非XMLデータに対するXPath検索のための ラッパーのインターフェイスの設計

渡谷 賢治 (310124)

School of 情報科学研究科,
Japan Advanced Institute of Science and Technology

2005年2月10日

Keywords: XML, 問合せ処理, 最適化.

近年，二次記憶の低価格化により，個人の計算機上やインターネット上の計算機上に様々なデータ形式の膨大な量のデータが保存されるようになっている．そのため，これらの雑多なデータ形式のデータ群から一元的に必要なデータを検索する機能に対する需要が高まっている．現在は，HTML，PDF，Microsoft PowerPointなどのデータ形式に対して，一元的にキーワードによる検索を行うシステムが実現されているが，データが膨大になり興味のあるデータの絞り込みが困難になるにつれて，単なるキーワードによる検索だけでなく，各データ中の構造を利用した「構造検索」が行えることが重要になっている．

この問題に対するアプローチとして，現在，汎用の標準データ形式としてXMLを用いることが提案されている．例えば，自分のアドレス帳，スケジュール表，メイルボックス等をXML形式でで保存しておけば，これらのデータに対してXML用の構造検索のためのシンプルな言語であるXPathで一括して検索を行うなどの処理が可能となる．しかし，各アプリケーション用のデータ形式や画像等のデータ種別毎の標準データ形式も依然として多く使用されており，データ量や性能上の問題から，今後も全てのデータがXML形式になるとは考えにくい．

そこで，本研究では，これらの各種のデータ形式のデータに対して，XPathを用いて統一的に検索を行うためのシステムを提案する．このシステムは，各データ形式毎に一つ用意され，各データ形式上である共通インターフェイスとなる操作群を実現するラッパーと，各ラッパーが提供する共通の操作群を使ってXPathの評価を行う共通モジュールから構成される．

ここで，各ラッパーと共通モジュールの間のインターフェイスの設計が重要になる．ラッパーが共通モジュールに提供するインターフェイスの操作群が低レベル過ぎると，各ラッパーの作成は容易になるものの，与えられたXPathの実行の際に，各データ形式毎に依存した様々な最適化を行うことが困難になるため，実行効率が悪くなる可能性がある．一

方，ラッパーが提供する操作群が逆に高レベル過ぎると，各ラッパー中に，データ形式に依存した最適化のロジックを組み込むことは容易になるものの，各データ形式毎にラッパーを作成するプログラミングのコストが高くなり，より多くのデータ形式をサポートするという上では障害となる．

以上の観察から，本論文では，どのような操作群が XPath の効率的評価に必要か考察し，ラッパーのプログラミングが比較的容易で，かつ，高効率の XPath 実行効率を達成できるラッパーのインターフェイスを設計する．また，本研究で提案するインターフェイスを用いる場合と，より低レベル，あるいはより高レベルのインターフェイスを用いる場合とについて，XPath の実行効率の性能比較と，ラッパーのプログラミング量の比較を行い，提案インターフェイスの有用性について検証する．

より具体的には，インターフェイスの方式として，操作群のレベルが高い物から低い物まで，以下の 4 方式を考案した．

方式 1: 共通モジュールは XPath の処理は一切行わなず，単に，対象データに応じたラッパーを起動して XPath 式を渡す．ラッパーは XPath 式を評価し，その結果を共通モジュールに返す．この方式では，各ラッパーが，データ形式には非依存で全ラッパーに共通の XPath 式の評価のロジックの部分を含むことになるため，ラッパーのコーディング量が増え，作成コストが非常に高くなる．

方式 2: 共通モジュールは適切なラッパーを起動するとともに，XPath 式を，simple path 式とよばれる，より単純なクラスの XPath 式の集合に分割する．そして，各 simple path 式はそれぞれファイルポインタの移動命令に変換されてラッパーに渡される．ラッパーは渡された命令に従いファイルポインタの移動（あるいは移動先のデータの読み出し）を行い，操作が完了すると共通モジュールに通知し，次の命令を待つ．この場合，一つの操作命令が，XML 木上での単純なナビゲーションの複数ステップをまとめたものに相当する．

方式 3: 共通モジュールは適切なラッパーを起動するとともに，XPath 式を解析し，その結果に基づいて，ラッパーにファイルポインタの移動命令を出す．一つの移動命令が XPath 式中のステップと呼ばれる単位の一つ一つに対応する．ラッパーは渡された命令に従いファイルポインタの移動（あるいはさらに移動先のデータの読み出し）を行い，操作が完了すると共通モジュールに通知し，次の命令を待つ．この場合，一つの操作命令が，XML 木上での単純なナビゲーションの一ステップに相当する．

方式 4: 共通モジュールは適切なラッパーを起動するとともに，XPath を評価する．ラッパーは，与えられた XPath 式依存の処理は一切行わず，常にファイルの先頭から最後まで全体を処理し，データが仮想的な XML データであるようなビューを共通モジュールに提供する．より具体的にば，ラッパーのインターフェイスとして，XML データへのアクセスの規格の一つである SAX インターフェイスを採用し，ラッパーは XML データを表す SAX イベント列を生成し共通モジュールに返す．ここで SAX イベント列とはエレメントの開始タグや終了タグなどの出現を表すイベントの列である．この方式では常にデータの先頭から最後までを処理するため，巨大なデータから，その最初の方の一部のみを取り

出すような検索においても，データ全体を読み出して XML に変換する処理をすることになり，XPath の処理効率は大幅に低下する．

以上の方式のうち，これまでに方式 1，3，4 について実装し，実験を行った．実験結果を比較すると，方式 3 が方式 1,4 よりも高速かつメモリ使用量も少なくなった．このことから，非 XML データに対し XPath 式で検索をするシステムに方式 3 を使用することが，方式 1 や方式 4 よりも適切であることが確認できた．

方式 1,3,4 のそれぞれのコード量は，方式 1 が一番少なく，次に 4,3 となった．この理由は，方式 1 は共通モジュールの機能自体が少なく，ラッパーも既に存在しているライブラリを使用することができたためであり，方式 4 も XPath 式の処理に既存のライブラリを使用することができたからだ．しかし，方式 3 は共通モジュールやラッパーなどを全て１から作成したため，コード量が多くなってしまった．