社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－DD－57 (2)
2006／9／29

RDBを用いたXML DBの実装

平 林 文 雄†

XMLは柔軟性が高く応用範囲も広い反面，歴史が浅いため実績が少なく，開発者も不慣れなため開発効率や信頼性の面で不安が多い．リレーショナル・データベースは技術の蓄積があり不安要素が少ない反面，柔軟性に欠ける．双方の長所を活かし短所を補うために，データベースはRDBを使い，インタフェースにXMLを用いる実装を試作し，実用的な実効速度で動作する事を確認した．

Implementation of XML DB with RDB

FUMIO HIRABAYASHI†

XML has excellent flexibility and can be applied to the large range. But XML is a new technology. XML has little results and few Developer. XML has uneasiness of Development efficiency and Reliability. Relational Database has excellent results and Reliability. But RDB has little flexibility. XML is used for the interface, and RDB used for the database, It supplements mutually. I implemented Prototype.

1. はじめに

XML[1)2)]が普及しつつあるが，アプリケーションを開発する際にはデータベースを必要とする場合が多い．その際，データベースは二つの方法がよく使われる．一つはXMLデータベース（以降XMLDBと略す）を使う方法であり，もう一つはリレーショナル・データベース（以後RDBと略す）にSQL[3)]を組み合わせる方法である．以下にそれぞれの長所と短所を挙げる．

- XMLDB
  - 長所
    - XMLは明確で標準化された文法を持つので，データ交換の形式として優れている．
    - 項目の追加や繰り返しの表現が容易であり，仕様変更に対応しやすく広い範囲に応用できる．
  - 短所
    - XMLDBの実装が少なく，選択の余地が狭い．
    - 歴史が浅いため技術やノウハウの蓄積が少く，信頼性や実効速度の面で不安が残る．
    - アプリケーション，データベースともに開発・運用の経験者が少なく，コストや開発期間・実効速度の正確な見積もりが困難である．
- RDB
  - 長所
    - 市販・オープンソース共に多数の実装があり，選択の幅が広い．
    - 既に多くの実績と技術とノウハウの蓄積があり，信頼性や実効速度の面で不安要素が少ない．
    - アプリケーション・データベース共に開発・運用の経験者が多く，コスト・開発期間・実効速度などを比較的正確に見積もりができる．
  - 短所
    - アプリケーションごとにデータ構造が異なるため，データ交換の形式として不便である．
    - 項目の追加や削除が困難であり，予め充分にデータ構造を設計しなければならず，仕様変更のコストが大きい．

つまり，XMLおよびXMLDBは柔軟性に優れている反面，実績と信頼性に乏しく開発や運用の経験者の確保が難しい．RDBは柔軟性に欠けるが，多くの実績

† 凸版印刷
Toppan Printing

と信頼性があり経験者の確保も容易である．

そこで，XML と RDB 相互の長所を活かし短所を補うため，インタフェースに柔軟性の高い XML を用い，データベースに RDB を用いる実装を試作した．これにより，RDB の運用経験と信頼性を活かしつつ，XML の柔軟性を活用できる．

2 章で仕様に加えた制限を述べる．3 章でデータ構造を示す．4 章で手続きの概要を述べる．5 章では当実装のインタフェース（問い合わせの形式）を示す．6 章でデータ構造と問い合わせの例を示す．7 章で実装および計測したプラットフォームを示す．8 章で計測結果を示す．

2. 仕　　様

この実装では，開発期間を短縮するため仕様にいくつかの制限を加えた．

2.1 XML 文書の形式

(1) 1 ドキュメント 1 件であり，1 ドキュメント複数件は認めない．
(2) 要素のみを扱い，属性は扱わない．
(3) 名前空間は扱わない．
(4) テキストデータのみであり，バイナリデータは扱わない．
(5) 各ノードは以下 2 種のいずれかである．テキスト要素と子ノードが混在してはならない．
　(a) テキスト要素を 1 つだけ持つ．
　(b) 0 個以上の子ノードを持つ．

2.2 機　　能

(1) 挿入と検索のみであり，削除と更新は実装していない．
(2) バックアップとリストアは実装していない．

3. データ構造

以下の 3 要素からなる．

- タグ表：タグの XML パスを管理する．
- 値表：値表はテキスト・ノードを管理する．
- ファイル集：XML 文書を圧縮して保持する．

以降で各要素の詳細を述べる．

3.1 タ グ 表

各ノードの XML パスと名前（表記）を管理する．以下 3 項目を持つ．

- 識別：主キー．行の識別．整数．
- パス：XML パス．文字列．索引あり．省略不可．"/html/body/h1" 等．
- 表記：タグの表記．文字列．全文索引あり．"大見出し" 等．

3.2 値　　表

テキスト・ノードの内容と，検索用の全文索引を管理する．以下 3 項目を持つ．

- 文書識別：XML 文書の識別．整数．索引あり．ファイル集の文書識別を参照する．
- タグ識別：タグ表の識別を参照する．整数．索引あり．省略不可．
- 値：文字列．全文索引あり．テキスト要素の内容．

3.3 ファイル集

各 XML 文書自体を zip で圧縮し，固有の文書識別をつけて 1 つのファイル・システムに保存する．文書識別は値表の文書識別項目から参照される．今回は以下の命名規約でファイル名を設定した．

- 文書識別（値表の文書識別項目）を 32bit の整数とする．
- 上記を 4 個の 8bit に分け，各要素を 2 桁の 16 進 ASCII 文字で表現する．
- 1〜3 個目の 8bit をディレクトリ名とし，4 個目の 8bit をファイル名とする．

例えば文書識別が 16 進で 01234567 なら，ファイル名は "〜/01/23/45/67" となる．

4. 手続きの概要

ここでは挿入と検索の処理の概要を述べる．

4.1 挿 入 処 理

ひとつの XML 文書を挿入する手続きは以下の手順である．

(1) XML 文書から，全てのテキスト・ノードを抽出する．
(2) 各テキスト・ノードを RDB の値表に挿入する．タグ識別はタグ表から得る．該当する XML パスがタグ表にない場合はタグ表に新規登録する．
(3) XML 文書を圧縮してファイル集に登録する．

4.2 検 索 処 理

XML 文書を検索する手続きは以下の手順である．

(1) 次章に述べる問い合わせ形式を検索の SQL (SELECT 文) に変換する．
(2) 上記で作った SQL 文を RDB に発行し，文書識別の一覧を得る．
(3) 上記で得た一覧にある全文書をファイル集から取り出し，圧縮を復元する．

5. 問い合わせ

問い合わせは XML 形式とした．タグ名・属性名・演算子共に大文字と小文字を区別しない．

5.1 タ グ 名

EXPRESSION または EXPR．どちらも同じ意味である．

5.2 属　　性

OPERATOR，VALUE，PATH，CONS の 4 種類がある．

5.2.1 属性 OPERATOR または OPR

演算子．XML パス・定数・比較演算子・論理演算子の 4 種類がある．

- P，PATH：XML パスを示す．
  テキスト要素または属性 VALUE で値を記述する．
- C，CONS，CONSTANT：定数を示す．
  要素または属性 VALUE で値を記述する．
- 比較演算子：以下 7 種類がある．それぞれ子要素を 2 個（例えばパスと定数）指定する．
  なお，= と EQ は同じ意味である．
  - =，EQ：等しい
  - !=，NE，<>，<=>：等しくない
  - <，LT：〜より小さい
  - <=，LE：以下
  - >，GT：〜より大きい
  - >=，GE：以上
  - LIKE：〜を含む
- 論理演算子：以下 2 種類がある．子要素（EXPRESSSION タグ）を 0 個以上指定する．
  - AND：かつ
  - OR：または

5.2.2 属性 VALUE

属性 P，PATH または 属性 C，CONS，CONSTANT を指定した際の値を指定する．

5.2.3 属性 P または PATH

XML パスを指定する．以下の例はいずれも XML パス /usr/bin を示す．

- `<expression operator="path">`
  `/usr/bin</expression>`
- `<expr opr="p" value="/usr/bin"/>`
- `<expr p="/usr/bin"/>`

5.2.4 属性 C, CONS, CONSTANT

定数を指定する．以下の例はいずれも定数 "8732_000000" を示す．

- `<expression operator="constant">`
  `8732_000000</expression>`
- `<expr c="8732_000000"/>`
- `<expr opr="c" value="8732_000000"/>`

6. 例

6.1 データの例

（図 1）の 2 件の XML 文書を保存した場合を示す．

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
    <type>書籍</type>
    <list>
        <name>本 1 </name>
        <name>本 2 </name>
    </list>
</catalog>
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
    <type>C D</type>
    <list>
        <name>C D 1 </name>
        <name>C D 2 </name>
    </list>
</catalog>
```

図 1　サンプルデータ
Fig. 1　Sample Data

表 1　タグ表
Table 1　Tag Table

識別	XML パス	表記
1	/calalog	カタログ
2	/catalog/type	種類
3	/catalog/list	一覧
4	/catalog/list/name	名前

表 2　値表
Table 2　Value Table

文書識別	タグ識別	値
1	2	書籍
1	4	本1
1	4	本2
2	2	C D
2	4	C D 1
2	4	C D 2

```
〜/00/00/00/01
〜/00/00/00/02
```

図 2　ファイル集
Fig. 2　File Collection

タグ表を（表 1）に，値表を（表 2）に示す．ファイル集は（図 2）の 2 ファイルである．

6.2 問い合わせの例

以下 3 条件を全て満たす文書を抽出する問い合わせを，（図 3）に示す．

- /item/identifier が "8732_000000" 以上である．

- "/item/identifier" が "8732_001000" 以下である.
- "/item/description/descriptionNote" に "_00001" を含む.

(図3) から変換して生成した SELECT 文を (図4) に示す. なお, 体裁を整えるため改行位置と空白を調整している. 実装時の表と項目の名前は以下である.

- タグ表 : tag
  - 識別 : id
  - パス : path
  - 表記 : spell
- 値表 : value
  - 文書識別 : document
  - タグ識別 : tag
  - 値 : value

7. 計測条件

ハードウェア, ソフトウェアの順に実装及び計測に用いた環境を示す.

7.1 ハードウェア

- 本体 : Mac OS X Server Dual 1GHz
- CPU : PowerPC G4 Dual 1GHz
- メモリ : 512MB DDR SDRAM
- HD : RAID 1(ミラーリング, スライス 2, ATA) 60GB × 2
- ファイルシステム : Mac OS 拡張フォーマット (ジャーナリング)

7.2 ソフトウェア

- OS : Mac OS X Server 10.3.9
- RDBMS : mySQL Ver 14.7
- プログラミング言語 : perl v5.8[4)5)]

7.3 テストデータ

全体で約 684MB.

- 全 100,000 件 : 約 684MB
- 1 件のバイト数 : 約 6.84k
- 1 件のタグ数 : 約 150 個
- 1 件のテキストタグ数 : 約 118 個
- 1 件のテキスト以外のタグ数 : 約 32 個

8. 計測結果

8.1 処理時間

挿入処理と検索処理の処理時間を計測した. 処理時間の計測には unix の time コマンドを用いた.

8.1.1 挿入処理

挿入は 1000 件を挿入した場合と 10 万件を挿入した場合を計測した. 1000 件で実時間 51 秒, 10 万件で実時間 5 時間 40 分 (表 3) となった.

表 3 挿入処理の所要時間 (単位 : 秒)
Table 3 Time of insert

件数	real	user	sys
1,000	50.532	42.090	2.24
100,000	20299.66	4365.89	205.01

表 4 検索処理の所要時間 (単位 : 秒)
Table 4 Time of select

	real	user	sys
1,000/ 1,000	1.250	0.800	0.430
10/100,000	3.194	0.350	0.160
100/100,000	3.241	0.350	0.190
1,000/100,000	6.777	0.850	0.770
9,999/100,000	57.292	5.490	7.870

表 5 ディスク容量
Table 5 Disk Space

XML 本体	400MB
value 表の本体	419MB
value 表の索引	649MB
合計	1,468MB

8.1.2 検索処理

検索は全件 1000 件中から 1000 件を抽出した場合と全件 10 万件中から 1 万件を抽出した場合を計測した. 1000 件中 1000 件抽出で実時間 1.3 秒, 10 万件中 1 万件抽出で実時間 57 秒 (表 4) となった.

8.2 空間 (ディスク容量)

全体で約 1.5G 消費した (表 5). 元データ量約 684MB に対し約 215% のディスクを消費した. 索引の消費量が全消費量の 44%と大きい. なお, タグ表の容量は本体が約 7k, 索引が約 16k と少量であるため無視した.

9. おわりに

制限の項で述べた条件はいずれも解決は可能であると考える. 以下に解決案を記す.

9.1 XML 文書の形式

(1) 1 ドキュメント 1 件であり, 1 ドキュメント複数件は認めない : 前処理などで 1 ドキュメント 1 件に分割する.

(2) 要素のみを扱い, 属性は扱わない : 属性は属性名表と属性値表を追加する.

(3) テキストデータのみであり, バイナリデータは扱わない : バイナリ値表を追加する. 形式は値表と同様とする.

(4) テキスト要素と子ノードが混在してはならない : 混在ノード中のテキスト要素は<text>

```
<?xml version="1.0" encoding="UTF-8"?>
<expression operator="and">
    <expression operator="ge">
        <expression operator="path">/item/identifier</expression>
        <expression operator="constant" value="8732_000000"/>
    </expression>
    <expression operator="le">
        <expression path="/item/identifier"/>
        <expression constant="8732_001000"/>
    </expression>
    <expr opr="like">
        <expression p="/item/description/descriptionNote"/>
        <expression c="%\_00001%"/>
    </expr>
</expression>
```

図 3 問い合わせの例
Fig. 3 Sample Query

```
SELECT DISTINCT v01.document
FROM
         tag t02, value v02,
         tag t01, value v01
WHERE
         v01.document =    v02.document
AND      v02.tag      =    t02.id
AND      t02.path     =    '/item/description/descriptionNote'
AND      v01.tag      =    t01.id
AND      t01.path     =    '/item/identifier'
AND (
        v01.value     >=   '8732_000000'
    AND v01.value     <=   '8732_001000'
    AND v02.value     LIKE '%\_00001_'
)
```

図 4 問い合わせの変換結果
Fig. 4 Sample Query of SQL

～</text>等の特殊タグと見なす．例えば（図5）の様なノードは（図6）と見なす．

9.2 機　　能

(1) 挿入と検索のみであり，削除と更新は実装していない：削除の実装は比較的容易である．更新は削除→挿入の２段階に分割する．

(2) バックアップとリストア：RDB のバックアップ機能とリストア機能を使う．

9.3 改善すべき点

以下の点は改善の余地があると思われる．

```
<node>
テキスト要素と
<sometag>子ノードが</sometag>
混在している
</node>
```

図 5 混在ノードの例
Fig. 5 mixed node

9.3.1 ファイル集

多くの RDB は長いテキストを圧縮して格納する機

```
<node>
<text>テキスト要素と</text>
<sometag>子ノードが</sometag>
<text>混在している</text>
</node>
```

図 6　混在ノードを変換した例
Fig. 6　converted mixed node

能がある．この機能を用いて，ファイル集を RDB に格納すれば，ディスク消費量を 10〜20%程度圧縮できるであろう．また，バックアップとリストアも RDB の機能を利用できる．

9.3.2　XML 文書の圧縮

今回の計測に用いたデータは XML のタグ名が長い．XML 文書を保存する際に，XML 文書中のタグ名をタグ表の識別に置き換えると，10%程度の容量を減らせるであろう．

参 考 文 献

1) 中山幹敏，奥井康弘：改訂版 標準 XML 完全解説（上），日本ユニテック (2001).
2) 中山幹敏，奥井康弘：改訂版 標準 XML 完全解説（下），日本ユニテック (2001).
3) 朝井　淳: SQL ポケットリファレンス，技術評論社 (1999).
4) Wall,Larry, Chiristiansen, Tom and Orwant, Jon : Programming Perl Third Edition Volume 1, O'Reilly (2002).（近藤嘉雪 訳：プログラミング Perl 第 3 版 Volume 1，オライリー・ジャパン (2002)）
5) Wall,Larry, Chiristiansen, Tom and Orwant, Jon : Programming Perl Third Edition Volume 2, O'Reilly (2002).（近藤嘉雪 訳：プログラミング Perl 第 3 版 Volume 2，オライリー・ジャパン (2002)）

平林　文雄

昭和 36 年生．昭和 56 年凸版印刷 (株) 入社．主に版下の自動生成に携わる．