保持して自動復旧します。つまり、ログファイルの内容を再実行し、システム障害発生直前の状態に復旧します。このとき、コミットが行われていないトランザクションは自動的にアボートし、その更新内容は廃棄されます。また、データベースのサービス実行中は定期的にスナップショットを記録しており、システム障害発生からの復旧時は、最後のスナップショット以降のログのみ読み出すため、短時間での復旧が可能です。

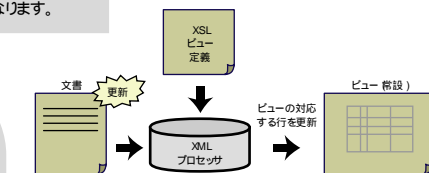
さらに 2 フェーズコミットにより、RDBMS や MQ など他の回復可能リソースとの一貫性を維持できます。

柔軟で高度な文書抽出

ワークフローシステムのように文書を扱うシステムでは柔軟で高度な文書一覧を表示する必要があります。また、エンドユーザーは頻繁に文書一覧を参照するため、これを SQL 文の問い合わせなどによって毎回動的に検索すると性能が確保できないという問題が生じます。

そこで TagStore では、文書一覧を参照時に検索するのではなく、更新時に作成しておくことにより、この問題を回避します。コストの高い文書一覧の更新を文書の更新時にあ事前に行っておくことでアクセス時の性能を確保します。

TagStore では、文書一覧をビューと呼び、その抽出方法と形式を定義したものをビュー定義と呼びます。ビューの定義には W3C 標準の XSL を使用しているため、独自言語を習得する必要はありません。これにより、XML で表現された文書から XPath 式による高度で柔軟な抽出が可能となります。



(ビュー生成の仕組み)

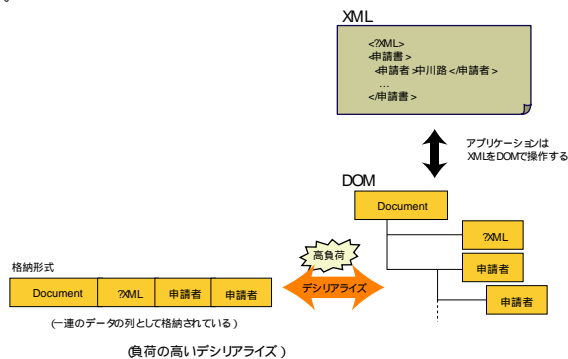
高速 DOM API

XML ではタグを付けることによって文書を構造化しますが、ちょっとした文書でも一般的に多くの構造を持っており、タグの数は数百から場合によっては数千という数になります。このように XML のノードの数が多いたが XML を利用する場合の性能上の落とし穴になっています。ここでは、その性能上の注意点と TagStore がいかにそれを回避するかについて説明します。

デシリアライズの負荷が高い

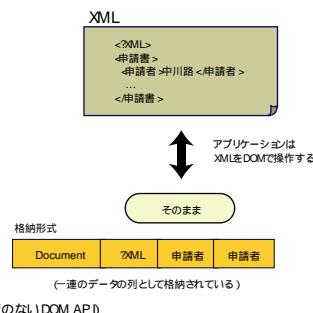
一般的には XML をアプリケーションから処理する場合、DOM と呼ばれる AP を利用しますが、通常の DOM AP は XML のノードをメモリ上に展開します。XML はネットワークやファイル上では一連のデータ列として格納されていますが、DOM AP では XML のノードごとにメモリブロックを確保し、それらをポインタで連結します。この処理をデシリアライズといいますが、デシリアライズ処理では、XML のノード毎に展開処理が行われるため、ノード数に比例して CPU を消費することになります。

大量のトランザクションを処理するサーバ上で、このようなデシリアライズ処理が頻繁に実行されると、CPU 資源が不足し、結果的にシステム全体の性能を劣化させる原因となります。



TagStoreの DOM API

TagStore の DOM AP では、負荷を避けるためにデシリアライズ処理を一切行わず、データ列のまま処理する方式を採用しています。DOM のハンドレはメモリブロックへのポインタではなく、データの列内のオフセット位置を指しています。



まとめ

ここまで述べてきたように次世代 XML コンピューティングによって情報システムはもっと簡単に、もっと速く、もっと確実になります。そしてそれを実現するためには、XML 文書 DBMS TagStore が最適であることがご理解いただけたことと思います。

company profile



株式会社プロキューブ
神戸市中央区北長狭通 4-9-23 K4元町ビル 〒650-0012
telephone.078-393-1334 facsimile.078-393-1344 info@procube.info
http://www.procube.info

telephone.078-393-1334