ブランチ機能付き版管理に適した木構造のラベリング手法

A New Tree Labeling Scheme for Managing Version Series with Branches

横山 昌平♥ 太田 学♦
片山 薫♦ 石川 博♦

Shohei YOKOYAMA Manabu OHTA
Kaoru KATAYAMA Hiroshi ISHIKAWA

近年，XML はデータ交換の共通フォーマットのみならず，データベースとしての利用機会が増えている．そこで，XML データを関係データベースに効率良く格納する手法の研究が盛んに行われている．またデータの更新を考える際，重要となるのがデータの版管理である．現在 RCS に代表される既存のテキストベース版管理ツールは，一つのバージョンから多数の次バージョンを作る事が出来るブランチ機能を有し，プログラムソースファイルのバージョン管理等に広く使われている．しかしながら，それらのツールはテキストファイルの行を基準に版管理を行っており，XML の文法を理解しない上，関係データベースに格納した XML 文書の版管理を行う事が出来ない．本稿では関係データベースに格納した XML 文書に対しブランチ機能を有する版管理を行う為の版識別子を提案する．ブランチ機能を持つバージョン系列はいわば木構造であり，提案する版識別子は木へのラベリング手法と考える事が出来る．

Recently, XML documents are not only used for data-exchange, but also used as formats for databases. Thus storing XML in relational databases becomes an influential topic in data engineering. When XML data with update is considered, management of the versions is very important. Traditional text document versioning systems, e.g. RCS, support branches allowing several lines of development to occur in parallel. And they are widely used by popular open-source projects like PostgreSQL. However the versioning tools use a line-based method. Accordingly, they cannot recognize XML's grammar and also cannot handle XML data in relational databases. In this paper, we propose a new version labeling method which supports branches and XML data in relational databases. As version series with branches can be viewed as a tree, we explain a new labeling scheme of trees in detail.

1. はじめに

XML文書はとりわけ企業間のデータ交換に用いられていたが，最近はデータベースと連携して利用される機会が増えてきている．それはXMLで扱うデータ量が増大すると共に，データを効率よく管理することが急務となっている為である．XMLデータの管理をする為に関係データベースを用いるシステムが複数提案されているが，我々はXMLデータの格納だけでなく，更新情報を管理すること，即ち版管理が重要と考えた．我々はXML文書の関係データベース格納手法であるSaxophone[1]を提案してきた．現在，このSaxophoneを拡張し，版管理機能を実装する為の研究を行っている．本稿ではその版管理手法で利用する版識別子の構成法を提案する．

インターネット・イントラネットという分散環境でデータ管理が行われている昨今，版管理をする上でブランチ機能を持つことが重要であると考えられる．ブランチ機能とはある一つのバージョンから複数の異なるバージョンを作り出す事で，複数の開発者が平行して作業を行う際に有用である．

ブランチ機能を有する版管理手法にはRCS[2]等がある．これはバージョン間の差分データのみを格納して，版管理を行う方法で，主にテキストドキュメントやプログラムソースコードの版管理に利用されている．RCSにてバージョンを区別する為に使われているrevision番号は幾つかの数字をピリオドで区切った形式で表される．例えば1.2.2.4のようになる．奇数番目にある数字はブランチを表しており，偶数番目の数字はそのブランチ内での特定のrevisionを表している．つまりrevision番号1.2.2.4は，1と言うブランチにあるrevision 1.2から派生するブランチ1.2.2上の4番目のrevisionを意味している．この手法ではブランチの識別子とバージョンの識別子を異なる手法を用いて構成し，なおかつそれを一つのrevision番号として表す為，バージョン系列の任意の版を検索する，特に祖先・子孫関係の検索が非常に難しい．例えばrevision1.2.2.3は1.2.2.4の親であるが，ブランチ1.2.1は1.2.2の親では無い．

我々はブランチを持つバージョン系列が木構造である事に着目し，木構造のラベリング手法を版管理に用いている．本稿では，そのラベリング手法“版識別子”の構成法及び検索手法について述べる．

本稿の構成は以下の通りである．続く2節では提案方式と関連研究について述べる．3節ではブランチ機能を持つ版識別子について説明し，4節では版識別子を利用して版検索を行う手法説明し，実験により有用性を示す．最後に5節で本稿をまとめる．

2. 関連研究

前述したように，提案手法ではバージョン系列を木とみなしている．各バージョンは木のノードに相当し，ノードの識別子を用いてバージョンを区別する．木構造のラベリング手法は多数提案されている．ラベリング手法の代表的なアプローチの一つに半構造データをk-ary treeとみなし，接点に識別子を付ける手法[3]がある．ある接点の子要素がk未満の時は，仮想的にノードが有ると仮定する．この手法では，識別子から木の位置が特定できるが，k-ary treeと大きく構造が異なるデータに対しては，仮想ノードの数も非常に多くなる等，効率の面で問題がある．

この問題に対し，佐藤らはレベル毎にkの値を変化させる手法[4]を提案している．しかしながら，番号の枯渇を防ぐ為に木の高さを規定する必要があり，バージョンの更新が木の高さ方向に進む版管理システムには向かない．

既存の版管理手法の殆どが前バージョンとの差分データのみを保存する形式をとっており，提案する識別子を利用し

♥ 学生会員 東京都立大学工学研究科博士課程
shohei@hikendbs.eei.metro-u.ac.jp
♦ 正会員 東京都立大学工学研究科
{ohta,katayama,ishikawa}@hikendbs.eei.metro-u.ac.jp

た版管理手法も同じように差分データのみを扱う事を目的としている．即ち，あるバージョンを検索する際，祖先の系列全てを検索し差分データを収集する操作が必要となる．よって指定された任意のバージョンから祖先の系列を効率良く探し出せる識別子であれば非常に効率が良い．このような手法としてPrefix Labeling手法[5]がある．この手法では，ある識別子のプレフィックスとなる識別子を持つノードは祖先である，という特徴を持っている．プレフィクスの検索は関係データベースの機能を用いて容易に行うことができるため，Saxophone上で差分データの断片を祖先にわたり集約するのに適していると考えられる．一方で欠点は，skewのある木の末端では識別子が非常に長くなってしまうことである．提案方式ではこのPrefix Labelingのアプローチを基本とした識別子をバージョン検索に利用しているが，木の高さを表す値と組み合わせることによって，また任意のn進数で識別子を構成することによって，それぞれ，木の高さ方向及び幅方向で識別子長の増加を防いでいる．

3. 版識別子の構成法と性質

単純なバージョン系列とブランチのあるバージョン系列の例を図1に示す．この例ではバージョン３に属する文書の版が４つあることを示している．このようにブランチのあるバージョン系列では，従来の単純なバージョン番号での管理は困難である．そこで，バージョン系列を木とみなし，接点に識別子を付け，それを用いて版管理を行う．

この節では，ブランチのあるバージョン系列で任意のブランチに属する任意のバージョンを一意に指定することが出来る版識別子について述べる．版識別子はバージョン番号とブランチ識別子から構成される．

3.1 バージョン番号

バージョン番号は木の根からの高さを表している(図2)．根となる最初のバージョンは1となり，高さ方向に1ずつ増加する整数値を取る．

3.2 ブランチ識別子

3.2.1 構成法

まず用語を定義する．あるバージョンの直系の子バージョンを幹と呼び，それ以外をブランチと呼ぶ．例えば図1のバージョン１にはブランチが二つあると言うことが出来る．ブランチ識別子はn進数(n=3,4,5･･･)で構成され，根となるバージョン１は1となる．ブランチの分岐点以外の場所では，一つ前のバージョンと同じブランチ識別子を持つ．またブランチの分岐点の場合は以下のようになる．ブランチ識別子Aの分岐点が幹B0およびk個のブランチB_1,･･･,B_kを持つとき，任意の幹・ブランチのブランチ識別子Bxは，以下のようになる．

$$B_x = nA + x$$

つまり十進数のブランチ識別子でA=11の時，そこから派生する3番目のブランチB_3は以下の通りである．

$$B_3 = 10 \times 11 + 3 = 113$$

図1のバージョン系列はブランチ識別子を用いて表現すると図2のようになる．

詳しくは後述するが，このブランチ識別子から，木の祖先を示すことが出来る．Bx(0≦x＜n)の時，あるブランチ識別子Aの祖先の識別子は必ずAのプレフィクスとなる．この性質をBx(n≦x)の時にも拡張するため，オーバフローフラグの概念を取り入れた．

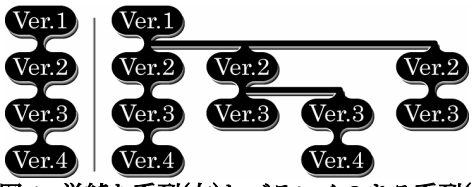

図 1: 単純な系列(左)とブランチのある系列(右)
Fig1: A Simple Version Series (Left) and A Version Series with Branches (Right)

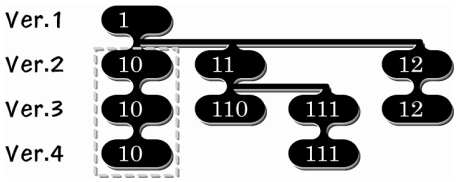

図 2: ブランチ識別子構成例
Fig2: An Example of Branch IDs

図 3: オーバフローの例
Fig3: An Example of the Overflow

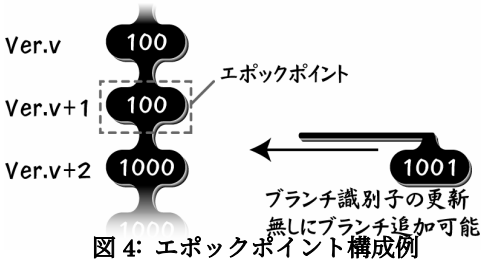

図 4: エポックポイント構成例
Fig4: An Example of the Epoch-Point

n-1のシンボル(十進数の時の9)をオーバフローフラグとして利用することにより，枝の数の制限をなくすことが出来る．オーバフローも考慮し一般化したブランチ識別子の構成法は以下の通りである．

$$B_x = n^{\left\lfloor \frac{x+n-1}{n-1} \right\rfloor}(A+1) - n + \{x \bmod (n-1)\}$$

例えば十進数のブランチ識別子でA=11のバージョンから発生する9番目及び19番目のブランチ識別子は

$$B_9 = (11+1)100 - 10 + 0 = 1190$$

$$B_{19} = (11+1)1000 - 10 + 1 = 11991$$

となる．オーバフローを用いれば，ブランチの数にかかわらず，プレフィクスの関係を用いて，祖先を示すことが出来る(図3)．

3.2.2 エポックポイント

分岐点以外のバージョン間ではブランチ識別子は変化しない(図2)．ブランチ識別子は可変長文字列型として関係データベースに格納する事を目的としている．すなわち桁が大きくなるとそれだけ多くのストレージ容量が必要となり，ま

たすぐに枯渇してしまう．そこでバージョン番号とブランチ識別子の組でバージョンを特定する事によって消費するリソースを低減させている．例えば図2のバージョン４・ブランチ識別子10であれば，一番左の系列のバージョン４と特定できる．

この手法で問題となるのが更新に伴いブランチ識別子の付け替えが発生する事である．提案方式では，予め後々ブランチが発生すると予想されるバージョンを指定する事で頻繁な更新を回避する事が出来る．このバージョンをエポックポイントと読んでいる．エポックポイントのバージョンAとその次のバージョンBの関係は以下のようになる．

$$B = nA$$

ブランチ識別子100のバージョンv+1をエポックポイントに指定した際の前後のブランチ識別子は図4の様になる．

3.2.3 バージョン系列指定

ブランチ識別子を用いれば、任意のバージョンからそのバージョンの祖先・子孫を特定する事が可能である．例えばブランチ識別子111の親となるブランチ識別子は11であり、そのまた親となるブランチ識別子は1である．このように，あるブランチ識別子Aのプレフィクスとなるブランチ識別子は全てＡの祖先のバージョンである．また，Ａがプレフィクスとなるブランチ識別子は全てＡの子孫である．

前述したようにブランチ識別子は可変長文字列型として関係データベースに格納して利用する．SQLを用いれば，ある文字列が別の文字列のプレフィクスになるかを問い合わせることが出来る．LIKE述語と%記号，そして各データベースの文字列関数を用いることにより可能である．

例えばブランチ識別子110自身及びその子孫のバージョンを取り出すには以下の構文を用いる．

```
SELECT * FROM テーブル
  WHERE ブランチ識別子 LIKE '110%';
```

この問い合わせによって，110をプレフィクスに持つブランチ識別子が選択される．

一方あるバージョンの祖先を調べる場合，指定した文字列のプレフィクスになる文字列をデータベース中から検索する必要がありLIKE述語を用いる事は出来ない．そこで文字列関数を用いる．

postgreSQLではposition(substring in string)関数を用いてstring内のsubstringの位置を知ることが出来る．つまりposition(branch_id in '110')が1を返す時，branch_idは自分自身又は祖先であると言える．

Microsoft SQL Server 2000でも同等の役割を持つCHARINDEX関数がある，また他の多くの関係データベースで同じ機能を持つ関数を利用することが出来る．

3.2.4 ブランチ識別子の拡張

本稿では説明の為に十進数のブランチ識別子を用いているが，ブランチの数が多くオーバフローが頻繁に発生すると予想されるならば，任意の数を用いることができる．また，途中から拡張することも可能である．例えば一つのバージョンから最大で998のブランチが発生する場合，十進数を用いるよりも千進数を用いた方がストレージ容量を低減できる．十進数のブランチ識別子が途中から千進数に変更されても，プレフィクスの関係は保たれる為，検索に影響はない．なお，検索は文字列として行われる為，各数のシンボルには各ロケールの文字セットやUNICODE文字を含む任意の文字を割り振ることが出来る．これは利用する関係データベースがサポートするエンコード方式による．

4. 版管理手法と実験

4.1 版データベースの構成と検索

RCSの様な版管理ツールはそれが管理する全てのバージョンで全てのデータを保存しているわけでは無い．ストレージ容量軽減の為に，前バージョンとの差分しか保存していない．我々が研究している手法も同じく差分データのみを格納する事によって版管理を行っている．版管理手法の具体的なアルゴリズムに関しては，差分データの取得手法や差分データを取る単位等の問題があり，本稿の範囲を越えてしまう為，ここでは詳述しない．本節ではより一般的な形の例を用いて説明を行う．図5のようなバージョン系列を想定する．これを格納する関係データベース上のテーブルはバージョン番号・版識別子・差分データから成る．バージョン1・ブランチ識別子1は無に対する差分として全てのデータを保存する．続いてバージョン2の2つの版はバージョン1に対する差分を保存する．同じように，新しい版が加えられる際には，親バージョンとの差分データのみを保存する．

このようにして構成されたバージョン系列から特定の版の再構成を考える．検索する版の版識別子と共に保存されているデータは，親バージョンとの差異のみである．よって祖先の系列に遡って，差分データを収集する必要がある．祖先の系列を検索する事は提案手法を用いれば簡単に出来る．例えばバージョン4・ブランチ識別子1111の版を構成すための問い合わせは次のようになる．

```
SELECT * FROM テーブル
  WHERE バージョン番号<=4
    AND CHARINDEX(ブランチ識別子,'1111') = 1;
```

この問い合わせにより，バージョン 4・ブランチ識別子 1111 を構成するデータの断片(🍙🍘🍙🍙)をバージョン系列から取り出すことが出来る．

実際には，この問い合わせによって取り出されたデータには，別の差分データによって上書きされた古いデータも含まれる可能性がある．これを考慮した版検索は文献[6]に詳しい．

4.2 版識別子の検索性能実験

版識別子の検索性能を調べる為の基礎的な実験を行った．varchar型のフィールドを一つもつテーブルを用意し，そこにランダムな文字列を10万レコード登録した．この文字列を擬似的なブランチ識別子とみなす．条件を変え9回ずつ子孫検索・祖先検索を行った．この際，結果として得られるレコード数が各条件の問い合わせ全てで必ず10になるように登録されているデータを調整した．こうすることにより，クライアントへの転送時間をほぼ一定にする事ができ，検索時間

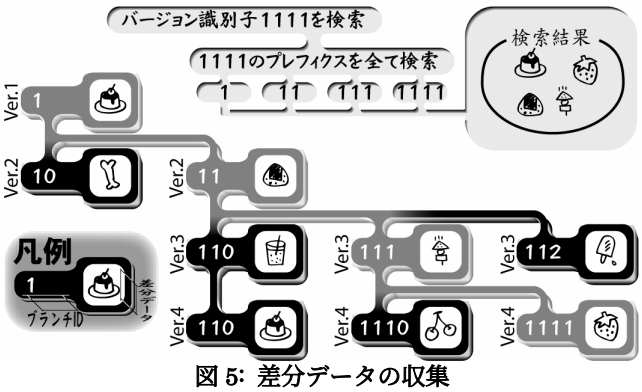

図 5: 差分データの収集
Fig5: Gathering Difference

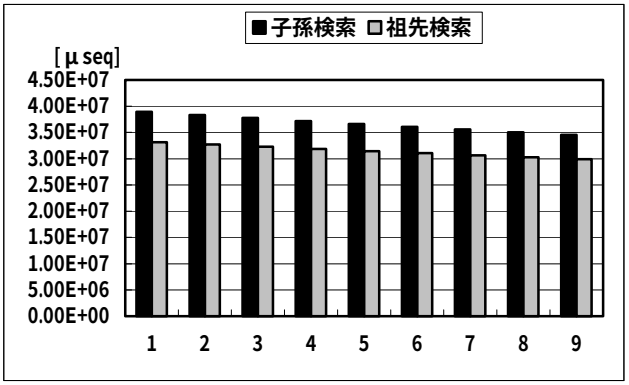

図 6: 検索時間
Fig6: Execution Time of the Query

のみの比較を行う事が出来る．

実験で得られた検索時間を縦軸にとり，結果を図6に示す．各条件1≤n≤9はそれぞれn文字一致のクエリを発行した事を意味する．例えば子孫検索の4は4文字前方一致を条件とした検索を行っている．版識別子はブランチの数が増えると桁が増加するが，検索する際に桁の大小が検索時間にどう影響するかを調べるのがこの実験の目的である．結果，全ての条件下でほぼ一定の検索時間を得られている．

revision番号を基に版管理を行うツールRCSは，最新のバージョンは全て保存し，古いバージョンは一つ新しいバージョンとの差分データを保持している．つまり最も古いバージョンを取り出す為には，その版の数だけ復元操作を繰り返す必要がある．提案する版識別子と関係データベースを用いる事によって，深い階層のブランチに属するデータも，また浅い階層に属するデータもほぼ一定の検索時間を持って取り出す事が可能である．この版識別子を利用すれば，ブランチの有るバージョン系列を効果的に管理する事ができる．

5. まとめと今後の課題

本稿では，ブランチを持つバージョン系列が木構造をとる事に着目し，木構造のラベリング手法を応用した版識別子を提案した．

版識別子はバージョン番号とブランチ識別子で構成される．ブランチ識別子は以下の特徴を持つ．

- ブランチ識別子Ａのプレフィクスになるブランチ識別子ＢはＡの祖先である．
- エポックポイントを効果的に用いることにより，バージョンの更新や新規ブランチ作成の際にブランチ識別子の更新が生じない．
- バージョン番号と組み合わせる方式によりバージョン識別子の枯渇が起きにくい．

さらにバージョン検索に関する実験を行い，検索コストが木構造をとるバージョン系列の位置に依存しない事を確認した．

今後は以下の課題に取り組む予定である．

- 提案した版識別子を利用して版管理手法の構築と，そのシステムの実装を行う．
- ブランチのマージ等や差分データの取得とそれに基づいた更新等，版管理手法としての発展を考える．
- 版管理手法の応用について考える．例えば，版を考慮したWebサイトの構築・管理ツール等を考えている．

[謝辞]

本研究の一部は，文部科学省科学研究費特定領域研究(2)[情報学:A02](課題番号:14019075)，東京都立大学総長特別研究費(特別重点研究)，日本データベース学会・マイクロソフト株式会社共催 2002 年度データベース研究支援プログラムによる．

[文献]

[1] 横山昌平，太田学，片山薫，石川博，"XML パーサを考慮した応用向き関係データベース構成法"，第 13 回データ工学ワークショップ(DEWS2002)論文集，A5-3，http://www.ieice.org/iss/de/DEWS/proc/2002/ ，Mar. 2002.

[2] Walter F Tichy, "RCS-A System for Version Control," Software-Practice & Experience, 15, 7, pp.637-654, July 1985.

[3] Quanzhong Li and Bongki Moon, "Indexing and Querying XML Data for Regular Path Expressions, "Proceedings of the 27th VLDB Conference, Roma, 2001.

[4] 佐藤隆士，里本智彦，小畑喜平，潘洪涛，"階層構造を識別可能な木節点の番号付け"，第13回データ工学ワークショップ（DEWS2002）論文集，A4-8，http://www.ieice.org/iss/de/DEWS/proc/2002/ ，Mar. 2002.

[5] Serge Abiteboul, Haim Kaplan, and Tova Milo, "Compact labeling schemes for ancestor queries," Proc. SODA 2001, pp. 547-556, 2001.

[6] 横山昌平，太田学，片山薫，石川博，"関係データベースを利用した XML 版管理手法"，第 14 回データ工学ワークショップ(DEWS2003)論文集，6-C-02，Mar. 2003.

横山 昌平 Shohei YOKYAMA
東京都立大学工学研究科博士課程在学中．2001 年 東京都立大学工学部電子情報工学科卒業，関係データベースを用いた XML 処理の研究開発に従事．情報処理学会学生会員．日本データベース学会学生会員．

太田 学 Manabu OHTA
東京都立大学大学院工学研究科助手．1999 年 東京大学大学院工学系研究科電気工学専攻博士課程修了，博士(工学)．情報検索，データマイニングとその Web 応用の研究開発に従事．電子情報通信学会，情報処理学会，日本データベース学会，各会員．

片山 薫 Kaoru KATAYAMA
東京都立大学大学院工学研究科助手．2000 年 京都大学大学院情報学研究科社会情報学専攻博士後期課程修了，博士(情報学)．データベースシステムに関する研究開発に従事．情報処理学会，日本データベース学会，各会員．

石川 博 Hiroshi ISHIKAWA
東京都立大学大学院工学研究科教授．富士通研究所をへて 2000 年より現職．1979 年東京大学理学部卒，博士(理学)．データベースシステムの研究開発に従事．情報処理学会，電子情報通信学会，ACM，IEEE，各会員．International Journal on Very Large Data Bases Editorial Board，Sigmod Japan 評議員，日本データベース学会理事．著書に「E-ビジネス技術入門教科書」「Database and Data Communication Systems」(Academic Press，共著)など．