SuperSQLによる関係データベースとXMLデータの統合利用

赤堀正剛[†] 有澤達也[††] 遠山元道[†††]

ネットワーク上の大量のXMLデータと既存の関係データベース内のデータの並存や連動において，変換処理や統合検索が必要である．しかし，XMLと関係データベース間にはデータモデルや検索言語の違いがあり，障害となっている．本研究ではデータベース出版機能を持つ検索言語SuperSQLを用いて関係データベース/XML間の相互変換，すなわち，関係データベース内のデータのXMLデータビューの生成，XML質問文（XQL）によるXMLデータビューへの問合せ，XMLデータの新規，既存のリレーションへの格納を実現した．この結果，ユーザがSQLやデータベース内部の構造等の知識を持たない場合でもXMLデータの仮想ビューを通してデータベース内部のデータの閲覧や問合せが可能となる．また，データ交換で受け取ったXMLデータと自らの関係データベース内の既存のデータを統合し，これらのデータをSQLで利用できるとともに新たなXMLデータとして第三者に提供することが可能である．

キーワード：XMLデータビュー，XMLデータ格納，XML, XQL, SuperSQL

Data Integration on Relational Database and XML Using SuperSQL

Masatake Akahori,[†] Tatsuya Arisawa[††]
and Motomichi Toyama[†††]

In this paper, we address the problem of automating the conversion between relational data and XML, and of querying the XML data. SuperSQL, a general database publishing tool, is used for generating XML data views from relations. Users can query XML data views so that they can select data without having knowledge on underlying database that they need. Furthermore, we expand SuperSQL to store XML data into RDB. Users can access DB through XML data views generated by SuperSQL, and query XML data using XML query language (XQL), without having to deal with the underlying SQL tables and query language. Also, they can use both of relational data and xml data without distinction.

Key Word: XML Data View, XML Data Storing, XML, XQL, SuperSQL

1. はじめに

近年，WWW利用の増大やネットワークのオープン化にともない多様な情報源の利用が可能となってきており，特に様々な情報を流通させる際の共通フォーマットとして利用されつつあるXML[1)]に注目が集まっている．XMLは柔軟なデータ表現力と拡張性を持ち，様々なデータの表現，B2Bの電子商取引やウェブサービスの提供等の活用が期待されている[2)~4)]．バイナリデータではなくテキストデータであるため人間とアプリケーション双方に対して可読性があることによるデータの取扱いの簡略化や，パーザや関連規格が充実してきたことによるアプリケーション開発の簡便化等のためである．

一方，関係データベースには膨大な量のデータが蓄積，運営されており，依然として多くのシステムの情報源となっている．また，SQLを検索言語とするアプリケーションも多く存在する．そこで，今後ネットワーク上に大量のXMLデータ☆と既存の関係データベース内のデータが並存，連動することが予想され，それらの変換処理や統合検索の要求が高まってくると考えられる．

しかし，XMLと関係データベース間にはデータモデルや検索言語の違いがあり，これらが障害となって容易に統合した検索ができないという問題がある．そこで相互にデータを変換し，互いの検索言語で双方の

† 慶應義塾大学大学院理工学研究科管理工学専攻
Department of Administration Engineering, Keio University
†† 慶應義塾大学大学院理工学研究科計算機科学専攻
Department of Computer Science, Keio University
††† 慶應義塾大学理工学部情報工学科
科学技術振興事業団さきがけ研究21「情報と知」領域研究員
Department of Information and Computer Science, Keio University. PRESTO, JST

☆ XMLで記述したデータを本論文ではXMLデータと呼ぶ．

データを検索するという手法が考えられる．しかしながら，関係データベース中のデータは膨大な量であり，そのすべてをXMLデータに変換してから問合せを行うのは効率的ではない．そこで関係データベースを中心としてこれに対しXMLデータビューを提供し，そのビューに対し検索を行うという手法が考えられる[5]．また，データ交換で受け取ったXMLデータを管理するという意味でも関係データベースを中心と考えてそこにXMLデータを挿入してデータの管理を行い，関係データベース内の既存のデータとともにSQLで検索可能にするという手法が考えられる[6]．この際，同種の情報を既存のリレーションへ追加を行ったり，異種の情報を新規リレーションへ格納したりする必要がある．

我々はデータベース出版機能を持つSuperSQL[7)~11)]を用いて関係データベース/XML間の相互変換，すなわち，関係データベース内のデータのXMLデータ（ビュー）およびその構造情報の生成[12]，XML質問文（XQL[13),14)]）によるXMLデータビューへの問合せ[5]，XMLデータの新規，既存のリレーションへの格納[6]を実現した．この結果，ユーザはSQLやデータベース内部の構造等の知識を持たずにXMLデータの仮想ビューを通しデータベース内部のデータの閲覧や問合せが可能となる．また，XMLデータを関係データベースへ格納することにより，データ交換で受け取ったXMLデータと自らの関係データベース内の既存のデータのSQLによる統合利用ができ，これらのデータを使用してさらに新たなXMLデータとしてを第三者へ提供可能となる．

本論文は以下のような構成である．2章ではXMLやXML/関係データ間の違い，関連する研究を，3章では提案システムの概要を述べる．また，4章では関係データベース内のデータに対するXMLデータビューの生成について，5章ではXQLによるXMLデータビューに対する問合せの処理について，6章では拡張したSuperSQLによるXMLデータの関係データベースへの格納について述べる．7章ではこれらの機能の連携について，8章では提案システムの評価および検討を行い，最後に9章でまとめる．

2. 関連技術

2.1 eXtensible Markup Language（XML）

XML文書の構成は大きく，‘XML宣言’，‘文書型定義（DTD：Document Type Definition）’，‘XMLインスタンス’の3つからなる☆．XML宣言はXMLのバージョンと符号化方式を指定する．文書型定義はどのような構造のXMLインスタンスを認めるかという定義であり，文脈自由文法にほぼ相当する．

XMLインスタンスでは要素が階層構造化されており，各要素は開始タグ（要素型を〈と〉で囲んだ文字列）と終了タグ（要素型を〈/と〉で囲んだ文字列）の対に囲まれ区切られている．開始タグと終了タグの間の文字列がその要素の内容である．開始タグと終了タグの対は入れ子とすることが可能であり，これにより階層構造を表現する．開始タグと終了タグの間に文字列と要素を交互に記述することも可能であり，これを混在内容という．たとえばDTDで$(\#PCDATA|Keyword)*$と表現され，実際の要素が

```
<Abstract>
  本研究ではデータベース出版機能を持つ
  検索言語<Keyword>SuperSQL</Keyword>を
  用いて...
</Abstract>
```

と表現される形式の内容のことを指す．また，開始タグでは要素型と＞の間に属性（属性名と属性値を‘=’でつないだもの）を記述することが可能である．なお，XMLの属性とデータベースの属性の意味するところは異なるため，本論文ではXMLの属性を単に‘属性’，データベースの属性を‘データベース属性’と呼ぶ．

2.2 データ中心的XML文書と文書中心的XML文書

XMLは応用の趣旨によって大きくデータ中心的なXMLと文書中心的なXMLに分けられる[15]．データ中心的なXMLは均一な構造であり，混在内容をほとんど含まず，データの最小単位がDTDでいう#PCDATA（分析対象となる文字データ）のみである要素や属性である．一方，文書中心的なXMLは不均一な構造であり，混在内容を多く含み，データの最小単位が混在内容，もしくは文書自体である．

さらに，各々のXMLでその意味が要素間の順序に依存するものと順序に独立のものがあり，以上よりXMLは以下のカテゴリへ分けられる☆☆．

(1) データ中心的かつ順序依存
(2) データ中心的かつ順序独立
(3) 文書中心的かつ順序依存
(4) 文書中心的かつ順序独立

本研究でターゲットとするXMLはカテゴリ(2)が中心である．

☆ XML宣言とDTDは必須ではない．

☆☆ 文書では情報の出現順に大きな意味があり，カテゴリ(4)は無意味である．

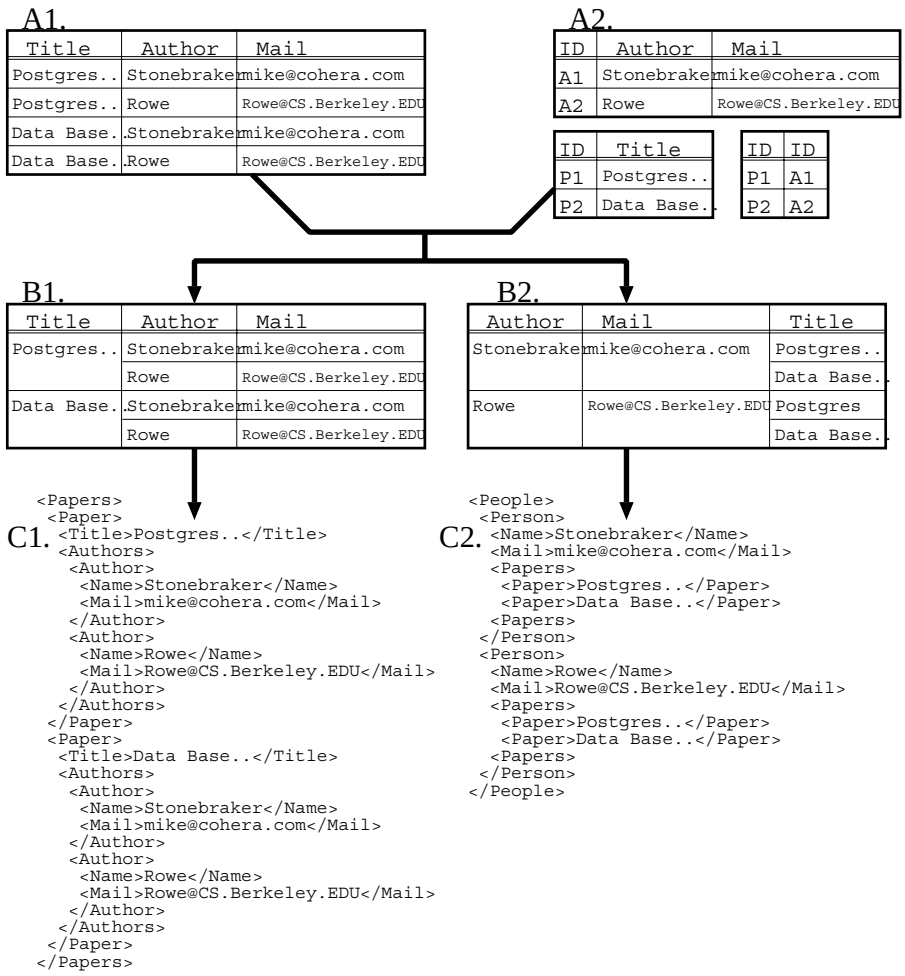

図 1 関係データから XML データへの変換
Fig. 1 Exporting relational data to XML data.

2.3 XML と関係データ

関係データは表であり，平坦で正規化された構造を持つ．各々のデータベースごとに異なったスキーマを持ち，直接のデータ交換には向かない．一方，XML データは半構造[17]なデータを表現する一手段であり，入れ子で正規化されていない木構造を持つ．いわば入れ子型関係データに似た構造を持ち，また，XML データは公開された構造情報（DTD 等）に従って交換される．このため，関係データから XML データを生成する場合には，様々なスキーマの関係データをある構造情報に従った XML データに変換したり，同じ関係データから様々な構造の XML データに変換したりする必要がある．両者の構造は完全に異なり，平坦な関係データを複雑な木構造の XML データに変換するツールには十分な一般性が必要である．例として**図 1** に，異なるスキーマの関係データから 2 種類の構造の XML データを生成する流れを示す．

関係データは正規化されており，XML データは正規化されていない．XML データの要素の包含関係（木構造の親子関係）に相当する情報は関係データでは外部キーによる参照関係によって間接的に表現している．XML データの親要素を共有する兄弟要素の関係は関係データにおける各タプル内や外部参照を通しての属性値の対応関係に相当する．特に，関係データの外部参照等により表現される 1 対多の関係は，DTD における‘*’による構造によって XML データでは表現される．たとえば，図 1-C1 の XML データにおける Title 要素と Name 要素（Mail 要素）や図 1-C2 の XML データにおける Name 要素（Mail 要素）と Paper 要素は 1 対多の関係を表現しており，これらは図 1-A1, A2 の多対多の関係を含む関係データに対応している．

以上のことから，関係データから XML データを生

成するには関係データを非正規形へ変換するために木構造化を行い，さらに要素の包含関係を付加する必要がある．たとえば，図1では関係データを木構造化して要素の包含関係を付加する流れを示している．

一方，木構造であるXMLデータを関係データに変換する際には

(1) 構造情報（包含関係，順序に関する情報等）
(2) 要素名に関する情報
(3) データの内容そのもの（DTDの‘#PCDATA’に相当する）
(4) メタデータ（バージョン情報，制作日時，制作者情報等）

の4種類の情報の格納が対象となる．

XMLデータを関係データとして格納する方法として，3つの方法が考えられる[16]．

A. XMLデータをファイルとしてそのまま管理し，XMLデータ内の指定された要素にのみインデクスを作成する．
B. XMLデータの各要素を関係データの属性に対応させる写像に基づき各要素の値を格納する[18]．
C. XMLデータの木構造に基づき，要素をノードとリンクに分解して対応するリレーションへ格納する[19]．

これらは格納したXMLデータを再度XMLデータとして再利用するため，先ほどあげた4つの情報（構造情報，要素名，内容，メタデータ）をすべて，もしくはメタデータを除いた3つの情報を保存している．

これに対し，本研究ではXMLデータの内容を既存の関係データとともに区別なく利用することを目的としている．このため，XMLの要素を格納方法(A)のようにファイルに残したり，格納方法(C)のようにある特定のリレーションへ格納することなく，任意のリレーションへ格納できるように任意のデータベース属性と射影を行う必要がある．すなわち，格納方法(B)を参考に，指定したXMLデータ内の要素を関係データの属性へ対応させる写像をユーザが宣言的に指定し，これに基づきXMLデータの内容情報を関係データとして格納する．また，要素名は任意の名称の属性名へ変更が可能である☆．

構造情報に関しては

(1) 要素の包含関係
(2) 要素の順序
(3) 単一要素の値の順序

等に分類できる．構造情報として本来データベースに格納したい情報（またはその一部）が表現されていることもあるが，このうち‘要素の包含関係’は外部キーによる参照関係により間接的に格納する．また，‘単一要素の値の順序’は，その要素の値から生成したキー値に順序情報を織り混ぜることにより格納することが可能である．

正確な‘要素の包含関係’に関する情報の格納は格納方法(C)に分類される文献19)において示されている．この方法では要素をノードやリンクに分解することにより‘要素の包含関係’の格納を実現しており，どのノードやリンクの情報も専用のリレーションに格納される．要素名に関する情報は‘要素名’属性の値として格納され，要素の包含関係を再現するためには入れ子のSQLによる問合せが必要となる．また，ある要素名のデータを取り出したい場合にはWHERE節にその要素名を条件として記述する必要がある．この場合，たとえその要素が少量しか存在しない場合でもすべての要素から派生したノードに関する情報との突き合わせが必要となり効率が悪くなる．また，この方法ではXMLの要素の値を任意のデータベース属性に射影させることはできず，値の格納とは別個に行う必要がある．

本研究では，関係データからXMLデータを生成する場合には要素の包含関係や順序の指定に関してはSuperSQL質問文のTFEの中に記述する．このため，SuperSQL質問文を生成する際に何らかの構造情報からTFEを生成し，ユーザを支援することが考えられる．現在DTDよりTFEの雛型を自動生成する手法を利用しているが☆☆，前述の文献19)の手法やXMLインスタンスから構造情報を抽出する手法を利用してDTDと同様の構造情報を関係データとして格納し，それを元にTFEを生成する手法も考えられる．

2.4 関連研究

本研究で対象とするのは関係データベース/XML間の相互変換，すなわち，(1) 元々データベースに格納され運営中であるデータに対するXMLデータビューとその構造情報の生成，(2) そのXMLデータビューに対するXML質問文（XQL）による問合せ処理，(3) データ交換で受け取ったXMLデータの関係データベースへの格納による既存データとの統合の3つであり，これらの連携を行う．

XMLデータビューの生成と問合せ処理の関連研究としてはSilkRoute[20]やXPERANTO[21),22)]が存在

☆ 構造や順序に関する情報を格納するためには，ユーザがそれを格納できるような属性を用意する必要がある．

☆☆ 現在，「｜」が含まれていない場合のみDTDからのTFE生成が可能である．

する．SilkRouteはRXLという言語を用いて関係データベースに対するXMLデータビューを定義する．一方，XPERANTOは関係データベース内のリレーションと同様の構造を示すXMLデータのベースビューを元にして，XML-QL[23]を用いて新たなXMLデータビューを生成する．両者ともビューに対してはXML-QLにより問合せを行う．XML質問言語の違いがあるが，XMLビューの定義とXML質問文の実行のために別の質問言語（SuperSQL）を使用する点では本研究はSilkRouteに似ている．しかし，質問結果を新たなビューとしてユーザに提供できる点ではXPERANTOに近い．ただし，本研究では質問を処理するとともに，生成されたXMLビューの構造情報（DTD，RELAX[24]）も同時に扱う点がSilkRouteと異なる．また，HTML等に変換し視覚化するためのXSLの生成も可能であり，これによりXMLデータの内容を閲覧することができる[5),12)]．この点がXPERANTOと異なる．なお，言語仕様に関する記述は8.3節で述べる．

構造情報の生成に関しては，DTDをXML文書自身から抽出する手法[25)]と，質問文と問合せの対象となるXML文書の構造情報を用いて求める手法[26)]等が提案されている．一方，本研究では，(1) XMLデータビューを定義するSuperSQL質問文より生成，(2) ビューを生成する元となったデータソースの情報を元に生成，等と構造情報を生成する際に利用する情報がまったく異なる．

XMLデータの格納に関しては，いくつか提案がなされている[18),19),27)]．しかしこれらは管理が主体であり，専用のリレーションやDTDに対応したリレーションにXML文書を格納し，格納したXML文書はあくまでXML文書として利用する．このため，データベース内の既存リレーションに追加したり，既存のデータとともに利用することは難しい．一方，データベース内の既存のデータとの統合利用を目的とした手法[28)]も提案されている．これは選択したXML要素を単一リレーションへ展開し，SQLにより関係データベース内の既存のデータと統合利用を行う．しかし，単一リレーションへの展開は同じデータの重複等弊害も多い．そこで本研究では拡張SuperSQL質問文により選択したXMLデータの要素を既存もしくは新規の複数リレーションへ格納し，これらの問題に対処する．

本研究では本節の冒頭で述べた3つの機能を連携し，関係データのXML質問文による利用や，（関係データベースへ格納した）XMLデータを関係データベース内に既存のデータとともに区別なくSQLやSuperSQLによる利用を行う．さらに，これらの2種類のデータを関係データベースのXMLデータビューを通してXML質問文でも使用することができる．つまり，XMLデータと関係データを統合的に利用することが可能である．ただし，いったん関係データベース内へ格納して関係データとして統合を行う．

XMLデータや関係データを統合的に扱うものとして，異種情報源統合システムが存在する[29),30)]．たとえばAgora[29)]は複数の情報源（関係データやXMLデータ等）を統合した仮想的なXMLに対し，XML質問文（QUILT[31)]）を用いて問合せを行い，XMLを結果として得るシステムである．いったんすべてのXMLデータを関係データとして関係データベースに格納する本研究とは異なり，各々の情報源は何も加工せずそのままの状態にしたままラッパーを介してアクセスする．このような点と，質問文の種類の違いが大きな相違点である．

3. システムの概要

提案するシステムは，データベースとユーザやアプリケーション間のミドルウェアシステムであり，関係データベース/XML間の相互変換を行う．主な機能は以下のとおりである．

(1) XMLデータビューの生成
SuperSQL質問文により，関係データベースに対するXMLデータビューとそのDTDやRELAX等の構造情報（および視覚化を行うXSL）の生成を行う．詳細については4章で述べる．

(2) XMLデータビューへの問合せ
XML質問文（XQL）の問合せと同等の結果を出力するSuperSQL質問文を生成・実行して問合せの結果を得る．ユーザはSQLやデータベース内部の構造等の知識を持たずにビューを通してデータベース内のデータの閲覧や問合せが可能である．詳細については5章で述べる．

(3) XMLデータを関係データベースに格納
受け取ったXMLデータを関係データベース内の既存のデータとともにSQL等で使用するため，拡張SuperSQLを定義し，その質問文により新規もしくは既存のリレーションに分割して格納する．詳細については6章で述べる．

これらにより，SQLやXQLでXMLデータと関係データベース内の既存のデータをともに利用可能となる．また，関係データベースへ格納したXMLデータと既存のデータを区別なく利用して第三者にさらに新たなXMLデータビューを提供できる．

本システムにはXMLデータベースビューの提供や

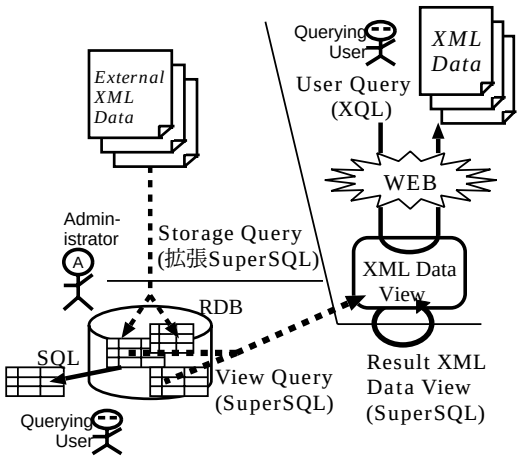

図 2 本システムの概要
Fig. 2 Overview of the system.

受け取った XML データの関係データベース内への格納を行う管理者（administrator）と，そのビューに対し XML 質問文（XQL）で問合せを行う，もしくは SQL で関係データベースへ問合せを行う問合せユーザ（querying user）が存在する．本システムを利用したデータ流通の概要を**図 2** に示す．

4. XML データビューの生成

SuperSQL 質問文により，関係データベース内のデータに対する XML データビューとその DTD や RELAX 等の構造情報（および視覚化を行う XSL）の生成を行う．XML データの構造情報は，ユーザやアプリケーションが XML データを使用する際に有益である．

4.1 SuperSQL（View Query）

SuperSQL はデータ，構造およびレイアウト，出力媒体（メディア）の 3 つの要素から成り，SQL のデータ操作機能に加えて出力構造やレイアウト，出力媒体の指定ができ，様々な応用データの生成が可能である．

SuperSQL の質問文は，SQL の SELECT 節を拡張し ‘GENERATE ⟨medium⟩ ⟨TFE⟩’ という構文を持つ GENERATE 節で置き換えたものである．ここで ⟨medium⟩ は出力媒体を示し，ここに XML を指定すると XML データを生成する．このほかにも HTML，Java，Excel，LaTeX 等の指定ができ[7),9),11)]，これにより関係データベース内のデータを中心とした One-source Multi-use の一環として，同じ構造のデータを容易に複数の出力媒体で利用可能である．⟨TFE⟩ はターゲットリストの拡張である Target Form Expression（TFE）を表し，結合子や反復子等によるデータの構造化とレイアウト指定を行う一種の式である．

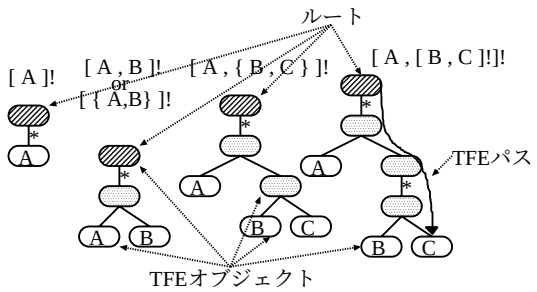

図 3 TFE による木構造の表現
Fig. 3 Tree structure representations in TFE.

4.1.1 Target Form Expression（TFE）

TFE はデータの構造化とレイアウト指定を行う．すなわち，結合子（「,」「!」等）によるデータや構造化されたデータの結合と，反復子（[ ] の対の後に結合子が続くもの）に囲まれた内容のグループ化，{ } によるまとまり（部分式）の指定が可能である（**図 3** 参照：図中の ‘*’ は反復を表す）．

図 1-A1 の関係データを例に，TFE による構造化について述べる．TFE において，XML の生成を行う場合 { } で括られた部分式は組を表す．[ ], や [ ]! の反復子でこの組を括った場合，組の反復を示す．たとえば，

```
[ { Title, Author, Mail } ]!
```

は以下の組の集合を表す．

```
{
  { Postgres...,
      Stonebraker, mike@cohera.com },
  { Postgres...,
      Rowe,        Rowe@CS.Berkeley.EDU },
  { Data Base...,
      Stonebraker, mike@cohera.com },
  { Data Base...,
      Rowe,        Rowe@CS.Berkeley.EDU }
}
```

ただし，この場合内側の { } は省略して

```
[  Title, Author, Mail  ]!
```

と記述してもよい．

図 1-A1 の関係データでは，Title データベース属性の値は同じではあるが $\{Author, Mail\}$ の異なる値を持つ組が存在する．この特徴を利用して，同じ Title データベース属性の値に対して $\{Author, Mail\}$ の組の複数の値が対応している☆構造のデータを表現して

☆ Title データベース属性により $\{Author, Mail\}$ の組がグループ化されていると表現する．

いる TFE は

```
[ { Title, [ { Author, Mail } ]! } ]!
```

や

```
[ Title, [ Author, Mail ]! ]!
```

等と記述する．この結果，図 1-A1 の関係データは

```
{
  { Postgres...,
    { { Stonebraker, mike@cohera.com      },
      { Rowe,        Rowe@CS.Berkeley.EDU } } },
  { Data Base...,
    { { Stonebraker, mike@cohera.com      },
      { Rowe,        Rowe@CS.Berkeley.EDU } } }
}
```

という 2 つのグループ化された組となる．

一方，TFE を

```
[ Title, [ Author ]!, [ Mail ]!  ]!
```

とすると，Title データベース属性により Author データベース属性がグループ化され，同時に Title データベース属性により Mail データベース属性が（author とは独立に）グループ化されていることを表す．この結果，図 1-A1 の関係データは

```
{
  { Postgres..., { { Stonebraker },{ Rowe } },
                 { { mike@cohera.com },
                   { Rowe@CS.Berkeley.EDU } } },
  { Data Base...,{ { Stonebraker },{ Rowe } },
                 { { mike@cohera.com },
                   { Rowe@CS.Berkeley.EDU } } }
}
```

となる．以上のように TFE ではデータベース属性値ごとのグループ化を行うことによってデータの構造化を行う．

TFE では関数が使用でき，データベース属性やグループ化されたデータベース属性等を引数とする．結合子の記号は水平方向や垂直方向といった次元に対応しており，構造化されたデータは結合子や反復子の次元に従ったレイアウトが適用される[32]．

以上のように TFE で使用するオブジェクトとしてデータベース属性や { }，反復子，関数があげられ，これらを TFE オブジェクトと称する．また，TFE オブジェクトのルートとなる最外の反復子からある TFE オブジェクトまでの経路を，その TFE オブジェクトの TFE パスと呼ぶ（図 3 参照）．

4.1.2 装 飾 子

各データベース属性や { } で括った部分式，反復子，関数に対するオプションを指定する．オプションは '@' の後に続く '{' と '}' の間に記述し，オプション名と引数の対を '=' で結合して指定する．複数のオプションを指定する場合は ',' を挟んで並べて記述する．たとえば，対象となる TFE オブジェクトに対して名前を付加する name オプション等が存在する．

4.1.3 関 数

TFE 内には関数の記述が可能であり，TFE 中のデータベース属性，もしくは { } で括った部分式や反復子等で構造化されたデータベース属性等を引数とする．たとえば以下の関数が存在する．

- NULL 関数 ··· 構造化の際に引数を利用するが，引数を出力しない場合に使用する．
- verb 関数 ··· 文字列を引数とし，その文字列を固定出力する．

4.2 SuperSQL による XML データビューの定義

XML データビューの定義はビュー定義質問文（SuperSQL）を使用する．関係データから XML データを生成するには

(1) 関係データの構造化（木構造化）
(2) 構造化データに対する要素の包含関係（親子関係）の付加

が必要である（2.3 節参照）．SuperSQL 質問文ではこれを TFE と装飾子によるオプションで実現し，関係データから XML データを出力する．同時に DTD や RELAX 等の構造情報（および視覚化を行う XSL）を生成する．

4.2.1 TFE による構造化データ生成

4.1.1 項で示したとおり，SuperSQL では TFE の記述を利用して関係データを構造化することができる．

4.2.2 TFE オブジェクトと XML データ

TFE はデータベース属性や結合子，反復子，{ } の組合せから成り立ち，図 3 のように木構造を示すことができる．

- データベース属性 ··· テキストの値を持つ要素や属性に対応する．
- 結合子 ··· データベース属性や反復子，{ } で括られた部分式を連結する．データベース属性や反復子，{ } は要素に対応するため，結合子は要素を並べる（要素の兄弟関係を形作る）役割を持つ．
- 反復子 ··· 反復子で括られた部分式は値がある限り反復して出力されるため，値必ず 1 つ以上存在する場合は DTD の「+」に相当する動作を，0 個以上存在する場合は，DTD の「*」に相当する動作を行う．反復子自身はこれら反復して出力された値を括る存在であり，反復している要素を括

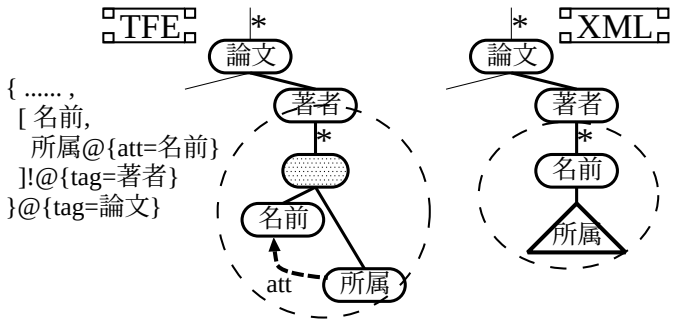

図 4 TFE と XML の構造の差
Fig. 4 The differences between TFE and XML structures.

る要素となる．

- { } ··· 括っている部分式には各々が要素となるデータベース属性や反復子，{ }，あるいはその組合せが含まれ，つまり要素の集まりとなる．{ } 自身はそれらを括る要素に対応する．

4.2.3 オプションによる構造化データの XML データ化

構造化したデータを XML データに変換するには，要素の包含関係を付加することが，つまり個々のデータやデータの集まりに要素名を付加することが必要がある．装飾子によるオプションで TFE オブジェクトに要素名を付けることでこれを実現する．

TFE により構造化されたデータに対し，要素名を付加する等を行って XML データ化を行う．要素名の付加等のためには装飾子によるオプションを利用する．たとえば以下の指定を行う．

(1) name（名前付け）··· 対象（データベース属性）の要素名を付ける☆．
(2) tag（名前付け）··· 対象（{ }，反復子）の要素名を付ける．
(3) att（属性付加）··· 引数で指定した要素に対し，対象（データベース属性）の名前を属性名として，対象の値を属性値として属性を付加する．

att オプションを持つオブジェクトは，オブジェクト自身を att オプションの引数で示された XML エレメントの属性とする．このため，TFE で示される構造と XML の構造は属性に関する構造が異なる（**図 4** 参照）．

図 5 で示すデータ☆☆を使用した XML データビューの例を**図 6** に示す．

☆ name オプションを指定しないとデータベース属性名が要素名となる．
☆☆ データ構造の簡便化のため，作為的に Book の Author は 1 人であると限定している．

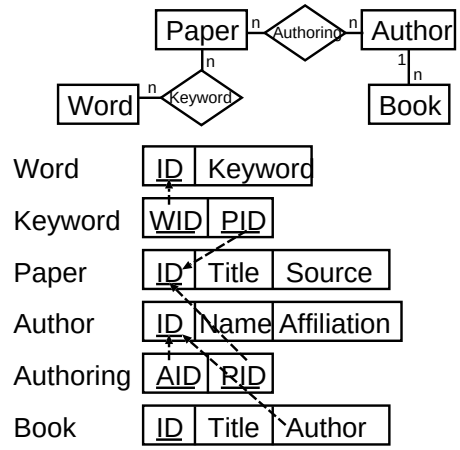

図 5 サンプルデータ
Fig. 5 Sample data.

4.3 SuperSQL による構造情報の生成

構造情報を生成するには以下の方法がある．基本的には (1) の手法で生成し，必要な要素のみ (2) の手法を用いる．これらにより DTD や，データベース内の情報を利用してさらに型情報を加えたパラメータ実体のない RELAX を生成する．

(1) TFE から生成（4.3.1 項参照）
(2) XML データビューの元となるデータを分析して生成（4.3.2 項参照）

4.3.1 TFE を用いた構造情報の生成

XML データビューを定義する SuperSQL 質問文の TFE から同じ構造を示す DTD や RELAX を生成する☆☆☆．たとえば反復子の中のグループ化されたオペランドはデータがある限り反復して出力されるため，DTD の「*」に相当する動作を行う．

··· 例：‘[A]!@{$name = List$}’
→ ‘ <!$ELEMENT\ List\ \ (A)*$ >’

☆☆☆ これは非常に緩やかな構造制約を生成する．

A.XMLビューの定義（SuperSQL質問文）

```
GENERATE xml
 [ [ { p.Title@{name=タイトル},
       [a.Name@{name=名前}, a.Affiliation@{name=所属, att=名前}
       ]!@{tag=著者}
     }@{tag=論文}
  ]!@{tag=論文リスト} ,
  [ { b.Title@{name=タイトル}, b.Author@{name=著者, att=タイトル}
    }@{tag=本}
  ]!@{tag=本リスト}
 ]!@{tag=リスト}
FROM  Author a, Authoring at, Paper p, Book b, Author ba
WHERE a.ID = at.AID and p.ID = at.PID and b.author = ba.ID
```

B.XMLデータのインスタンス

```
<?xml version="1.0" encoding="Shift_JIS"?>
<リスト>
  <論文リスト>
    <論文>
      <タイトル>SilkRoute</タイトル>
      <著者>
        <名前 所属="AT&T Labs-Research">Mary Fernandez</名前>
        <名前 所属="Univ. of Pennsylvania">Dan Suciu</名前>
        <名前 所属="AT&T Labs-Research">Wang-Chiew Tan</名前>
      </著者>
    </論文>
      …
  </論文リスト>
  <本リスト>
    <本>
      <タイトル 著者="S.Abiteboul">Data on the Web</タイトル>
    <本>
      …
  </本リスト>
</リスト>
```

図6 XMLビューの定義
Fig. 6 The definition of XML view.

図6-AのSuperSQL質問文のTFEからは，図6-Bのインスタンスとともに**図7**のDTDが生成される．

4.3.2 データソースを利用した構造情報の生成

上記の（4.3.1項）の方法では反復子はつねに‘*’に変換されるが，データの構造制約としてこれでは緩やかすぎる場合がある．そこでXMLデータを生成する元となったデータソースの情報を元に構造情報を生成する．たとえば，データベース内のデータを分析して反復するデータの数等の特徴を調べ，その特徴に即した構造情報を生成する☆．

詳細な構造情報を調べたいTFEオブジェクトに対

☆ 抽出される構造情報はデータベース内のデータの状態により変化する．

```
<!ELEMENT リスト (論文リスト, 本リスト)>
<!ELEMENT 論文リスト (論文)*>
<!ELEMENT 論文 (タイトル, 著者)>
<!ELEMENT タイトル (#PCDATA)>
<!ELEMENT 著者 (名前)*>
<!ELEMENT 名前 (#PCDATA)>
<!ATTLIST 名前 所属 CDATA #REQUIRED>
<!ELEMENT 本リスト (本)*>
<!ELEMENT 本 (タイトル)>
<!ELEMENT タイトル (#PCDATA)>
<!ATTLIST タイトル 著者 CDATA #REQUIRED>
```

図 7 TFE より生成した DTD
Fig. 7 DTD derived from TFE.

しては「sch」オプションを使用する．このオプションの引数の値により，どのように構造情報を調べるかを指定する．

たとえば図 6-A に示した SuperSQL の 'a.Name' に対し詳細な構造情報を生成する，つまり，図 7 の

```
<!ELEMENT 著者 (名前)*>
```

を変更する場合，

```
a.Name@{sch=range, name=名前}
```

等と記述する．

sch オプションの引数に関しては以下のとおりである．

- off ··· 対象に関し，詳細な構造情報は調べない（デフォルト値）．
- ll（lower limit）··· 対象の出現回数の下限値のみ調べ，構造情報へ反映させる．
  例：`a.Name@{sch=ll, name=名前}`
  → 結果が「0 以上」の場合:
  `<!ELEMENT 著者 (名前)*>`
  結果が「1 以上」の場合:
  `<!ELEMENT 著者 (名前)+>`
  結果が「2 以上」の場合:
  `<!ELEMENT 著者 (名前 名前+)>`
- ul（upper limit）··· 対象の出現回数の上限値のみ調べ，構造情報に反映させる．
  例：`a.Name@{sch=ul, name=名前}`
  → 結果が「3 以下」の場合:
  `<!ELEMENT 著者 (名前? 名前? 名前?)>`
- range ··· 対象の出現回数の上限値，下限値を調べ，構造情報に反映させる．
  例：`a.Name@{sch=range, name=名前}`
  → 結果が「2 以上 3 以下」の場合:
  `<!ELEMENT 著者 (名前 名前 名前?)>`

たとえば実際の XML データの出力が

```
<論文>
  <タイトル>XML データの格納</タイトル>
  <著者><名前>M.Akahori</名前>
        <名前>M.Toyama</名前>
  </著者>
</論文>
<論文>
  <タイトル>SuperSQL と XML</タイトル>
  <著者><名前>M.Akahori</名前>
        <名前>T.Arisawa</名前>
        <名前>M.Toyama</名前>
  </著者>
</論文>
<論文>
  <タイトル>SuperSQL の最適化</タイトル>
  <著者><名前>T.Arisawa</名前>
        <名前>M.Toyama</名前>
  </著者>
</論文>
```

となっていた場合，名前要素は著者要素やタイトル要素に対し

- つねに 2 個以上存在（出現数の下限に注目した場合）
- つねに 3 個以下存在（出現数の上限に注目した場合）
- 2 個以上 3 個以下存在（出現数の範囲に注目した場合）

のいずれかであるといえる．

前述の名前要素は，タイトル要素によってグループ化されている．これらの XML データはデータベース内の属性値を射影したものである．このため，名前要素の構造上の特徴を知るにはタイトル要素の元となっているデータベース属性でグループ化した場合の，名前要素の元となっているデータベース属性の出現回数の特徴を分析すればよい．そこで，名前要素の詳細な構造情報を調べるには XML データを生成する元となった SuperSQL 質問文からこれらのデータベース属性について分析する SQL 質問文を生成し，XML データを生成する元となったデータベースを分析する必要がある．この分析用 SQL 質問文では以下を行う．

- 分析対象となる要素をグループ化している要素の元となったデータベース属性でグループ化を行い（Group By 節に記述する），
- グループ化された分析対象となる要素の出現回数を調べる（Count 関数の引数とする）．

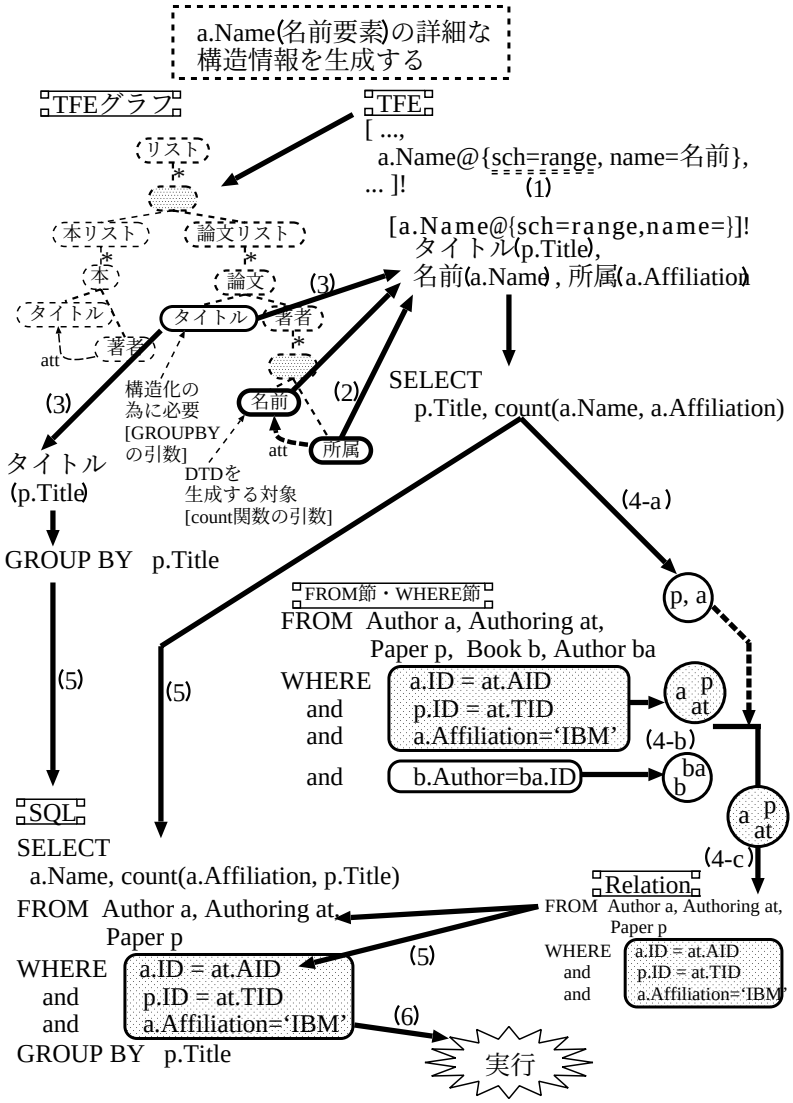

図 8 データソースを利用した構造情報の抽出
Fig. 8 Induction of structural information from data.

分析の手順は以下のとおりである．また，例として図 6-A の SuperSQL 質問文から著者名に関する構造情報を生成するための SQL の生成手順について**図 8** に示す．図中の番号は以下の手順番号を示す．

(1) 分析対象の発見
··· TFE を分析して sch オプションを持つ TFE オブジェクトを発見し，これらに対し以下の作業を行う．

(2) 分析対象と，分析対象とともにグループ化されている TFE オブジェクトの選択
··· XML データを生成する元となった SuperSQL 質問文の TFE から詳細な構造情報を求めたい TFE オブジェクト，およびこれと共通の親 TFE オブジェクトを持ち反復子でない TFE オブジェクトを取得し，Count 関数の引数として分析用 SQL 質問文の SELECT 節に記述する☆．

(3) 分析対象をグループ化（構造化）している TFE オブジェクトの選択
··· すでに選択した TFE オブジェクトの先祖の TFE オブジェクトを親オブジェクトとして持ち，反復子でない TFE オブジェクトを選択し，

☆ なお，自身が子要素を持ち値を持たない（対応するデータベース属性がない）場合は子要素を取得し，これらの情報から構造情報を生成する．

分析用SQL質問文のSELECT節やGroup By節に記述する．

(4) 不要なリレーションの排除，およびFROM節やWHERE節の編集
··· 元となるSuperSQL質問文のFROM節とWHERE節を編集し，分析用SQL質問文のFROM節とWHERE節を生成する．選択したTFEオブジェクトに対応するデータベース属性の属するリレーション，もしくはこれらとWHERE節で結合されている属性を持つリレーション以外は無用な直積を生じるため排除し，その残りを分析用SQL質問文へ記述する．必要なリレーションの選択は以下のとおりである．
(a) すでに選択したTFEオブジェクトに対応するデータベース属性を調べ，必要なリレーション変数を取り出す．
(b) 元となるSuperSQL質問文のWHERE節の結合条件からWHERE節内の条件のグループ分けを行い（あるグループに属する条件と同じリレーション変数を使用する条件は同じ条件グループに組み込む），各々のグループで使用されているリレーション変数を調べる．
(c) (a)のリレーション変数と(b)の条件グループを突き合わせ，必要な条件グループを決定する．

(5) 分析用SQL質問文の実行
··· 以上より分析用SQL質問文を作成し，実行する．

(6) 分析の実行
··· 詳細な構造情報を求めたい分析対象について，分析用SQL質問文の結果からMin，Max等を調査し，その結果に基づいて構造情報を作成する．

4.4 XMLデータ生成の支援

XMLデータを生成する際には，あらかじめ出力するべきXMLの構造が規定されている場合がある．このため，既存の文書構造に対し妥当なXMLデータを生成するための支援機能が必要となる．支援機能としては

- 既存の文書構造情報を利用したSuperSQL質問文の生成
- 出力したXMLデータと既存の文書構造との妥当性の確認

等が考えられる．

- 既存の文書構造情報を利用したSuperSQL質問文の生成
XMLデータを出力するためのSuperSQL質問文を作成するには
(1) 必要なデータベース属性の選択（WHERE節）
(2) データベース属性とXMLの要素の射影の定義（TFE内のデータベース属性の記述，nameオプションの使用）
(3) データの構造化の定義（TFEにおける反復子や結合子等の記述）
といった作業が必要である．このうち，(2)の作業はデータベース側のスキーマに依存し，どのような射影を定義するか（どのデータベース属性とXMLの要素を結び付けるか）はユーザの意思が介在するため，この作業も自動化できない．一方，(3)の作業も文書構造情報ではなくユーザの意思により決定されるものであるため自動的に行うことはできない．しかし，(1)の構造化の定義には既存の文書構造情報が存在すればその情報に従う必要があり，この情報の利用が可能である．
したがって，(1)の作業を自動的に行ってSuperSQL質問文のテンプレートを生成し，その質問文テンプレートを変更することにより(2)，(3)の作業を行う，といったSuperSQL質問文の半自動生成が可能となる．この際元となる文書構造情報はDTDを利用する．なお，現在このDTDには「| (alternative) は使用できない」という制約が存在する．
DTDからSuperSQL質問文を半自動生成する手順を以下に示す．このうち，(1)，(2)は自動的に行う．
(1) DTDの構造情報を解析する．
(2) 解析した情報を元にTFEのテンプレートを生成する．
(3) データベース属性をテンプレートに書き込む．
(4) GENERATE宣言，FROM節，WHERE節を書き加える．
図9 にSuperSQL質問文の半自動生成の例を示す．図中の番号は上記の手順の番号である．
- 出力したXMLデータと既存の文書構造との妥当性の確認
文書構造がファイルに記述されている場合，その文書構造ファイルを利用して出力したXMLデータの妥当性を確認できる．このためには以下のオ

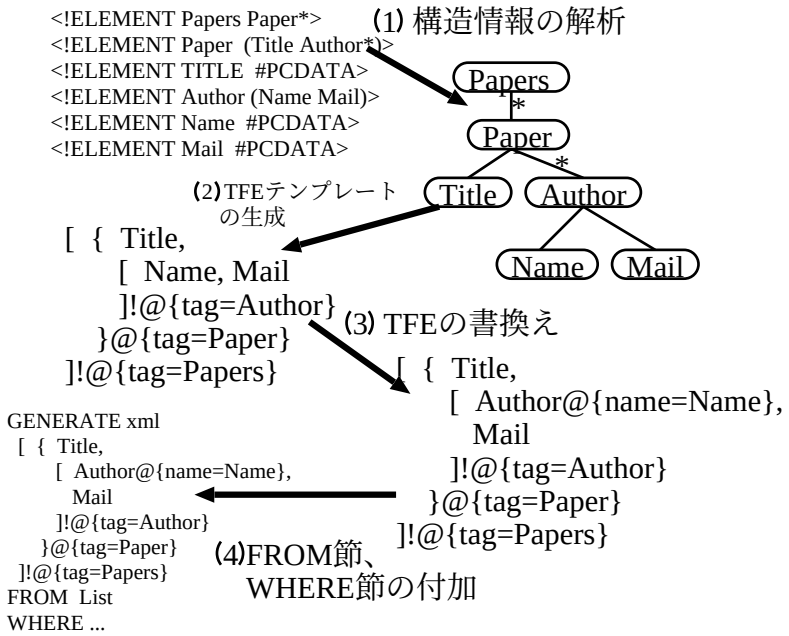

図 9 DTD を利用した SuperSQL 質問文の半自動生成
Fig. 9 Semi-auto generating SuperSQL query using DTD.

プションを最外の反復子へ加える．

– valid ··· 引数に示された文書構造ファイルを利用して，出力となる XML データの妥当性を検証する．

なお，現在引数として利用できる文書構造は DTD のみである．

5. XML データビューに対する質問処理

XML データビューを定義する SuperSQL 質問文が評価されると関係データベース内のデータを XML データ文書として出力できる．評価していない SuperSQL 質問文はデータベース内のデータの仮想ビューと見なせる．この XML データの仮想ビューに対する XML 質問文による問合せは，問合せ結果に対応する SuperSQL 質問文に変換され処理される．この結果，ユーザに XML データの実体化ビューとしての XML データ文書，または XML データの仮想ビューを返す．

本システムでは，XML データビューに対する問合せに使用するユーザ言語として XQL を用いる．XQL による XML データビューへの問合せを行うには以下の手順を踏む（**図 10** 参照）．

(1) ビューを定義する SuperSQL 質問文から構造情報を取得．
(2) 構造情報に基づき XQL を SuperSQL 質問文に変換．
(3) SuperSQL 質問文から SQL，構造化情報（および DTD，RELAX，XSL）等を生成．
(4) SQL を実行し，データタプルを得る．
(5) 構造化情報に従いデータを構造化．
(6) 構造化したデータから XML データを生成．
(7) ユーザに問合せ結果として XML データ（および DTD，RELAX，XSL）を返す．

5.1 XQL（User Query）

XQL は 1998 年に提案された XML 文書のための汎用的な質問言語で，パス式を基本とした簡潔な記述で要素の抽出を行う．XQL の基本的な問合せは ‘/’ または ‘//’ の経路演算子と要素型をつないだ文字列で表現する．属性名を示す際には ‘@’ を属性名の前につけた文字列で示す．また，条件式は条件内で使用する相対パスの起点となる要素型の直後の ‘[ ]’ 内に記述する．‘[ ]’ 内で ‘/’ から始まるパスはルート要素からの絶対パスを表す．問合せ結果はパス式の末尾の要素型のノードの集合である．主な機能を**表 1** に示す．

5.2 XQL による問合せ

XML データビューに対する質問文は，結果に対応する SuperSQL 質問文へ変換され，問合せの処理がなされる．質問文変換の流れは以下のようになる．

(1) XQL に対応する XML パスから TFE パスを選択し，これに基づき新たな TFE を生成する．
(2) 生成した TFE 内に見受けられる重複等を削除

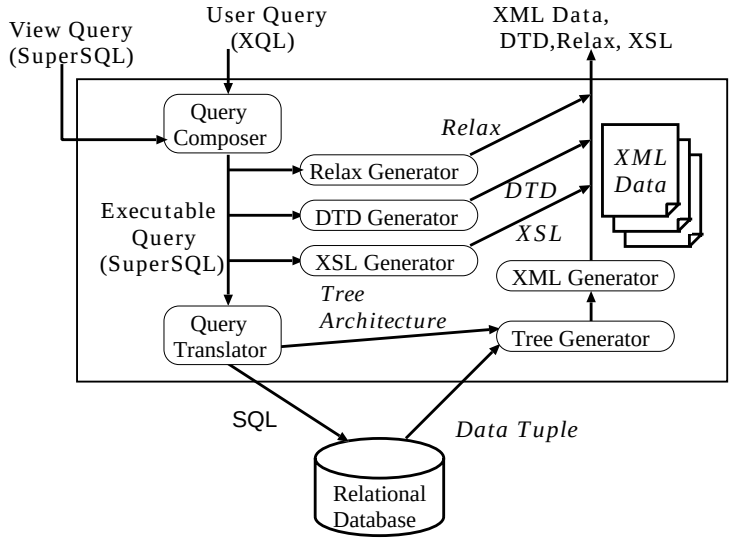

図 10 XMLデータビュー生成システム
Fig. 10 XML data view generating subsystem.

表 1 XQLの基本的な用語や演算子
Table 1 The basic terms and operators in XQL.

Feature	Example
Element name	author
Wildcard as element name	*
Attribute name	@id
Wildcard as attribute name	@*
parent/child	author/first-name
ancestor/descendant	invoice//product
filters	author [first-name='bob']
union	a union b
shallow return operation(?)	/auhor?/first-name
deep return operation(??)	/auhor??/first-name

する（TFEの簡約化）.

(3) XQLのフィルタリングの条件に対応する条件リレーションを，FROM節やWHERE節に追加する.

(4) 不要なリレーションを排除し，FROM節やWHERE節を編集する.

(5) 新たに生成したTFE，FROM節，WHERE節をまとめ，新しい実行形式のSuperSQL質問文を生成する.

以降では4章に示したXMLデータビューに対し

```
/リスト/論文リスト/論文/著者/名前[@所属='IBM']
```

の問合せを行った例を中心に解説する.

5.2.1 XMLデータビューの構造情報の取り出し

SuperSQL質問文中のTFEはデータを構造化し，XMLデータの構造を規定する．このTFEから，TFE内での各オブジェクトのパス（TFEパス）と出力のXMLデータの各要素のルートからのパス（XMLパス）の対応情報等をXMLデータビュー生成時にあらかじめ取り出しておく．4.1.2項で述べたとおり，TFEとXMLは属性に関してその構造が異なる.

5.2.2 XQLに基づくTFEの変換

以下の手順に基づき，ビューを定義したSuperSQL質問文中のTFEからXQLに従ったTFEを生成する．また，例として図6-AのSuperSQL質問文に対し

```
/リスト/論文リスト/論文/著者/名前[@所属='IBM']
```

という問合せを行った際のTFEの生成の様子を**図11**に示す．図中の番号は下記の手順番号を表す.

(1) XQLに対応するXMLパスを取得.

(2) XMLデータビューの構造情報に基づきXMLパスに対応するTFEパスを取得.

(3) 取得したTFEパスで示されるTFEオブジェクトを選択☆.

(4) 構造化に必要なTFEオブジェクトを選択.
… すでに選択したTFEオブジェクトの先祖のオブジェクトを親とする，反復子ではないTFEオブジェクトを選択する．このTFEオブジェクトは出力しないのでNULL関数を使用する.

(5) 選択したTFEオブジェクトから新たなTFEを生成.

☆ リターン演算子が使用されていれば，その要素に対応するオブジェクトも選択する.

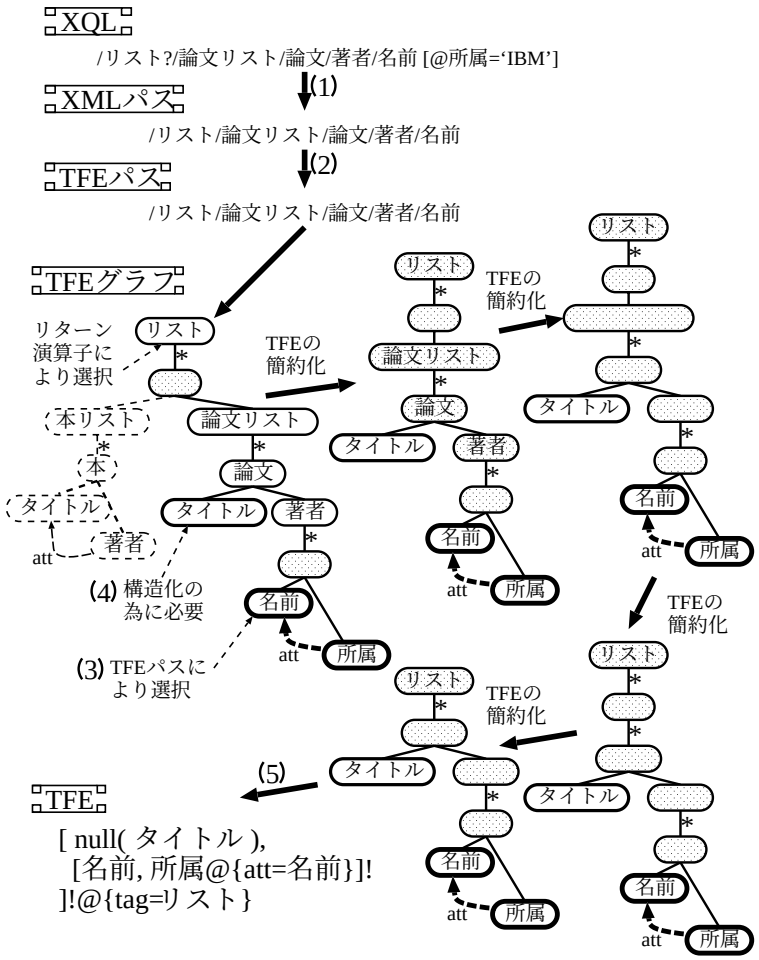

図 11 XQL による TFE オブジェクトの選択と簡約化
Fig. 11 Selection and reduction of TFE objects using XQL.

5.2.3 TFE の簡約化

XQL から TFE を生成した結果，重複した構造指定をしていることがある．たとえば反復子が直列の場合（例：'[[ ],]!'）は，内側の反復子は 1 つの集合を外側の反復子へ渡すため外側の反復子による反復は起こらず，構造化やレイアウト指定に貢献しない．このため反復子の重複がある場合には外側の反復子を省略する（図 11，**図 12** 参照）．

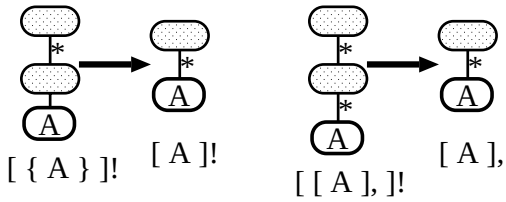

図 12 TFE の簡約化
Fig. 12 Reduction of TFE.

5.2.4 XQL の条件による FROM 節や WHERE 節の編集

XQL は '[ ]' の利用により，フィルタリングの条件を指定する（表 1 参照）．ここで使用する要素名は，'[ ]' が付加されるエレメントを基点とする相対パスで示す．

たとえば図 1-A1 の関係データから図 1-C1 の XML データを生成する SuperSQL 質問文

```
GENERATE xml
  [ { Title, [ { Author@{name=Name}, Mail
                 }@{tag=Author}
               ]!@{tag=Authors}
    }@{tag=Paper}
  ]!@{tag=Papers}
FROM List
WHERE ...
```

に

```
(1) /Papers/Paper[Authors/Author/Name='Rowe']
                    /Authors/Author/Name
```

という XQL（Name 要素が ‘Rowe’ である Paper の Name 一覧を示す）で問い合わせるとする．

XQL より問合せ結果を示す SuperSQL 質問文を生成する際には，フィルタリングの条件を変換し，WHERE 節に付加するが，単純に元の XML データを定義していた SuperSQL 質問文に ‘Name 要素に対応するデータベース属性 (Author)=Rowe’ を WHERE 節に付加するとすべての Name 属性の値が ‘Rowe’ となってしまう．つまり，

```
{ Postgres..., Rowe, Rowe@CS.Berkeley.Edu }
{ Data Base..., Rowe, Rowe@CS.Berkeley.Edu }
```

という組しかデータベースからの問合せ結果にしか現れず，論文の作者が複数いるという情報が失われてしまう．これを避けるため，WHERE 節には入れ子の副質問文を付加する．

Name 要素をグループ化しているのは Title 要素である．XQL(1) の結果は「著者が ‘Rowe’ である論文のタイトルを書いている著者全体」であるので，タイトルに対応するデータベース属性（Title）の値が，著者が ‘Title’ である論文のタイトル一覧に，つまり

```
SELECT Title
FROM   List
WHERE  Author='Rowe'
```

の問合せ結果に含まれればよい．そこで上記の SQL を副質問文として

```
GENERATE xml
  [ { Title, [ { Author@{name=Name}, Mail
               }@{tag=Author}
             ]!@{tag=Authors}
    }@{tag=Paper}
  ]!@{tag=Papers}
FROM List
WHERE ...
  and Title IN (  SELECT Title
                  FROM   List
                  WHERE  Author='Rowe'
                )
```

という条件を元の XML データビューに付加する．

問合せ結果となる論文要素の子孫には値が ‘M. Akahori’ である名前要素を含むだけでなく同じ論文を書いたすべての著者の名前情報を含む．

ただし，

```
(2) /Papers/Paper/Authors/
                Author[Name='Rowe']/Mail
```

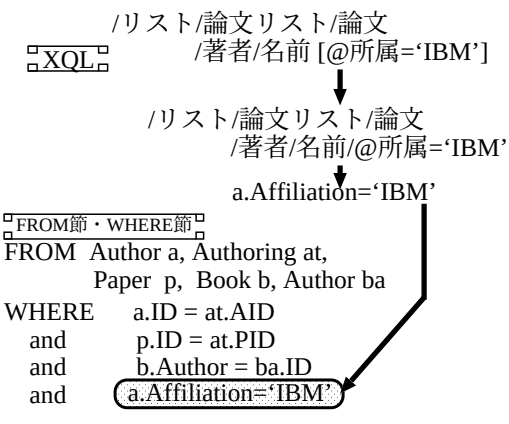

図 13 フィルタリングの条件の処理
Fig. 13 Processing of filtering conditions.

のように，条件に使用されている要素や属性と 1 対 1 の要素しか問合せ結果とならない場合には入れ子の副質問文を使用しなくてもよい．4 章に示した XML データビューに対し

```
(3) /リスト/論文リスト/論文/著者/名前 [@所属='IBM']
```

の問合せを行った場合も同様で，この問合せによる処理を**図 13** に示す．

5.2.5 XQL による FROM 節や WHERE 節の編集

XQL に従い TFE を変換した結果，問合せ結果の TFE にその属性が現れなくなるリレーションがある．出力，あるいはデータの構造化に使用する，もしくは XQL 中の条件で使用されている要素に対応しているリレーションは，質問文の実行に必要である．しかし，これらのリレーションに含まれず，またこれらと WHERE 節で結合されていないならば，無用な直積を生じてしまうのでこれを排除する．この際の処理の手順を以下に示す．また，例として図 6-A の SuperSQL 質問文に対し

```
/リスト/論文リスト/論文/著者/名前 [@所属='IBM']
```

という問合せを行った際の TFE の生成の様子の続きを**図 14** に示す．図中の番号は下記の手順番号を表す．

(1) 問合せ先となる XML データビューを定義している SuperSQL 質問文から XQL の問合せ結果となる TFE を生成する．
(2) 生成した TFE から必要なリレーション変数を取り出す．
(3) XQL の条件部分を取り出し，データベース属性を利用した条件に変換する．
(4) 変換した条件から必要なリレーション変数を取り出す．
(5) XML データビューを定義している SuperSQL

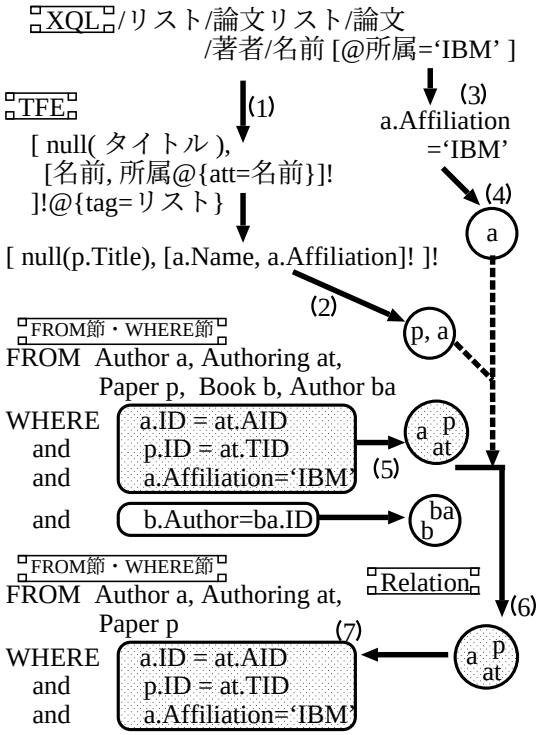

図 14 TFE に基づくリレーションの選択
Fig. 14 Selection of relations based on TFE.

質問文の WHERE 節の条件を結合条件を用いてグループ分けし，リレーション変数群を取り出す．

(6) 抽出したリレーション変数とグループ分けしたリレーション変数群を突き合わせ，必要なリレーション変数群を決定する．

(7) 必要なリレーション変数群に基づき，FROM 節と WHERE 節を編集する．

5.2.6 XQL 問合せによる XML データビューの生成

以上の手法を用い，XQL を元に XML データビューを定義する SuperSQL 質問文の TFE や FROM 節，WHERE 節を編集し，実行形式の SuperSQL 質問文を生成する．この SuperSQL 質問文を評価すれば実体化ビューとしての XML データを生成し☆，評価しなければ XML データの仮想ビューを生成する．

5.3 XQL 問合せによる構造情報の生成

問合せの XQL 質問文から生成した SuperSQL 質問文の TFE から DTD や RELAX 等の構造情報を生成する．また，必要な要素に対しては問合せ結果の XML データの元となるデータを分析し，より詳細な構造情報を生成する（4.3 節参照）．

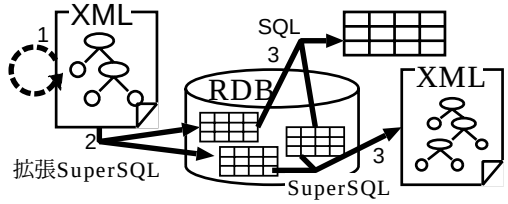

図 15 XML データの格納
Fig. 15 Storing XML data.

6. 拡張 SuperSQL 質問文による XML データの格納

受け取った XML データを関係データベース内の既存のデータとともに SQL 等で使用するためには，SuperSQL 質問文とは逆に XML データを関係データとして格納する必要がある．このための変更を加えた SuperSQL 質問文を拡張 SuperSQL 質問文と呼ぶ．拡張 SuperSQL 質問文により XML データを新規もしくは既存のリレーションに分割し格納することができる．

SuperSQL は本来，関係データベース内のデータを木構造データへ構造化して様々な媒体の応用データを生成する．我々はこの関係を逆に利用して XML データを関係データに変換し，関係データベース内の新規もしくは既存のリレーションへ格納可能なように SuperSQL を拡張した（**図 15** 参照）．格納する XML データはデータベース内の既存のデータと同種であれば既存のリレーションに追加し，異なるデータであれば新規リレーションを生成する．データの不一致（異種性）に関してはオプションや関数，前処理で対応する．この機能を利用した大まかな使用方法は以下のとおりである．

(1) データを格納しやすい形に加工する☆☆．

(2) 拡張 SuperSQL 質問文により XML データを複数リレーションに分割し格納する．

(3) 格納したデータと既存のデータは区別なく利用可能となり，SQL や SuperSQL 等で利用できる．

6.1 質問文の構文

拡張 SuperSQL 質問文は従来の SuperSQL 質問文とは異なり，XML データから関係データを生成する問合せ言語である．このため，FROM 節では XML データ内の要素を指定するために用い，GENERATE 節で

☆ 結果は 'xql:result' 要素の下に格納される．

☆☆ この処理は必須ではない．

表 2 拡張 SuperSQL のオプション
Table 2 Options in Extended SuperSQL.

オプション名	対象	動作
xmldistinct	リレーションを生成する内側の反復子	引数が on の場合，同値のオブジェクトを単一のオブジェクトと見なす.
dbdistinct	リレーションを生成する内側の反復子	引数が on の場合，挿入先に存在しないデータしか格納しない.
relation	リレーションを生成する内側の反復子	データを挿入するリレーションを指定.
name	XML 要素	データベース属性名を指定（指定しなければ XML 要素名を流用）.
type	XML 要素	格納した際のデータ型を指定.
size	XML 要素	格納した際のデータ型のサイズを指定.
value	KEY_GEN 関数，XML 要素	対象の値を変数で束縛する．この変数を他の場所で使用することで参照関係を生成する.
func	KEY_GEN 関数	KEY_GEN 関数で呼び出す外部関数を指定（現在の仕様では，JAVA で直接関数定義を行う）.

表 3 拡張 SuperSQL の関数
Table 3 Functions in Extended SuperSQL.

関数名	引数	動作
KEY_GEN	XML 要素のリスト	外部関数を呼び出し，キーの生成を行う（これはスコーレム関数である）.
verb	文字列	固定の文字列を挿入.

は関係データを生成し☆，GENERATE 節では ‘GENERATE’ の後には ‘relation’ しか記述しない．また，関係データの挿入先を指定する INTO 節を加えた.

GENERATE 節では TFE と経路式で XML データと関係データベース間のマッピング（写像）を行う．TFE では入れ子の内側の反復子（[ ]）でリレーションを表し，その名前は relation オプションで指定する．構成するデータベース属性は変数（$ で始まる）とパス式による XML データの要素（属性は -> でアクセス）のリストで表現する．FROM 節では元となる XML データと経路パスの指定を，INTO 節では格納先データベースとリレーションについての指定を変数に代入する．変数の値は TFE や WHERE 節で使用する．SQL の INTO 節と異なり，拡張 SuperSQL 質問文の INTO 節は挿入先として複数のリレーションを指定可能である．なお，関係データを挿入するデータベースは JDBC プロトコル（*jdbc:* ⟨*subprotocol*⟩*:*⟨*subname*⟩）を用いて指定する．WHERE 節では条件を指定する．また，質問文では様々な関数やオプションが指定可能である（**表 2**，**表 3** 参照）.

例として，図 5 に示すリレーションに XML 化された SIGMOD Record の XML データ[33]（**図 16** 参照）を挿入する．このための拡張 SuperSQL 質問文を**図 17** に示す．また，これにより格納されたデータの一部を**図 18** に示す.

☆ したがって，複雑な構造化能力は必要でない.

6.2 データの平坦化と分解

本格納システムでは，元となる XML データより

(1) パス式を用いて必要なデータ（要素の集合）を抽出,
(2) 同じ，もしくは変数により関連付けられた一連のリレーションに格納するデータを平坦化,
(3) 各リレーション構造に合わせてデータを射影し，分解して格納,

を行う.

本格納システムでは格納するべき要素を指定する際にはパス式を基本として要素を取得する．複数のパス式で指定された要素群の共通の親要素があった場合，その要素に囲まれた要素群は 1 つの情報単位の候補と見なされる．FROM 節の変数による要素の階層的な束縛を利用し，宣言した要素を情報単位として扱う．宣言されなかった共通の親要素は存在しない要素として扱う.

情報単位内の要素の対応は組として取り出される．要素の組は情報単位内の要素の直積となる．この結果，情報単位内の対応する要素群は 1 つ，もしくは複数のタプルとして関係データに変換される．たとえば**図 19**-A で情報単位を People，Person 要素にして ‘Person.Name’，‘Person.Mail’ の 2 つのパス式で Name 要素と Mail 要素を指定した場合，取り出される組は図 19-(1) のようになる．一方，図 19-

```
<SigmodRecord>
 <issue>
  <volume>11</volume>
  <number>1</number>
   <articles>
    <article>
     <title>Architecture of Future Data Base Systems.</title>
     <initPage>30</initPage>
     <endPage>44</endPage>
     <authors>
      <author position="00">Lawrence A. Rowe</author>
      <author position="01">Michael Stonebraker</author>
     </authors>
    </article>
    ...
    <article>
     <title>The Design of Postgres.</title>
     <initPage>340</initPage>
     <endPage>355</endPage>
     <authors>
      <author position="00">Michael Stonebraker</author>
      <author position="01">Lawrence A. Rowe</author>
     </authors>
   </article>
    ...
</SigmodRecord>
```

図 16 SIGMOD record の XML データ
Fig. 16 XML data of SIGMOD records.

A で People 要素を情報単位とした場合は情報単位として宣言されなかった Person 要素は無視されて図 19-B と同様に扱われ，Name 要素の値 {M.Akahori, T.Arisawa} と Mail 要素の値 {aka@db.ics.keio.ac.jp, ari@db.ics.keio.ac.jp} の直積である図 19-(2) の組が取り出される．図 19-B において要素の順序に意味があり連続する Name 要素と Mail 要素が組となる場合，つまりこのデータから図 19-(1) の組を取り出したい場合，XSL 等で前処理をして情報単位の候補となる要素を追加して図 19-A のようにする．また，図 19-C で article 要素と authors 要素を情報単位として ‘article.title’，‘article.authors.author’ の 2 つの経路式で title 要素と author 要素を指定した場合，図 19-(3) のような組が取り出される．

この結果，要素の兄弟間の順序は無視される．このため，たとえば図 19 の A と A’ で情報単位を People 要素と Person 要素とした場合，情報単位の中で Name 要素と Mail 要素の順番が異なっても等価な情報と見なして同じ組の集合の図 19-(1) が関係データベースに格納される☆．

たとえば SIGMOD Record の XML データに図 17 を適用すると，図 19-C のような木構造データが得られ，これを平坦化する．その後各リレーションごとに分解して図 18 の結果を得る☆☆．キーや参照関係はこの平坦化された表から生成される．分解の際には単なる射影を行うのではなく，同じデータが平坦化されたためにできた複数のタプルは 1 つのタプルに戻る．ま

☆ これらの XML データを変換した関係データから SuperSQL を用いて元の XML データへ生成し直す場合，要素の順番は SuperSQL の TFE で指定した順番に固定される．

☆☆ Paper リレーションの Source の値 ‘SIGMOD Record’ は verb 関数により挿入される．

```
GENERATE relation
 [ [ KEY_GEN($IA.title)@{name=ID,
         value=$PID, type=int, func=IAID()},
     $IA.title@{size=50}, verb(SIGMOD Record)
   ]!@{relation=$DP, xmldistinct=on, dbdistinct=on},
   [ KEY_GEN($IAA.author)
       @{name=ID, value=$AID, type=int, func=AID()},
     $IAA.author@{size=50, name=Name}
   ]!@{relation=$DA, xmldistinct=on, dbdistinct=on},
   [ $AID@{name=AID}, $PID@{name=PID}
   ]!@{relation=$DAT, dbdistinct=on}
 ]!@{database=xml}
FROM  SigmodRecord.xml $X, $X.SigmodRecord.issue $I,
      $I.articles.article $IA, $IA.authors $IAA
INTO  jdbc:postgresql://postgres/LIST $D,
      $D.Paper $DP, $D.Author $DA, $D.Authoring $DAT
WHERE $I.volume >= 11;
```

図 17 拡張 SuperSQL 質問文例
Fig. 17 An example of Extended SuperSQL query.

Paper

ID	Title	Source
P1	The Design of Postgres	SIGMOD Record
P2	Data Base Systems	SIGMOD Record

Authoring

AID	PID
A1	P1
A2	P1
A1	P2
A2	P2

Author

ID	Name	Affiliation
A1	Rowe	—
A2	Stonebraker	—

図 18 格納されたデータ
Fig. 18 Stored data.

た，xmldistinct オプションがあると同じ値のタプルは 1 つのタプルにまとまる．

6.3 XML データの格納における問題点

XML データを関係データとして格納する際の問題点として，以下があげられる．

(1) キー値の生成
(2) 参照関係の生成
(3) 同一オブジェクトの同定（XML データ内，XML データと既存の関係データ間）
(4) 格納する XML データと既存の関係データの不一致（名前，データ項目，データ型）

(1)〜(3) は互いに強く関連した，オブジェクトに関する問題であり，本格納システムではデータを分割する際にユーザの関数やオプション指定に従って対処する．

6.3.1 キー値の生成

キーにあたる ID 属性が存在しない XML データを格納する場合にはキーの生成が必要であり，本システムでは KEY_GEN 関数でキーの生成を行う．これはユーザが定義した外部関数を呼び出すこともできる．ユーザはデータの値やデータベース内の情報を用いて多様なキーや既存のキー値を使用しない等の整合性のとれたキーの生成を行う．

6.3.2 参照関係の生成

XML データを複数のリレーションに分解する場合には，参照関係の生成が重要である．参照関係は，主キーを生成する KEY_GEN() 関数を value オプションで '$' から始まる変数へ束縛し，この変数を他の場所で使用することで表現する．

6.3.3 同一オブジェクトの同定

複数のリレーションへ分解する際，たとえば同じ著者が複数の論文を書いている場合等では同一視できる（すべてのデータが一致する）オブジェクトが複数作られ，同一オブジェクトの同定が重要である．また，関係データベース内の既存のデータと重複する場合もあり，XML データ内もしくは XML データと既存データ間で，同一オブジェクトの統合や同一視，あるいは

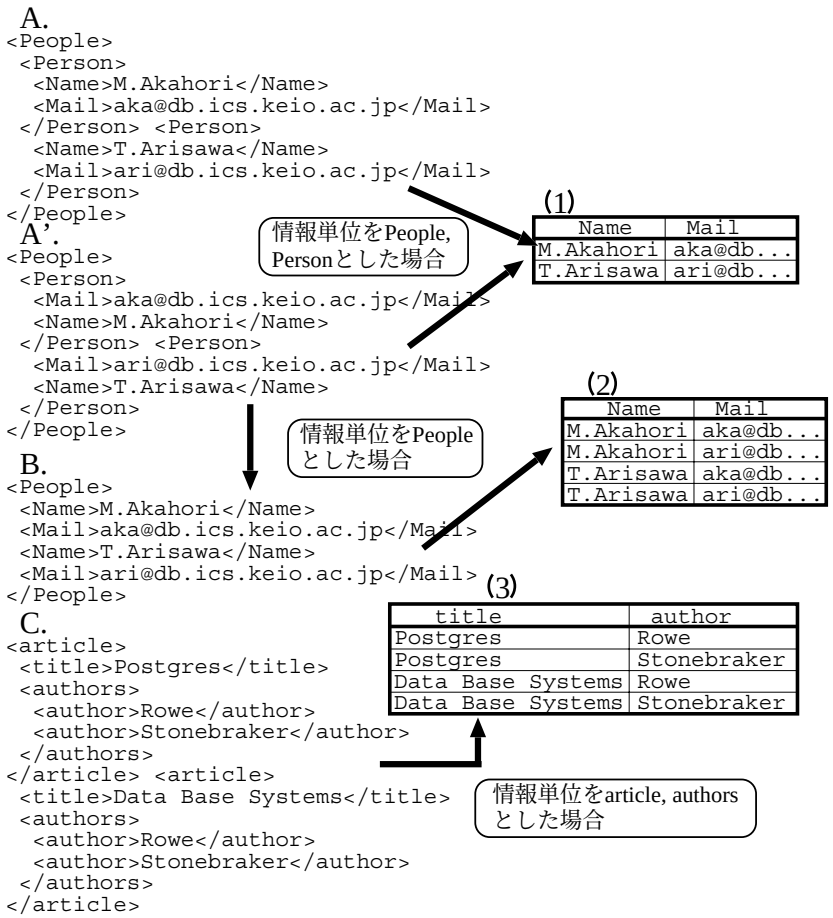

図 19 XML データの平坦化
Fig. 19 Un-nesting XML data.

区別するかの選択が必要である．区別して格納する場合はすでに使用されていないキーの生成が必要である．

この問題に対し，本格納システムでは xmldistinct オプションと dbdistindt オプションにより対処する．これにより，同じ値の（分解された）XML データや関係データを同定し同じオブジェクトとして扱う．

KEY_GEN 関数が指定されていた場合，決定したオブジェクトに対してキーが生成される．ただし，db-distinct オプションが指定されていて，すでに存在する関係データと同じであると同定されたオブジェクトに関してはキーの生成は行われず，また関係データベースへの挿入は行われない．XML データから生成されたオブジェクトと既存の関係データとの同定の際には，データベース中の組の中の NULL 値は単なる値として扱われる．たとえば，{ 姓，名，年齢 } という属性を持つリレーションで { 赤堀，正剛，– }（– は NULL 値を示すとする）という組が存在した場合を想定する．XML データより { 赤堀，正剛，24} というオブジェクトが生成された場合は先の組には一致せず，新たな組としてがデータベースへ格納される．一方，{ 赤堀，正剛，– } というオブジェクトが生成された場合には先の組と一致し，データベースには挿入されない．

6.3.4 格納する XML データと既存の関係データの不一致

元となる XML 要素名と格納側のデータベース属性名が異なる場合には，name オプションで指定して合わせる．

XML 要素とデータベース属性の構成の不一致については

(1) KEY_GEN 関数で呼び出される外部関数を拡張し，キーを生成する代わりにデータの構成を変換する，
KEY_GEN 関数は複数の引数（要素の値）を

取ることができ，これらの値や出現した順序を利用してキー値を生成する．これを利用し，複数の要素の値から 1 つのデータベース属性値を生成する等の属性の構成の単純な変換が可能である．たとえば,「姓」要素と「名」要素の値を単純に結合し,「姓名」データベース属性の値として生成することができる，

(2) 本格納システムを使用する前にデータを前処理して対処する，

等の方法が考えられる．

データ項目の不一致に関しては，格納するデータが既存のデータに対して足りない項目があればそれは null 値となり，また，余分であれば格納しない，もしくは新規のリレーションを生成してそれに格納する．

XML データ内の要素の値はどのようなデータ型であっても基本的にはテキスト型として扱われている．このため，関係データとして格納する際には本来のデータ型に変換する必要がある．このようなデータ型の異種性に関しては，type オプションによるデータ型の指定により対処する．大きさの概念のあるデータ型に対しては，さらに size オプションにより大きさを指定する．

7. 本システムにおける各機能の連携

7.1 XML データと関係データベース内の既存のデータの統合利用

格納した XML データと関係データベース内の既存のデータは区別なく，関係データとして使用することができ，2 種類のデータの統合利用が可能である．SQL 等で使用するほか，アプリケーション等で利用したり，SuperSQL で XML データとして出力したりすることもできる[12]．また，受け取った XML データと関係データベース内の既存のデータから新たな XML データビューを生成して第三者に提供することが可能となる（図 2 参照）．

7.2 格納用の拡張 SuperSQL 質問文の自動生成

格納する XML データと同構造の XML データを出力する SuperSQL 質問文がある場合は，それを元にデータ格納のための拡張 SuperSQL 質問文を生成する．以下にその手順を示す．

(1) 格納する XML データと同構造の XML データを出力する SuperSQL 質問文から，データベース属性とそれに対応する要素の経路式を抽出する．

(2) 経路式を情報単位に従って分解し，変数を割り当てる．

(3) 取り出されたデータベース属性をリレーションごとにまとめ，そのリレーションをまとめるデータベースオブジェクトを付加する．この構造が格納用の拡張 SuperSQL 質問文の TFE の構造を示す．

(4) リレーションやデータベースオブジェクトに変数を割り当てる．

(5) データベースオブジェクト以下にあるリレーションやデータベース属性に対し，変数を適用する．

(6) 格納用の拡張 SuperSQL 質問文の TFE を生成する．

なお，どの要素を情報単位とするかは XML データを出力する SuperSQL 質問文の情報単位にしたい { } や反復子に infounit オプションを指定する．引数が 'on' である { } や反復子に対応する要素を情報単位と見なすように処理する．

著者情報の XML データビューを定義する SuperSQL 質問文から格納用拡張 SuperSQL 質問文を生成する例を**図 20** に示す．図中の番号は上記の手順を示す．

8. 評価・検討

本システムは Java1.2 で実装し，UltraSPARC-II（248 MHz）+Memory 256 MB を使用し，SunOS 5.6 上で PostgreSQL7.0 を用いて評価実験を行った☆．

8.1 XML データの格納

6.1 節に示した格納を実際に行った．SIGMOD Record のすべての XML データを格納すると，論文数約 1,500 本，著者数約 2,000 人，論文/著者関連約 3,700 件，計約 7,200 件のデータの挿入が必要となる．**図 21** にそのうちの一部（$200, 400, \cdots, 1,600$ 件）を格納した際の実験結果を示す☆☆．

拡張 SuperSQL 質問文のパーズは約 0.12 秒とほぼ無視できる時間で処理している．また，XML データをパーズして関係データベースに格納するために分割したデータを生成する時間は，挿入すべきデータ件数に比例している☆☆☆．結果として全体の処理時間はほぼ挿入するデータ件数に比例している．

実験環境で XML データを PostgreSQL に格納する際には特にタプルの挿入に時間がかかっており，格納

☆ 計測時間には Java VM マシンの起動時間（約 0.65 [s]）を加えていない．

☆☆ すべてのデータを格納する場合，拡張 SuperSQL による処理時間は 41 秒，PostgreSQL によるタプル挿入を加えた格納の総実行時間は約 7.5 分である．

☆☆☆ 実際に格納するにはさらにタプルの挿入を行う時間がかかる．

<<著者情報を定義した SuperSQL 質問文>>

```
GENERATE xml
 [ { a.Name@{name=名前},
     a.Affiliation@{name=所属}
   }@{tag=著者, infounit=on}
 ]!@{tag=リスト}
FROM Author a
```

⟹

<<XML データを格納する拡張 SuperSQL 質問文>>

```
GENERATE relation
  [ [ $XA. 名前@{name=Name},
      $XA. 所属@{name=Affiliation}
    ]!@{relation=$DA}
  ]!@{database=$D}
FROM <XML ファイル名> $X,
     $X. リスト. 著者 $XA
INTO <データベース> $D, $D.Author $DA
```

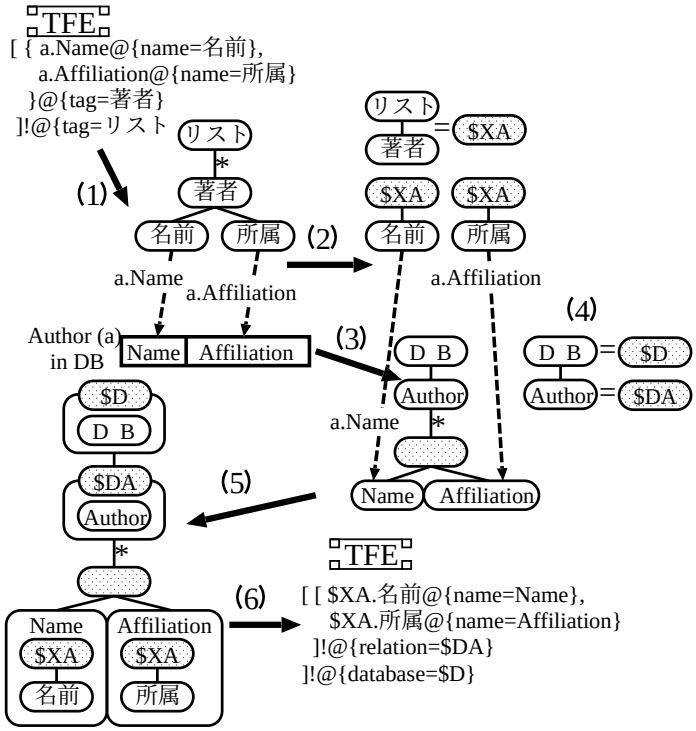

図 20 TFE の自動変換
Fig. 20 Automatic conversion of TFE.

処理全体の実時間の約 95%を占めている．また，挿入データ数が増えるにつれその比率は増大している．

8.2 XML データビューへの問合せ

8.1 節でデータベース内に格納した論文データとデータベース内の既存のデータを利用して図 6 のビューを生成する．論文と本データの総件数の異なる 2 種類の XML データ（ビュー）を用意し，以下の問合せを行う．

(A) //論文 [著者/名前 = 'Jim Gray']/タイトル
（Jim Gray の書いた論文のタイトル一覧が欲しい）

(B) //論文 [著者/名前='Jim Gray']/(タイトル $union$ 著者)
（Jim Gray の書いた論文のタイトルと著者一覧が欲しい）

(C) /リスト/論文リスト/論文/著者/名前 [. = /リスト/本リスト/本/タイトル/@著者] （本と論文両方を書いている人の名前が欲しい）

(D) /リスト/論文リスト （論文リストすべてが欲しい）

実験は本来 SilkRoute や XPERANTO との比較を行うべきだが，2001 年 2 月現在，双方ともソースを公開していないため直接比較を行うことができない．

そこで，本システムを利用した XML データビューへの動的な問合せと本出力システムで生成した静的な XML データへの問合せの比較を行うため，本システムと XQL を Java で実装した GMD-IPSI XQL[34]で比較実験を行う．

GMD-IPSI XQL は

- 通常の XML データファイル
- PDOM（Persistent DOM）··· XML データを解析した DOM をそのまま保存した，インデックスを含むバイナリファイル

の 2 種類を検索対象とできる．このため，本システムは XML データビューを通して関係データベースへ検索する一方で，GMD-IPSI-XQL の検索対象として (a) 通常の XML データファイル，(b) PDOM ファイルの 2 つを用意した．実験結果を**表 4** に示す．

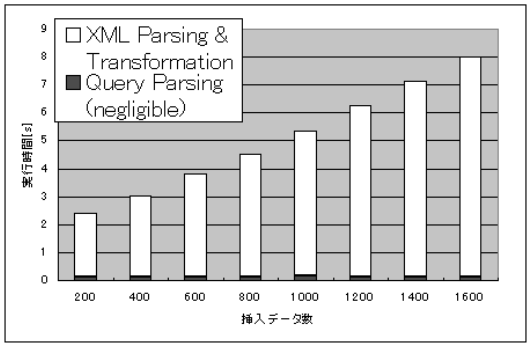

図 21 XML データの格納実行時間
Fig. 21 Performance of storing XML data.

問合せごとにデータソースである XML データをすべて読み込む GMD-IPSI XQL は，問合せの結果の件数と問合せの対象となるデータが大きくなるほど処理時間が長くなる．PDOM ファイルを検索対象とした場合は，処理時間が速くなるがこの傾向は同様である．

一方，本システムは関係データベースをデータソースとしており，データベース内のインデックスの使用が可能で，大量のデータから条件に合うデータを抽出する処理を素速く実行可能である．ビューを構成するすべてのデータをデータベースから取り出すことなく，XQL を変換して直接データベースへ問い合わせることにより，検索対象の件数よりも結果出力のデータ量に処理時間が依存することとなる．

両者を比較すると，本システムは大量のデータから条件付き問合せにより限定された件数の結果を得ることに有効である．

8.3 XML データ生成における問合せ言語の比較

関係データから XML データを生成する際に，本システムでは SuperSQL 質問文を使用する．一方，SilkRoute では RXL を使用する．また，XPERANTO では関係データをいったん同じ構造の XML データに変換し，これに対して XML-QL で問合せを行い，必要な構造の XML データを生成する．本節はこれら 3 つの問合せ言語の比較を行う．

SuperSQL 質問文においては，必要な関係データの選択については FROM 節と WHERE 節に，データの構造化に関しては GENERATE 節の TFE に集約されている．一方，RXL や XML-QL ではデータを構

表 4 XQL 問合せの比較
Table 4 A comparison of querying in XQL.

A：対象データ数約 400 件

質問文	結果件数	本システム	GMD-IPSIXQL	GMD-IPSI XQL (PDOM)
A	6	2.618[s]	2.304[s]	1.310[s]
B	67	2.827[s]	2.360[s]	1.297[s]
C	6	2.552[s]	2.677[s]	1.862[s]
D	400	7.961[s]	3.164[s]	1.804[s]

xml=96,625 byte, PDOM=129,884 byte

B：対象データ数約 3,800 件

質問文	結果件数	本システム	GMD-IPSIXQL	GMD-IPSI XQL (PDOM)
A	18	2.910[s]	4.538[s]	4.070[s]
B	105	3.397[s]	4.477[s]	3.990[s]
C	18	2.902[s]	5.692[s]	4.352[s]
D	3,774	21.583[s]	8.617[s]	4.751[s]

xml=451,732 byte, PDOM=634,606 byte

```
<PaperList>
  <Paper>
    <Title>SuperSQL と XML</Title>
    <Author>M.Akahori</Author>
    <Author>T.Arisawa</Author>
    <Author>M.Toyama</Author>
    <Keyword>SuperSQL</Keyword>
    <Keyword>XML</Keyword>
    <Keyword>RDB</Keyword>
  </Paper>
  <Paper>
    <Title>SuperSQL の最適化</Title>
    <Author>T.Arisawa</Author>
    <Author>M.Toyama</Author>
    <Keyword>SuperSQL</Keyword>
    <Keyword>optimize</Keyword>
  </Paper>
</PaperList>
```

図 22 XML データの出力例
Fig. 22 Sample of XML data.

造化するのに入れ子の質問文（ブロック）を使用する．この際，各質問文（ブロック）には各々でデータの選択や構造化を行う．このため，データの選択のための記述と構造のための記述が分散され，ユーザが質問文全体を把握しにくくなっている．また，Skolem 関数の引数（どのような値の組を一意と見なすか）によって出力構造が異なるのだが，この記述も分散してしまい質問文の記述から出力結果の構造を直感的に把握することは難しい．

論文のタイトルごとに著者リストとキーワードリストが存在する，論文の一覧データ（**図 22**）を図 5 から生成する SuperSQL 質問文は，以下のとおりである．

```
GENERATE XML
  [ { p.Title, [a.Name@{name=Author}]!,
      [k.Keyword]!
    }@{tag=Paper}
  ]!@{tag=PaperList}
FROM Paper p, Authoring ap, Author a,
     Keyword k, Word w
WHERE p.id=ap.pid and ap.aid=a.id
  and p.id=k.pid and k.wid=w.id;
```

同様の出力を行う RXL を記述すると，以下のとおりとなる．

```
construct
<PaperList ID=PList()>
  {
    from Paper $p
    construct
      <Paper ID=Bk($p.id)>
        <Title>$p.Title</Title>
        { from Authoring $ap, Author $a
          where $p.id=$ap.pid
            and $ap.aid=$a.id
          construct
            <Author ID=At($p.id,$a.id)>
               $a.Name</Author>
        }
        { from Keyword $k, Word $w
          where $p.id=$k.pid and $k.wid=$w.id
          construct
            <Author ID=At($p.id,$w.id)>
               $w.Keyword</Author>
        }
      </Paper>
  }
</PaperList>
```

XPERANTO では関係データをいったん同じ構造の仮想 XML ビューに変換し，XML-QL で必要な構造の XML へ変換する．図 5 のリレーションと同じ構造の仮想 XML ビューに対する XML-QL による質問文は以下のとおりとなる．

```
CONSTRUCT <PaperList>{
  WHERE <library.Paper.PaperTuple>
          <ID>$pid</>
          <Title>$ptitle</>
        </> IN "postgres:xml:material/library"
  CONSTRUCT
    <Paper>
      <Title>$ptitle</>
      {WHERE
         <library.Authoring.AuthoringTuple>
           <PID>$pid</>
           <AID>$aid</>
         </> IN "postgres:xml:material/library"
         <library.Author.AuthorTuple>
           <ID>$aid</>
           <Name>$aname</>
         </> IN "postgres:xml:material/library"
       CONSTRUCT <Author>$aname</>
      }
```

```
    {WHERE
       <library.Keyword.KeywordTuple>
         <PID>$pid</>
         <WID>$wid</>
       </> IN "postgres:xml:material/library"
       <library.Word.WordTuple>
         <ID>$wid</>
         <Keyword>$keyword</>
       </> IN "postgres:xml:material/library"
     CONSTRUCT <Keyword>$keyword</>
    }
  </>
}</>
```

一方，入れ子の質問文を使用すると if 文のような記述が簡潔となる．たとえば，‘SuperSQL’ という単語を含む論文のタイトルは ‘PaperOfSuperSQL’ 要素の，‘XML’ という単語を含む論文のタイトルは ‘PaperOfXML’ 要素の値として出力するには XML-QL では以下のように記述する．この場合，内側の WHERE 節で束縛した$ptitle の値が内側の CONSTRUCT 節に渡された順番に処理され，渡された$ptitle の値が ‘SuperSQL’ を含めば ‘PaperOfSuperSQL’ 要素として，‘XML’ を含めば ‘PaperOfXML’ 要素として出力される．このため，‘PaperOfSuperSQL’ 要素と ‘PaperOfXML’ 要素は混在する．

```
CONSTRUCT <PaperList>{
  WHERE <library.Paper.PaperTuple>
          <Title>$ptitle</>
        </> IN "postgres:xml:material/library"
  CONSTRUCT
       {WHERE $ptitle LIKE '*SuperSQL*'
        CONSTRUCT <PaperOfSuperSQL>$ptitle</>}
       {WHERE $ptitle LIKE '*XML*'
        CONSTRUCT <PaperOfXML>$ptitle</>}
 }</>
```

これに対し，SuperSQL では以下のように記述する．ただし，同等な記述ではなく，この記述では ‘PaperOfSuperSQL’ 要素と ‘PaperOfXML’ 要素とは完全に別個に処理され，‘PaperOfSuperSQL’ 要素がすべて出力された後に ‘PaperOfXML’ 要素が出力される．

```
GENERATE XML
  [ [s.Title@{name=PaperOfSuperSQL}]!,
    [x.Title@{name=PaperOfXML}]!
  ]!@{tag=PaperList}
FROM Paper s, Paper x
WHERE s.Title like '*SuperSQL*'
  and x.Title like '*XML*'
```

なお，SuperSQL で問合せを行う場合，中間結果として関係どうしの直積演算や不要な組の生成が行われる．本論文の趣旨ではないためこの問題には触れていないが，文献 35) の Outer Join に相当する，質問文を複数の独立な副質問文に分解しこの問題を回避する質問文変換については文献 10) で述べている．

8.4 構造変換言語としての拡張SuperSQL質問文

拡張 SuperSQL 質問文は XML に対する問合せ・構造変換言語の一種と見なせる．この視点から見た場合の，XML-QL，Quilt，YATL，XSLT 等の他の構造変換言語との比較を行う．

代表的な構造変換言語[36),37)]としては以下があげられる．

- XML-QL [23)]
  ··· AT&T 研究所で設計された，XML を対象とした宣言的な構造変換言語．SQL を拡張し，WHERE 節で束縛・選択した要素を CONSTRUCT 節で再構築できる．
- Quilt [31)]
  ··· 比較的新しい言語であり，XPath や XQL の階層化文書のナビゲーションや XML-QL の変数による束縛，SQL の節の組合せ，OQL の関数等いくつかの問合せ言語の影響を受けている．FOR 節と LET 節で要素を変数に束縛し，WHERE 節で条件に合う要素を選択した後，RETURN 節で出力の XML を再構築する．
- YATL [38),39)]
  ··· YAT システムで使用される問合せ言語であり，当初は論理型のプログラムに基づく異種情報間のデータ変換言語であった．新しい YATL は関数型言語であり，XML の構造変換を取り扱える．match 節で変数に束縛した要素を where 節で条件に基づき選択し，make 節で出力として再構築する．
- XSLT [40)]
  ··· Extensible Stylesheet Language Transformations（XSLT）は W3C XSLT ワーキンググループにより設計された．テンプレートルールの集合から成り立つ．テンプレートルールは
  – パターン ··· XML のソース木のノードに照合する
  – テンプレート ··· 結果木を形作る

  から成る．要素の選択等には XPATH [41)]を使用する．

これらに本論文で提案する拡張 SuperSQL 質問文

表 5 要素の束縛，選択と出力結果の再構築
Table 5 Pattern, filter and constructor.

機能	XML-QL	Quilt	YATL	拡張 SuperSQL 質問文
要素の束縛	WHERE 節	FOR 節，LET 節	match 節	FROM 節および GENERATE 節（TFE）
要素の選択	WHERE 節	WHERE 節	where 節	WHERE 節
出力結果の再構築	CONSTRUCT 節	RETURN 節	make 節	GENERATE 節（TFE）

を合わせた 5 つの構造変換言語に関して比較する．

すべての言語は XML を対象として構造変換を行うが，拡張 SuperSQL 質問文のみ出力が関係データとなる．ただし，XSLT は基本的には XML のソース木から結果木への変換を行い通常の出力は XML であるが，テンプレート中に出力のためのテキストデータを記述することができるため LaTeX 等の構造化文書やテキストデータを出力することもできる．

XSLT は宣言的な記述もできるが，for 文や if 文等が使用でき手続き的な記述が可能となっている．このため，XSLT では結果となる XML を生成するには match 属性，select 属性に書かれたパターンと照合した XML のソース木のノードに対し，再帰的なテンプレートの適用や for 文，if 文等の手続き的な処理により結果木を生成する．

一方，XSLT を除く 4 つは宣言的な問合せ言語である．これらは

(1) 要素の束縛（pattern）
(2) 要素の選択（filter）
(3) 出力結果の再構築（constructor）

を行う．この 3 つの機能と各々の言語の対応を**表 5** に示す．拡張 SuperSQL 質問文は関係データを出力する．このため，出力のデータは平坦なデータの集合となり，複雑な構造を出力する必要はない．そこでこれらの 3 つの機能のうち，要素の束縛と選択に関して中心に論ずる．

要素を束縛する際，XML-QL，Quilt，YATL では要素を束縛する節と結果を再構築する節が異なるため，変数に束縛して再利用する必要がある．一方，拡張 SuperSQL 質問文では同じ場所で束縛と再構築を行うため，基本的には変数に束縛する必要はない☆．しかし，GENERATE 節の TFE の出力となる関係データの複数箇所に同じ要素の値を使用したい場合には GENERATE 節の TFE の中で要素を変数で束縛し，TFE 中に再度その変数を記述する必要がある．また，XSLT においてはテンプレート中のどこからでも XPath による経路式により要素を束縛し出力することができるため，基本的に変数は使用しない．

束縛する要素を指定する際には経路式や，タグや [ ] の入れ子等で表す．Quilt や XSLT，拡張 SuperSQL 質問文では経路式（および変数との組合せ）で要素を指定する．一方，XML-QL ではタグの入れ子により入力となる木を表現し，その中の変数を利用して要素を指定する．また，YATL では [ ] の入れ子や ‘*’（要素の反復を示す）を利用して構造を示し，その記述の中の変数により対応する要素を指定する．

複数の変数により束縛を行った場合，結果として変数の値の組が返される．ここでいくつかの特徴が存在する．たとえば図 19-B のデータの Name 属性と Mail 属性を組として取り出した場合，拡張 SuperSQL 質問文や通常の XML-QL，Quilt では各々の要素の直積の結果，図 19-(2) のようになる．つまり，最初の Name 要素と最初の Mail 要素，2 個目の Name 要素と 2 個目の Mail 要素の間の関係が失われてしまう．これは経路式を主として要素を束縛する際の制約である．

これに対し，YATL で

```
...
match "...xml"
  People [ *(Name [ $n ],
             Mail [ $m ]) ]
```

と記述すると ‘(Name Mail)*’ という正規表現に相当し，Name 要素と Mail 要素の間の関連をそのまま残して図 19-(1) のような結果となる．つまり，最初の Name 要素と最初の Mail 要素，2 個目の Name 要素と 2 個目の Mail 要素の間の関係は保存される．

一方，XML-QL を拡張して順序を考慮したデータモデルを使用した場合，

```
WHERE <List>
        <Name>$n</>
        <Mail>$m</>
      </>
```

☆ ただし，拡張 SuperSQL 質問文では要素を指定する経路式の表記の簡便化や情報単位の指定のために経路式を代入する変数を使用する．

```
CONSTRUCT ...
```

といったXML-QLでは

```
{ M.Akahori, aka@db.ics.keio.ac.jp },
{ M.Akahori, ari@db.ics.keio.ac.jp },
{ T.Arisawa, ari@db.ics.keio.ac.jp }
```

のような組の集合が得られる．つまり，最初のName属性の値である‘M.Akahori’は後に続く2つのMail属性と結合し，2番目のName属性の値である‘T.Arisawa’は後に続くMail属性，つまり‘T.Arisawa’のみしか結合しない．

また，XSLTは手続き的な記述が可能であるため，記述内容次第で結果は異なる．

要素の束縛の結果同じ値の組が存在した場合，XML-QLでは重複を削除して組の集合を返す．YATLは重複を削除しない．これらに対し，拡張SuperSQL質問文ではxmldistinctオプションの引数を‘on’にすれば重複を削除して集合として，‘off’とすれば重複を残して組を返す．

要素の選択に関してはXSLT以外の4つはすべてWHERE節で行う．条件の記述の仕方はほぼ同様である．

9. まとめと今後の課題

9.1 ま と め

本論文では関係データベース/XML間の相互変換，すなわち，

(1) SuperSQLによる関係データベース内のデータに対するXMLビューの生成，
(2) 生成したXMLデータビューに対するXML質問文（XQL）の処理，
(3) 拡張SuperSQLによるXMLデータの関係データベース内の新規もしくは既存のリレーションへの格納，

について論じた．これらにより，SQLやXML質問言語による関係データベース内のデータとXMLデータの統合検索とが可能となった．ただし，関係データとして格納する際の主な対象となるXMLデータは，データ中心的かつその意味が順序に独立であるXMLデータである．

本システムによる関係データベースに対するXMLデータビューの提供により，内部構造やSQLの知識を持たずにXQLによるデータベースへの問合せが可能である．特に，大量のデータから条件を満たす限定された件数の結果を得る場合に有効である．また，SQLで関係データベース内のデータと格納したXMLデータの統合検索が可能となるとともに，これらを連携させデータ交換で受け取ったXMLデータと自前のデータを統合して新たなXMLデータとして第三者に発信することが可能である．

本問合せシステムでは問合せ結果のXMLデータとともに，SuperSQLのTFEの持つ構造情報を用いてDTD，RELAX等の構造情報（およびレイアウト指定を行うXSL）も同時に生成できる．ただし，TFEから直接生成した構造情報は同時に生成したXMLデータ文書以外の構造も認めてしまう緩やかな構造制約であり，データベースから抽出した情報を用いる方法等との併用の必要がある．

9.2 今後の課題

本システムで扱える構造情報として，DTDやRELAX以外にXML Schema等の生成に関して考慮中である．さらに，SuperSQLはGENERATE節での出力媒体の指定を利用してデータベース内のデータを中心としたOne-source Multi-useを実現しているが，これと同様に，XMLデータに付随してLaTeXや他の出力媒体に変換するXSLを生成し，関係データベースから生成したXMLデータを中心としたOne-source Multi-useを実現し，XMLを中心としたデータ利用をさらに促進することを検討する．

今回問合せの際に使用したXML質問言語はフィルタリング機能が中心のXQLであるが，今後再構築の機能を持つXML-QL等の言語に対応する予定である．これによりXML質問文により生成されるXMLデータの仮想ビューが多様になり，その有効性が高まる．一方で，多様なビュー定義が可能となるとユーザの負担が増大することとなり，作業の負担を軽減するためにユーザインタフェースの開発等が必要である．そこで，問合せ文やその結果，構造情報等を対話的に生成できるインタフェースについて検討を進めている．

謝辞 本研究の一部は情報処理振興事業協会（IPA）の高度情報化支援ソフトウェアシーズ育成事業の補助による．

参 考 文 献

1) Bray, T., Paoli, J. and Sperberg-McQueen, C.M.: Extensible markup Language (XML) 1.0 W3C Recommendation (1998). http://www.w3.org/TR/REC-xml.
2) Rosetta Net: http://www.rosettanet.org, http://www.rosettanet.gr.jp.
3) ebXML: http://www.ebxml.org.
4) UDDI: http://www.uddi.org.
5) 赤堀正剛，有澤達也，遠山元道：データベース

の XML ビューに対する XQL 質問の処理，電子情報通信学会技術研究報告，DE2000-92 (2000).
6) 赤堀正剛，有澤達也，遠山元道：SuperSQL を利用した XML データの格納と利用，情報処理学会第 61 回全国大会，5J-03, pp.3-19–3-20 (2000).
7) Toyama, M.: SuperSQL: An Extended SQL for Database Publishing and Presentation, *Proc. ACM SIGMOD International Conference on Management of Data*, pp.584–586 (1998).
8) 有澤達也，実政宏幸，遠山元道：TFE 処理系における問い合わせ文の分割による最適化，情報処理学会第 55 回全国大会，2X-03 (1997).
9) SuperSQL: http://ssql.db.ics.keio.ac.jp.
10) 有澤達也，遠山元道：SuperSQL 処理系におけるグルーピング操作の効率的な実装，電子情報通信学会 DEWS2001 論文，3B-1 (2001).
11) 遠山元道ほか：レイアウト式 TFE の拡張，情報処理学会研究会報告，95-DBS-104, pp.217–224 (1995).
12) 赤堀正剛，有澤達也，遠山元道：SuperSQL による XML データドキュメントの自動生成，情報処理学会研究会報告，2000-DBS-122, pp.455–462 (2000).
13) Robie, J., Lapp, J. and Schach, D.: XML Query Language (XQL), *Proc. Query Languages Workshop*, Cambridge, Mass. (1998). http://www.w3.org/TandS/QL/QL98/pp/xql.html.
14) Robie, J.: The design of XQL. http://www.w3.org/Style/XSL/Group/1998/09/XQL-design.html.
15) Bourret, R.: XML and Databases. http://www.rpbourret.com/xml/XMLAndDatabases.htm.
16) 石川　博：XML とデータベース，情報処理学会誌，Vol.41, No.1, pp.68–73 (2000).
17) 田島敬史：半構造データのためのデータモデルと操作言語，情報処理学会論文誌：データベース，Vol.40, No.SIG 3 (TOD 1), pp.152–170 (1999).
18) Shanmugasundaram, J., Tufte, K., Zhang, C., He, G., DeWitt, D.J. and Naughton, J.F.: Relational Databases for Querying XML Documents: Limitations and Opportunities, *Proc. 25th International Conference on Very Large Data Bases*, pp.302–314 (1999).
19) 吉川正俊，志村壮是，植村俊亮：オブジェクト関係データベースを用いた XML 文書の格納と検索，情報処理学会論文誌：データベース，Vol.40, No.SIG 6 (TOD 3), pp.115–131 (1999).
20) Fernandez, M., Suciu, D. and Tan, W.-C.: SilkRoute: trading between relations and XML, *9th International World Wide Web Conference*, Amsterdam (2000). http://www9.org/w9cdrom/202/202.html.
21) Carey, M.J., Florescu, D., Ives, Z.G., Lu, Y., Shanmugasundaram, J., Shekita, E.J. and Subramanian, S.N.: XPERANTO: Publishing Object-Relational Data as XML, *WebDB* (*Informal Proceedings*) *2000*, pp.105–110 (2000).
22) Carey, M.J., Kiernan, J., Shanmugasundaram, J., Shekita, E.J. and Subramanian, S.N.: XPERANTO: Middleware for Publishing Object-Relational Data as XML Documents, *Proc. 26th International Conference on Very Large Data Bases*, pp.646–648 (2000).
23) Deutsh, A., Fernandez, M., Florescu, D., Levy, A. and Suciu, D.: XML-QL: A Query Language for XML, *Proc. 8th International World Wide Web Conference*, Tronto (1999). http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/.
24) RELAX (Regular Language description for XML): http://www.xml.gr.jp/relax.
25) Garofalakis, M.N., Gionis, A., Rastogi, R., Seshadri, S. and Shim, K.: XTRACT: A System for Extracting Document Type Descriptors from XML Documents, *Proc. 2000 ACM SIGMOD International Conference on Management of Data*, Vol.29, No.2, pp.165–176 (2000).
26) Papakonstantinou, Y. and Velikhov, P.: Enhancing Semistructured Data Mediators with Document Type Definitions, *Proc. 15th International Conference on Data Engineering*, pp.136–145 (1999).
27) 西岡秀一，鬼塚　真：XML のオブジェクトリレーショナルマッピングに関する一手法，情報処理学会研究会報告，2000-DBS-122, pp.17–24 (2000).
28) 春日史朗，林　孝志，飯塚裕一，鈴木源吾：異種情報源環境における情報検索・統合検索方式，電子情報通信学会 DEWS2000 論文，7A-5 (2000).
29) Manolescu, I., Florescu, D., Kossmann, D., Xhumari, F. and Olteanu, D.: Agora: Living with XML and Relational, *Proc. 26th International Conference on Very Large Data Bases*, pp.623–626 (2000).
30) Haas, L.M., Miller, R.J., Niswonger, B., Roth, M.T., Schwarz, P.M. and Wimmers, E.L.: Transforming Heterogeneous Data with Database Middleware: Beyond Integration, *IEEE Data Engineering Bulletin*, Vol.22, No.1, pp.31–36 (1999).
31) Chamberlin, D.D., Robie, J. and Florescu, D.: Quilt: An XML Query Language for Heterogeneous Data Sources, *WebDB* (*Informal Proceedings*) *2000*, pp.53–62 (2000).
32) Seto, T., Nagafuji, T. and Toyama, M.: Gener-

ating HTML sources with TFE enhanced SQL, *SAC 1997*, pp.96–100 (1997).
33) ACM SIGMOD Record XML Version *v*.1.0: http://www.acm.org/sigs/sigmod/record/xml/.
34) GMD-IPSI XQL Engine: http://xml.darmstadt.gmd.de/xql/.
35) Shanmugasundaram, J., Shekita, E.J., Barr, R., Carey, M.J., Lindsay, B.G., Pirahesh, H. and Reinwald, B.: Efficiently Publishing Relational Data as XML Documents, *Proc. 26th International Conference on Very Large Data Bases*, pp.65–76 (2000).
36) Bonifati, A. and Lee, D.: Technical Survey of XML Schema and Query Languages (2001). Submitted for journal publication, http://www.cobase.cs.ucla.edu/tech-docs/dongwon/vldbjCom.pdf.
37) Fernandez, M., Siméon, J. and P.W. (Eds.): XML Query Languages: Experiences and Exemplars (1999). http://www-db.research.bell-labs.com/user/simeon/xquery.html.
38) Cluet, S., Delobel, C., Siméon, J. and Smaga, K.: Your Mediators Need Data Conversion!, *Proc. ACM SIGMOD International Conference on Management of Data*, pp.177–188 (1998).
39) Cluet, S., Delobel, C. and Siméon, J.: YATL: a Functional and Declarative Language for XML. draft manuscript, http://www-db.research.bell-labs.com/user/simeon/icfp.ps.
40) Clark, J. (Ed.): XML Transformations (XSLT) Version 1.0 (1999). http://www.w3.org/TR/xslt.
41) Clark, J. and DeRose, S. (Eds.): XML Path Language(XPath) Version 1.0 (1999). http://www.w3.org/TR/xpath.

(平成 12 年 12 月 20 日受付)
(平成 13 年 3 月 28 日採録)

（担当編集委員　横田 一正）

赤堀　正剛（学生会員）

1999 年慶應義塾大学理工学部管理工学科卒業．現在，同大学院理工学研究科管理工学専攻修士課程在学中．データベースと構造化文書，XML に関する研究に従事．情報処理学会第 61 回全国大会優秀賞受賞．

有澤　達也（学生会員）

1997 年慶應義塾大学理工学部管理工学科卒業．1999 年同大学院理工学研究科管理工学専攻修士課程修了．現在，同大学院理工学研究科計算機科学専攻博士課程在学中．データベースと構造化文書，データベース質問処理に関する研究に従事．ACM 会員．

遠山　元道（正会員）

1979 年慶應義塾大学工学部管理工学科卒業．1981 年同大学院修士課程修了．1984 年同学科助手．1992 年専任講師．1996 年同学部情報工学科に移籍．現在に至る．博士（工学）．1996 年オレゴン大学院大学客員研究員．1998 年から 2001 年まで科学技術振興事業団さきがけ研究 21「情報と知」領域研究員．主にデータベースの研究に従事．電子情報通信学会，日本ソフトウェア科学会，IEEE Computer Society，ACM 等会員．