

平成16年度
研究開発成果報告書

多次元ナレッジマネジメントを可能とする
高度ペタバイトXMLストレージの研究開発

委託先：(株)メディアフュージョン

平成17年5月

情報通信研究機構

平成16年度 研究開発成果報告書

「多次元ナレッジマネジメントを可能とする高度ペタバイトXMLストレージの研究開発」

目次

1	研究開発課題の背景.....	2
2	研究開発の全体計画	
2-1	研究開発課題の概要	4
2-2	研究開発目標	5
2-2-1	最終目標	5
2-2-2	中間目標	6
2-3	研究開発の年度別計画	8
3	研究開発体制	
3-1	研究開発実施体制	9
4	研究開発実施状況	
4-1	XMLに対する統合的問い合わせ操作言語の研究開発	10
4-1-1	序論、サブテーマの内容、研究開発全体からみた位置づけ	10
4-1-2	実施状況	11
4-1-3	まとめ・今後の課題	13
4-2	多次元ナレッジマネジメントを実現するストレージの 動的な拡張技術の研究開発	13
4-2-1	序論、サブテーマの内容、研究開発全体からみた位置づけ	13
4-2-2	実施状況	14
4-2-3	まとめ・今後の課題	15
4-3	総括	15
5	参考資料・参考文献	
5-1	研究発表・講演等一覧	16

1 研究開発課題の背景

社会状況から見る背景

IT社会の進展により、コンピュータシステムが扱うデータのサイズはテラバイト（＝10の12乗バイト）級に達しつつある現状である。今後も、データのサイズは必然的に大きくなり、数年以内にはテラバイト級を超えて、ペタバイト（10の15乗バイト）級が現実のものとして視野に入ってくる。ペタバイト級のデータの例としては、次のようなものが挙げられる。

- ・3次元医用画像（サブミリメートル単位）：1万人分
- ・高精細ビデオ（ハイビジョン画質，1本2時間）：6万人分
- ・ヒトゲノムの塩基配列データ（30億塩基対）：100万人分
- ・無線装置（ICタグ，ICカード，携帯電話からの位置信号）：100万台×10日分

数年前までは、ペタバイト級のデータを扱うことなど全くの夢物語であった。しかし、メインメモリの低価格化（2004年4月現在、1Mバイトあたり20円）・ハードディスクの低価格化（2004年4月現在、1Gバイトあたり60円）の進展によって、ペタバイト級のハードディスクとテラバイト級のメインメモリが、数億円程度のコストで入手可能となってきた。メインメモリ及びハードディスクは、おおよそ1年半で半分に値下がり続けている状況（ムーアの法則）であり、この10年以内には、必ずペタバイト級が現実のものになると見込まれる。一方、64ビットアーキテクチャ（64ビットのアドレス幅）・64ビットOS（Linux, Windows, Solaris など）の普及が急速に進んでいる現状である。64ビット版のWindowsは、現在ベータ版が、インターネットで配布されている状況である。従来の32ビットのシステムでは、扱える実メモリのサイズ・ファイルサイズともに32ビットの壁（4ギガバイト）があったのだが、64ビットのシステムでは、遥かに大きな実メモリとファイルが扱え、ペタバイト級のデータを扱うのに必要なハードウェアとオペレーティングシステムは整いつつある。

ペタバイト級データシステムの必要性- 産業社会への貢献-

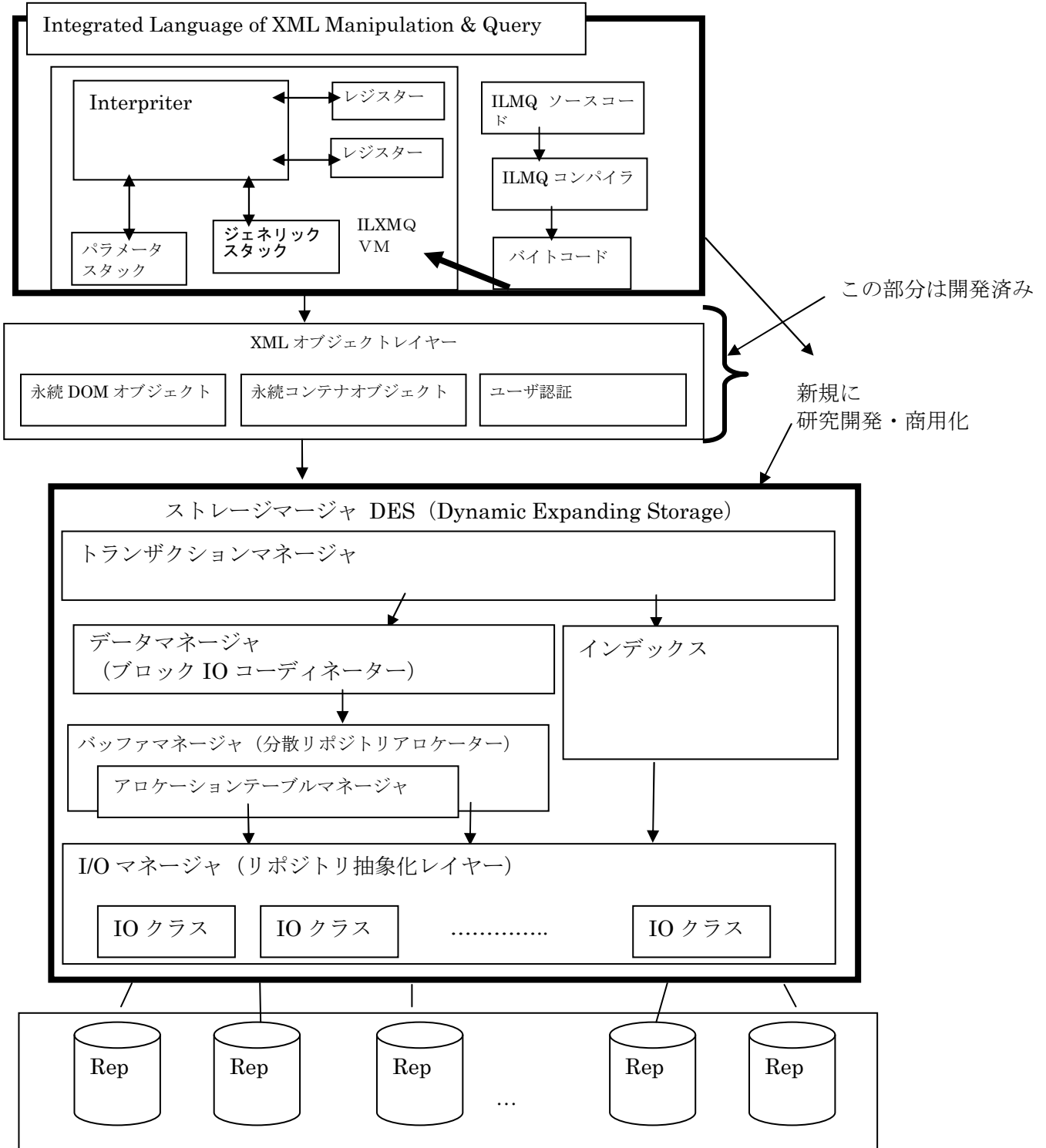
ペタバイト級のデータを扱えるシステムの登場によって、新たなビジネスチャンス・高度な行政サービス等が開拓できる。例えば、医療分野では、患者の電子カルテのデータベース作りがすでに各所で始まっているが、ベッド数数百にもなる大病院では、1日のカルテ作成数は1000以上にも達するし、各カルテは、測定装置（脳波，心電図，レントゲン，3次元医療画像など）のデータを含み、数十メガバイトから数十ギガバイトに達する。法律によって、カルテの10年間保存が義務付けられているから、管理すべきカルテ数は、数百万件に達し、データの総数はテラバイト級に達する。しかし、テラバイト級のデータを扱えない現状では、こうしたカルテのデータの多くの電子化を断念せざるを得ない。一方、プラントなどの複雑な設備が組み合わさった状況において（工場，発電所など）は、各装置に取り付けられた数百万個以上に達するセンサーから、絶えず収集されるログを蓄積し、「そのようなログが記録されているときに、どのような故障が起きるか」などの分析・解析を行い、安全管理に役立てることが、実際の生産現場で期待されている。現状では、すべてのログのデータサイズはペタバイト級に達し、すべてを蓄積することは無理である。現場担当者の勘によって必要なデータを抜き出すことを行わざるを得ない。

以上のように、ペタバイト級のデータを扱えるコンテンツ管理システムが実現できれば、製造業、流通業、小売業などの産業分野・行政分野・科学技術分野において、これら分野向けの新たなS Iシステムの需要を喚起でき、さらに、多量に蓄積されたデータの山を使って新しいマーケティング・製造管理・研究開発その他新サービスが開拓が可能となり、産業の活性化につながる。以上が、ペタバイト級のデータベースシステム開発と実用化を着想した背景である。

2 研究開発体の全体計画

2-1 研究開発課題の概要

研究開発概念図



2-2 研究開発目標

2-2-1 最終目標（平成18年3月末）

「多次元ナレッジマネジメント実現のための新言語体系の創設及び ペタバイト実現のためのストレージの動的な高度拡張技術の研究開発」

- 1) XML に対して一括した操作・問い合わせが行える新しい言語体系設計・実装し、そのために必要な各項目ごとの設計・実装・テストを行う。（これまでに実現できなかったことへの挑戦）
- 2) XML データベースの大規模化に伴い、ストレージの動的拡張（データベース容量が付則した場合、その容量を追加する機能）を実現する。複数の物理的なりポジトリ（データベースの実体を収めたファイル）を統合し、論理的な一つのリポジトリ(DES Dynamic Expanding Storage)とすることを可能とするシステムを研究開発することで、テラバイトを超えるストレージを扱うことを可能とする。（これまでの技術にはない、最新の技術）

ア) 【サブテーマ1】XML に対する統合的問い合わせ操作言語の研究開発

XML に対するクエリー、ノード操作、更新、トランザクション管理を行える言語 (Integrated Language of XML Manipulation & Query) を設計し、その言語によって XML に対する統合的な操作を実現する。

そのための研究課題として次のような項目があげられる。

統合的問い合わせ操作言語 (Integrated Language of XML Manipulation & Query) の設計・実装

- ・統合的問い合わせ操作言語の設計
- ・同言語を解釈、実行するための仮想マシンの設計・実装
- ・仮想マシンの入力となるバイトコードを生成するバイトコードコンパイラを実装

最終目標

XML データベースに対する操作（クエリー、DOMノード操作・更新、トランザクション）を一括して実行するための言語体系を確立する。

イ) 【サブテーマ2】多次元ナレッジマネジメントを実現するストレージの動的な拡張技術の研究開発

XML データベースの大規模化に伴い、ストレージの動的拡張（データベース容量が不足した場合、その容量を追加する機能）を実現する。

複数の物理的なりポジトリ（データベースの実体を収めたファイル）を統合し、論理的な一つのリポジトリ(DES Dynamic Expanding Storage)とすることを可能とするシステムを研究開発することで、テラバイトを超えるストレージを扱うことを可能とする。その研

究課題としては次のような項目が挙げられる

- 動的拡張を実現するストレージマネージャー (Dynamic Expanding Storage) の設計・実装
(詳細項目)
 - ・トランザクションマネージャー
 - ・データマネージャー (ブロック IO コーディネータ)
 - ・バッファーマネージャー (分散リポジトリアロケータ)
 - ・アロケーションテーブルマネージャー
 - ・IO マネージャー (リポジトリ抽象化レイヤー)

最終目標

ストレージの動的拡張を実現、高速な処理を実現する。
最終目標は今まで出来なかった新しい技術 (ストレージの動的拡張) が実現できるようにする、さらに商用に耐えられる商品とすること。

なおサブテーマ2については、本提案において研究指導者となる九州大学大学院システム情報科学研究院牧乃内教授及び金子助教授の指導を受けながら、研究開発を実施する。

2-2-2 中間目標 (平成18年1月末)

ア) 【サブテーマ1】XML に対する統合的問い合わせ操作言語の研究開発

- 統合的問い合わせ操作言語 (Integrated Language of XML Manipulation & Query) の設計・実装
 - ・統合的問い合わせ操作言語の設計
 - ・同言語を解釈、実行するための仮想マシンの設計・実装

中間目標

XML への統合的な問い合わせを実現するための基本設計を終了し、同言語を解釈・実行するための仮想マシンの設計・実装を完了する。

イ) 【サブテーマ2】多次元ナレッジマネジメントを実現するストレージの動的な拡張技術の研究開発

- 動的拡張を実現するストレージマネージャー (Dynamic Expanding Storage) の設計・実装
(詳細項目)
 - ・トランザクションマネージャー
 - ・データマネージャー (ブロック IO コーディネータ)

- バッファーマネージャー (分散リポジトリアロケーター)
- アロケーションテーブルマネージャー
- IO マネージャー (リポジトリ抽象化レイヤー)

中間目標

ストレージの動的拡張を実現するための基本設計を終了し、上記項目の実装をほぼ完了し、実用化に向けたβ版の配布に備える。

2-3 研究開発の年度別計画

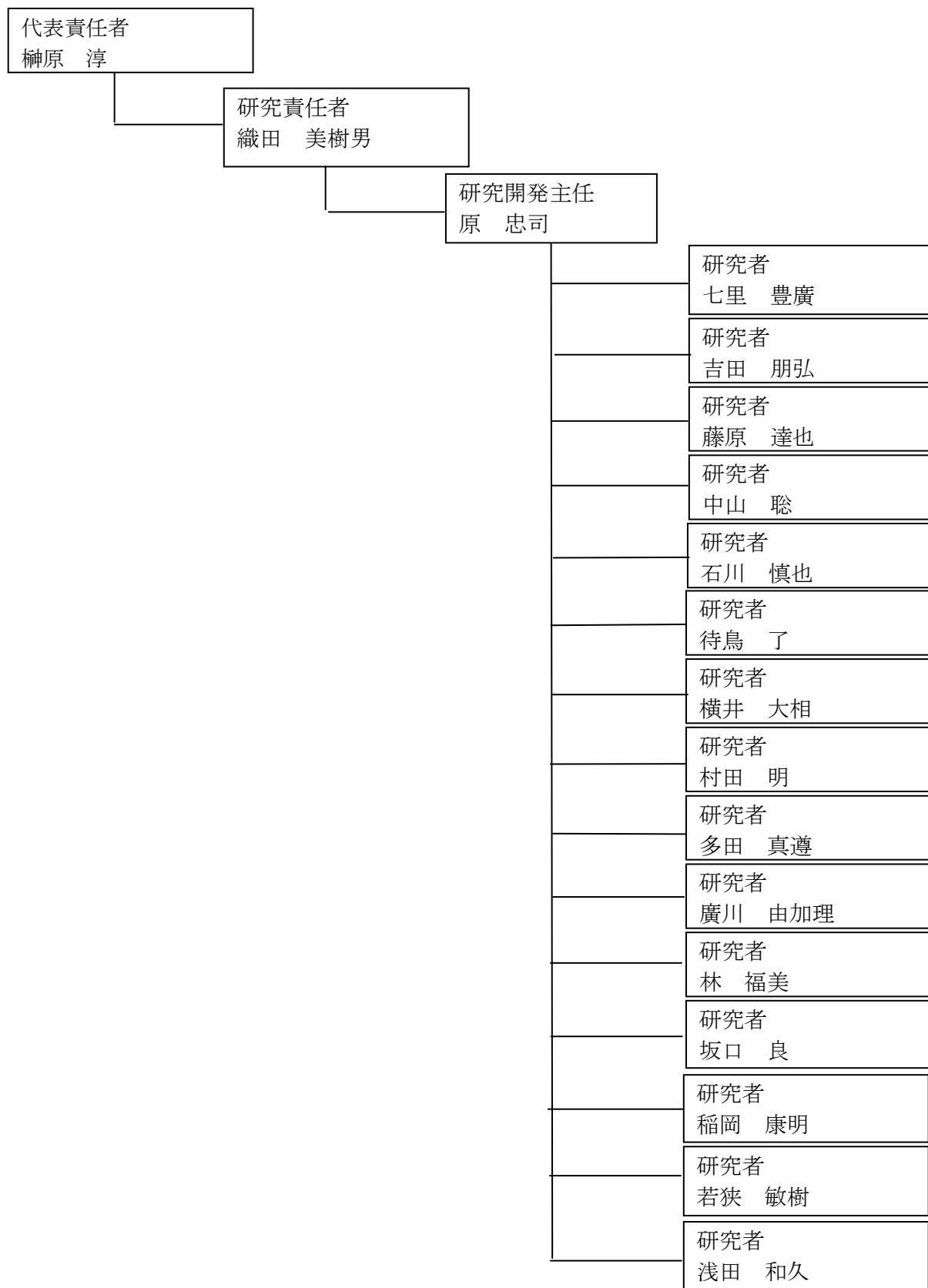
(金額は非公表)

研究開発項目	16年度	17年度	18年度	19年度	20年度	計	備考
ペタバイトを実現する新言語体系の創設とストレージ動的拡張の研究開発							
ア) 新言語体系の (XML に対する総合的問い合わせ操作言語) 研究開発	→	→					
イ) 多次元ナレッジマネジメントを実現するストレージの動的な拡張技術の研究開発	→	→					
テスト用サーバリース費用	→	→					
間接経費							
合計							

- 注) 1 経費は研究開発項目毎に消費税を含めた額で計上。また、間接経費は直接経費の30%を上限として計上 (消費税を含む)。
 2 備考欄に再委託先機関名を記載
 3 年度の欄は研究開発期間の当初年度から記載。

3 研究開発体制

3-1 研究開発実施体制



4 研究開発実施状況

4-1 XMLに対する統合的問い合わせ操作言語の研究開発

4-1-1 序論、サブテーマの内容、研究開発全体から見た位置づけ

序論

現状、規格化されているXMLの操作体系は、検索はXPath,XQuery、操作はDOMノード、というように、複数の規格の集合となっている。

XMLの操作を行う場合には、その複数の規格を理解し、矛盾なく調整を行いながら実装を行う必要がある。また、規格間の基本概念の相違から発生する実行効率の低下を受け入れる必要がある。

そこで、その規格で規定された操作を満たしつつ統合された操作体系を実装するために、XMLの操作に適した操作言語を考案し、実装することとする。

サブテーマの内容

XMLに対する操作を考える上では、更新、参照ともにDOMノードの操作が最小限の単位となる。

よって、現状のW3CにおけるDOMノードの操作規格であるXML-DOM規格を元に、更新、参照に必須の機能を抽出する。

また、検索に関しては、同様に規格化されているXPath、XQueryを元にその検索機能を抽出する。

抽出された機能に対して、それを言語要素として実現するような言語を考慮する。

言語としては、通常用いられる言語としての利便性、学習性から、手続き型言語とし、以下のような特徴を持つものとする。

- ・変数 XPath、Xquery で使用される Literal, Boolean, Numeric, NodeSet 型を扱えるようにする。
- ・式、組み込み関数 XML-DOMにおいて使用されるノード間の移動、XPath,Xquery において使用されるノードフィルタリングの機能を実現する。
- ・制御構造 変数の状態による条件分岐、ループ等を行えるものとする。

研究開発全体から見た位置づけ

研究開発のメインテーマである多次元ナレッジマネジメントの中核であるXMLストレージに対して、統一された操作言語を与える。

これによって、現状、各種言語に対して各々用意されていたプログラミングインターフェイスを簡略化する。

簡略化によるメリットとしては、使用者の学習容易性、可読性、メンテナンス性向上などの変更容易性によるシステム構築のコストダウン、操作粒度が縮小することからくるアプリケーションに対する適合性の向上などがあげられる。

また、統一されたインターフェイスにより、他の言語、システム等との柔軟な連携が容易となる。

4-1-2 実施状況

操作言語は、その言語を実際の命令に翻訳するための言語コンパイラと、そのコンパイラによって生成された命令を実際に行う仮想マシンという、2つのコンポーネントとして開発する。

これにより、言語と実行環境のインターフェイスが簡素化、抽象化され、お互いの依存性を抑えることが可能となる。

また、言語コンパイラによって翻訳されたバイトコードは再利用可能であり、実行時における言語コンパイラの翻訳時間による速度低下を避けることが出来る。

操作言語を以下のように実装し、動作を確認した。

1. 言語コンパイラ

言語仕様

言語仕様の抜粋を以下に示す。

- ・プログラム := 関数定義 | 関数定義 複合式 | 複合式
- ・関数定義 := 関数式 | 関数式 関数定義
- ・複合式 := 式 | 式 複合式
- ・式 := 代入式 | ループ式 | 条件式 | ブロック式
| 出力式 | Self 式 | 値定義 | リターン式;
- ・値定義 := 数値型 | 文字列型 | 論理型 | ノードセット型 ;
- ・演算式 := 部分式 '+' 演算式 | 部分式 '-' 演算式 | '!' 部分式 | 部分式
- ・部分式 := 括弧式 '*' 部分式 | 括弧式 '/' 部分式 | 括弧式;
- ・括弧式 := '(' 演算式 ')' | 変数 | 数値式 | 関数呼び出し
| '(' インクリメント ')' | '(' デクリメント ')'

上記のような操作言語で記述されたテキスト列からバイトコード列を生成する。

バイトコード生成時に発生する冗長性をなるべく減少させるために、簡易な最適化を行っている。

バイトコードの1命令は、以下のような体系となっている。

- ・オペランド 2 バイト
- ・レジスタ参照もしくは直値 (複数可) n バイト

オペランドは、2バイトの値で、命令ごとにユニークな値が定められている。

レジスタ参照もしくは直値の部分には、レジスタの番号を現す値、もしくは実際の値（論理地、数値、文字列等）を記述する。

通常はこのような命令を複数並べた形をとる。命令のシーケンスの最後には、命令の終了を表すターミネータを置く。

- ・ 命令（複数可） n バイト
- ・ ターミネータ 2 バイト

このような構造の命令を順次仮想マシンに送ることで、実際の動作を行うこととする。

2. 仮想マシン

言語コンパイラによって生成されたバイトコードを逐次実行する。

バイトコードを収めるための主記憶を用いることなく、状態遷移によって動作する。仮想マシンは以下のような構造を持っている。

● レジスタ

仮想マシンは複数のレジスタと呼ばれる値を格納、演算するための領域を持つ。レジスタには以下の値を格納することができる。

- ・ 文字列
文字列。1文字は2バイトで表される。Unicode 体系で表すことの可能な文字を収める。他の型への変換が可能。
- ・ 数値
実数。倍制度の浮動小数点型、演算、他の型への変換が可能。
- ・ 論理
論理値。他の型への変換が可能。
- ・ ノードセット
XML-DOMノードのノードセット。複数のノードに対する参照を扱うことが可能である。ノードセット同士の集合演算、他の型への変換が可能。
主に検索におけるノードセットのフィルタリング操作、検索結果の保持に使用される。
- ・ バイト列
任意のバイト列。各々のXML-DOMノードは、任意のバイト列と結びつけることが可能で、その任意のバイト列はこのバイト型として扱うことが出来る。
- ・ バイトコード
バイトコード自体を収める。条件分岐、ループなど、主に制御のために使用する。

- ・ステータス
仮想マシン自体のステータス。

●スタック

値を先入れ先出し方式で保持する。以下の2つのスタックがある。

- ・パラメータスタック
パラメータの授受に使用する。
- ・汎用スタック
一般的なアルゴリズムを実現するために使用する。

レジスタ、スタックに収められた値は、XMLストレージとのインターフェイスともなっており、その変数に対する演算、操作がすなわちXMLストレージへのアクセスとなる。

4-1-3 まとめ・今後の課題

まとめ

言語インターフェイスからXMLストレージに対する各種操作を一元的に行うという、当初の目的は達成されたと思われる。

今後の課題

言語仕様については、幅広い応用分野での使用を視野に、XMLストレージ外部との連携機能を充実させる等の拡張を行う。

また、有効と思われる言語概念等が新たに現れた場合、その概念を取り込みたい。それに伴う仮想マシンの機能向上、速度向上、柔軟性の確保のためのアーキテクチャの見直し等を行う。

4-2 多次元ナレッジマネジメントを実現するストレージの動的な拡張技術の研究開発

4-2-1 序論、サブテーマの内容、研究開発全体から見た位置づけ

序論

多次元ナレッジマネジメントの中核であるXMLストレージにおいては、主に速度的な問題から、そのストレージ容量を事前に設定し、後日の変更は不可能であった。本研究開発は、速度要求に対しては現状にとどめ、その制限のみを緩和する手法を探ることとした。

サブテーマの内容

XMLストレージは、その扱う情報を納めるファイルと密接な関係にある。ファイル上の位置を“参照”と呼び、XMLストレージ上の原子的な情報は全てその参照によって連結することで構造をあらわしている。

ストレージ容量の変更にはそのファイルの物理的構造の変更が不可欠となるが、構造記述には多分に物理的な側面のある“ファイル上の位置”そのものを使用しているため、原子的な情報のファイル上での位置を変更等は事実上、不可能に近い。

これを可能とするためには、ストレージ上の現状の参照、構造等を全く変化させることなく容量追加に伴う構造の変化のみを行う必要がある。

本研究では、そのような変更を可能とするための抜本的なファイル構造の変更について考慮する。

研究開発全体から見た位置づけ

XMLストレージを運用開始後に拡張可能とすることで、システム全体の可用性を高めることを主眼とする。このために行うXMLストレージの基本構造の変更に伴うシステム全体の変更は最小限とする。

4-2-2 実施状況

XMLストレージを構成するファイルは、構造を異にする複数の領域からなる。各々の領域は、ビットマップ、ブロック、インデックス、ハッシュテーブルと呼ばれ、独自の構造を持つ。

各々の構造は、一つの物理的ファイル上にその順で配置されている。

上記のうち、領域拡張に対して構造を変化させる必要のある部分は、ビットマップ、ブロック、インデックスの領域である。

各々の領域について、その構造から考慮することのできる領域拡張の手法を実装する。

●ビットマップ領域

ビットマップ領域は、後述するブロックの使用状況を管理するための構造である。ビットマップ領域全体は、複数のビットマップブロックの集合を3重の木構造とすることで構成されている。

このような木構造の利用により、ひとつのビットマップブロックが保持するビット数の3乗までのブロックを管理可能となる。

この構造の領域拡張における問題は、ビットマップブロック間の構造連携を演算から求めていることに起因する。

詳細には、領域が拡張された場合、ビットマップも同様に拡張する必要があるが、ビットマップはその木構造の階層ごとに連続しているという前提から演算を行うため、木構造の各々の階層ごとのビットマップブロック数を変更できない、ということになる。

よって、領域拡張後の最大サイズを事前に指定し、それに準じた構造としておき、後の拡張に備えるという手法を用いた。

●ブロック領域

ブロック領域は、原子的な情報を単位として扱う領域で、実際の情報は全てここに

収められる。

ブロック領域は大きさの一定なブロックを複数収めるが、常にその先頭から使用されてゆく。よって、その使用されている最終のブロック以降については、構造的には自由に拡張が可能であるが、実際にはブロック領域以降の構造であるインデックス領域、ハッシュテーブル領域の再配置を行う必要がある。

上述した理由により、再配置を行わず拡張を行うため、インデックス領域、ハッシュテーブル領域については別ファイルを用い、ブロック領域の終端=ファイルの終端とすることで、構造変化を伴わない自由な拡張を可能とした。

●インデックス領域

インデックス領域は、参照を基本単位とし、その参照を複数収める構造となっている。

インデックス領域の大きさはブロック領域の大きさと比例していることが求められる。インデックス領域は比較的単純な構造であるため、上述のブロック領域の拡張と同様の手法、すなわち物理ファイルの終端=論理構造の終端とすることで、拡張可能となる。

上記手法をすべて実装することによって、現状、XMLストレージに対して、事前に定められた大きさまでの任意の単位での拡張が可能となっている。

4-2-3 まとめ・今後の課題

まとめ

元来、拡張に対する配慮を行っていない構造に対して、その構造の変化を行わず拡張のみを行うということは非常に困難だと考えられるが、今回はそれに起因する性能の低下などを起こさない範囲でその実装が完了した。

今後の課題

ビットマップ領域のみ、構築時にその最大サイズを指定する必要があり、それに起因してシステム全体がやや柔軟性、可用性に欠ける。

ビットマップ領域の木構造をその階層ごとに別ファイルとし、物理ファイルの終端=論理構造の終端の原則を適用することを可能とすれば、その制限を撤廃することが可能となる。

その場合の性能等に対する影響を把握し、今後考慮してみたい。

4-3 総括

XML に対する統合的問い合わせ操作言語の研究開発およびストレージの動的な拡張技術の研究開発によって、多次元ナレッジマネジメントの運用面に置ける柔軟性が向上し、より一層システムとしての理想に近づいたと思われる。

今後も、上述した今後の課題等について研究を進めることで、より一層の機能の充実、一般性の向上を図りたい。

5 参考資料・参考文献

5-1 研究発表・講演等一覧

本研究開発に関する研究発表・講演等はありません。