関係データベース上の階層関係を持つ妥当なXMLビューの設計法

武川 肇[†] 太田 学[††]
片山 薫[††] 石川 博[††]

関係データベース（RDB）上のXMLビューは，問合せに従いRDB中のデータを変換し，XML文書を出力するための仕組みである．RDB上のXMLビューがデータの無駄な重複を含むとXML文書に冗長性が発生する．冗長性は更新時異状の原因となるため，RDB上のXMLビューがデータの無駄な重複を含んでいないか確認する必要がある．一般にこの確認のためには，属性間における従属性の把握が必要となる．しかし正規化されていない関係データにおいて，この従属性の把握は機械的にはできない．そこで本論文は正規化されたRDBスキーマにおける制約を利用して，冗長性がなく，更新時異状を発生しないXMLビューの設計手法を提案した．さらに階層関係を用いて表現したビューの方が，階層関係を用いないで表現したビューより，XML文書のサイズが減少することを実験的に示した．その結果，本設計法が関係データの情報量を変更することなく，通信コストを低減させることが明らかになった．

Design Method for Valid XML Views with Hierarchical Relations on a Relational Database

Hajime Takekawa,[†] Manabu Ohta,[††] Kaoru Katayama[††]
and Hiroshi Ishikawa[††]

An XML view on a relational database (RDB) is a mechanism converting data in a RDB to XML documents according to a query. If an XML view on a RDB contains useless repetitions of data, it will induce redundancies in the XML documents. It is necessary to ascertain whether such useless repetitions are contained in the XML view, because redundancies lead to potential update anomalies. In order to realize the ascertainment, it is necessary to grasp dependencies between attributes. However, it is impossible to grasp dependencies automatically from non-normalized relational data. In this paper, we propose a method to design views which have neither redundancies nor update anomalies by using primary key and foreign key constraints from a normalized RDB schema. It was confirmed theoretically and experimentally, the size of the XML document with a hierarchical relation is smaller than that of the XML document without hierarchies. It became clear that it reduces a communication cost without reducing quantity of information in relational data by using this design method.

1. 研究背景

現在インターネット上のデータ交換の標準形式としてXML（Extensible Markup Language）が利用されている．一方関係データベース（Relational Database; RDB）はデータのリポジトリとして広く利用されている．そのためにRDBからXML文書としての変換はデータの再利用という重要な意味を持ち，XMLをRDBのXMLインタフェースとして利用できるような技術が研究されている[5]．

たとえば，RDB上のXMLビューは，問合せに従いRDB中のデータを変換し，XML文書を出力するための仕組みである．その1つであるSilkroute[6]はRDBと連携し，データを仮想のXML文書（これをXMLビューという）として提供するミドルウェアである．Web/Internet上のアプリケーションはXMLビューを通して間接的にRDBに接続する．そしてアプリケーションがXML文書を取得するために問合せを行うと，XMLビューは動的に処理し，関係データ（relational data）をXML文書に変換し，アプリケーションへ返す．

川田ら[18]は，問合せにRDBで処理できない外部

† 職業能力開発総合大学校情報工学科
Department of Information and Computer Science, Polytechnic University
†† 東京都立大学大学院工学研究科
Graduate School of Engineering, Tokyo Metropolitan University

```
<Order ID="1" date="9/30" ItemID="A001" qty="10" />
<Order ID="1" date="9/30" ItemID="A002" qty="20" />
```
(a) 階層が無い表現(冗長)

```
<Order ID="1" date="9/30">
    <OrderItem ItemID="A001" qty="10" />
    <OrderItem ItemID="A002" qty="20" />
</Order>
```
(b) 階層をもつ表現

図 1 XML 文書に含まれる文字列の無駄な重複
Fig. 1 Useless repetitions between strings in an XML document.

```
<Order ID="1" date="9/30">
    <OrderItem ItemID="A001" qty="10">
        <Item Name="消しゴム" />
    </OrderItem>
    <OrderItem ItemID="A002" qty="20">
        <Item Name="鉛筆" />
    </OrderItem>
</Order>
<Order ID="2" date="11/26" >
    <OrderItem ItemID="A001" qty="20">
        <Item Name="消しゴム" />
    </OrderItem>
</Order>
```
図 2 無駄な重複を含む階層を持つ表現
Fig. 2 Hierarchical expression with a useless repetition.

関数が含まれる場合の XML ビューの最適化について提案している．しかしこの最適化は高速化を目的としており，生成される XML 文書に含まれる文字列などのデータの無駄な重複（以降，無駄な重複）の冗長性を防ぐことは考慮されていない．もし XML 文書に冗長性が含まれると，ネットワーク上の通信コストの増大，更新時異状（update anomaly）などの問題点が発生する．

一般に RDB において，更新時異状を未然に防ぐ必要がある．そのために正規形に基づいた冗長性の除去，つまり正規化が行われている．一方 XML において，XNF（XML Normal Form）[7] が XML の正規形として提案されている．この XNF は BCNF（Boyce-Codd Normal Form）と入れ子関係（nested relation）を利用している．

ここで XML 文書に含まれる文字列の無駄な重複の例を**図 1** に示す．図 1 (a) と図 1 (b) は同じ関係データを表現している．図 1 (a) と図 1 (b) を比較すると，図 1 (a) は「ID=1 date=9/30」の無駄な重複を含んでおり，図 1 (b) にはそのような無駄はない．よって図 1 (b) のような階層を持つ表現の方が冗長性を省けるのではないかと期待できる．

しかし安易に階層を持つ表現を用いると，逆に無駄な重複を含むことがある．その例を**図 2** に示す．図 2 のデータは階層を利用しているが，同じデータが現れている．これは無駄な重複と考えられる．単に階層を持つ表現に変換することは，その意図とは逆に冗長性があるデータを生ずることになる．

以上のように一般には階層を利用して冗長性を除去できるが，逆に階層を利用することで冗長性を発生させることがある．冗長性は更新時異状の原因となるため，RDB 上の XML ビューが無駄な重複を含んでいないか確認する必要がある．一般にこの確認は正規形を利用するため，属性間における従属性の把握が必要となる．しかし最初に示した図 1 (a) のような正規化されていない XML 文書からは，従属性の把握が機械的にはできない．たとえば冗長性の除去に用いる XNF [7] の正規化手続き（normalization procedure）では入力として従属性を与える必要がある．

そこで本論文は，正規化された RDB スキーマにおける主キーと外部キーの制約を利用して，冗長性がなく，更新時異状を発生しない XML ビューの設計手法を提案した．この設計法は NF–N3 [10] の条件を満たすことで冗長性がないことを保障している．

本論文の構成は以下のとおりである．続く 2 章では RDB 上の XML ビューの問題点を明らかにする．3 章では本設計法の方針を述べる．4 章では本提案の設計法とそれを支援するアルゴリズムについて述べる．5 章では本設計法を RDB 上の XML ビューへ適用した評価実験を示す．6 章では本設計法と関連研究との比較をする．7 章ではまとめと今後の課題について述べる．

2. RDB 上の XML ビューの問題点

この章では RDB 上の XML ビューの問題点を明らかにするため，RDB 上の XML ビューの冗長性とその冗長性がもたらす悪影響について述べる．

2.1 冗 長 性

本論文でいう冗長性とは，RDB 上の XML ビュー経由の XML 文書に含まれる無駄な重複のことである．

一般に RDB をデータソースとする RDB 上の XML ビューの仕組みは，コンパイル処理，データ構造変換処理，問合せ処理の 3 つのモジュールからなる（**図 3**）．コンパイル処理モジュールは，ビュー定義言語によって記述された XML ビュー作成指示に従い，それをコンパイルし，XML ビューオブジェクトを作成する．その作成指示には，配置情報と XML 問合せの 2 つについての記述が含まれている．

残りの 2 つのモジュールは問合せ実行指示を合図に処理を開始する．データ構造変換処理モジュールは，問合せ処理モジュールに必要な XML データソースを供給するために，配置情報に従い，RDB データ

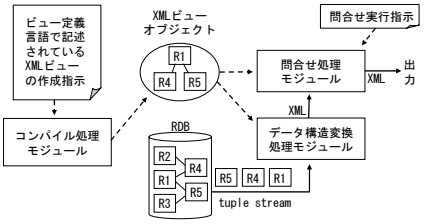

図 3 RDB 上の XML ビューの仕組み
Fig. 3 A mechanism of XML views on a relational database.

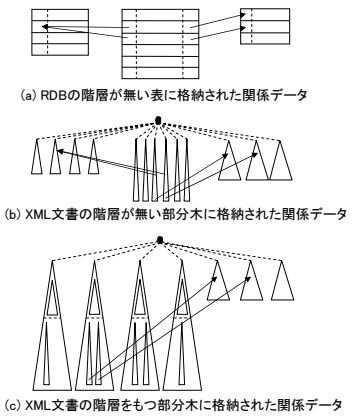

図 4 RDB と XML 文書の関係データの表現
Fig. 4 Expression of a relational data for RDB and an XML document.

ソースから関係データを取り出し，その関係データを XML の構造へ変換する．一方問合せ処理モジュールは，XML 問合せに従い，データ構造変換処理モジュールから得た XML データソースに集計や結合などの階層構造の変換をともなう問合せ処理を行い，その結果を出力する．

以上のように RDB 上の XML ビュー経由の問合せにおいて，データ構造と階層構造の 2 段階の構造変換が行われる．このとき配置情報が RDB データソースの適切なデータ表現をとっていないと，配置情報に従うデータ構造変換によって，出力される XML 文書に無駄な重複が発生する．本論文は RDB 上の XML ビューの冗長性のないデータ構造変換に関する研究である．

2.2 正規化された関係データの XML 文書への変換とその冗長性

この節では XML 文書における関係データの 2 つの表現と，RDB データソースから XML 文書へのデータ構造変換における冗長性について述べる．

まず関係データの表現について述べる（**図 4**）．関係データは一般に RDB に格納されている．RDB における関係データはあらかじめ正規化された関係の集合であり，それぞれの正規化された関係は RDB の表へ格納される．RDB において，それぞれの表は対等であり，階層を持たない．関係はタプルの集合で，そのタプルの属性はすべて構造を持たない．表に格納されたタプル間の結びつきは，外部キーから主キーへの参照によって表現される（図 4 (a)）．

この RDB に格納されている関係データから XML 文書への変換において，RDB の表に格納されているタプルは XML 文書の部分木へ配置される．なお XML 文書のルートは要素が 1 つしか許されていないが，XML 文書のルート以外はその制限がないため，配置が複数可能である．配置される部分木どうしは対等である．そしてキー参照によって対等な部分木に格納されたタプル間の結びつきも表現できる（図 4 (b)）．さらに RDB とは異なる表現方法がある．これは部分木に階層を持たせ，木構造としてタプルの親子関係を表現する格納法である．異なる表に格納されたタプルの組合せも，部分木への配置が可能である（図 4 (c)）．ただしある 1 つのタプルが分割され，それらが階層間にまたがって配置されることはない．以降，親子関係による関係データの木構造の表現を「階層関係」，キー参照による対等な部分木間に格納したタプル間の結びつきの表現を「参照関係」と呼ぶ．階層関係はその構造上，子から親は唯一であり，親から見た子は複数存在しうる．

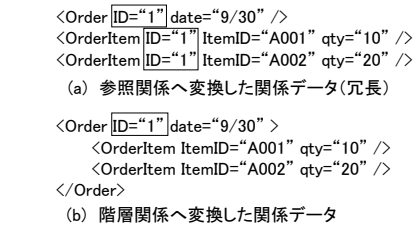

図 5 データ構造変換時の冗長性の例
Fig. 5 An example of redundancy on converting data structure.

次に RDB データソースから XML 文書へのデータ構造変換で発生する冗長性について述べる．例として同じ関係データから異なる表現を用いて変換した例を**図 5** に示す．図 5 (a) は参照関係，図 5 (b) は階層関係へ変換した関係データである．図 5 (a) は図 5 (b) に比べて，ID 属性が多く現れ，その属性は，RDB 中では OrderItem 表の外部キーである．よって図 5 (a) は外部キーの無駄な重複を含んでいるといえる．図 5 の例では参照関係が無駄な重複を含むことであるが，以前に示した図 2 のように階層関係も無駄な重複を含む

ことがあるので注意が必要である．

データの無駄な重複を次の 2 つに分類する．

- ある階層関係の子から見た親のタプルの対応関係（カーディナリティ）が 1 対 N であるとき，その階層関係の子に同じデータの無駄な重複が必ず起こる．
- ある参照関係を同じデータの無駄な重複がない階層関係で表現できるとき，その参照関係には「外部キーの無駄な重複」が存在する．

我々は RDB から XML 文書へのデータ構造変換において，主キーと外部キーの組合せによる関連は階層関係と参照関係，またはそのどちらかへの変換が可能である．そのため本設計法は RDB 表の XML 構造における配置情報とタプル間の関連の情報を一緒に表現可能な xE/R 図という変換ダイアグラムを利用する．

2.3 冗長性がもたらす悪影響

冗長性はネットワーク上の通信コストだけでなくアプリケーションにも悪影響を与える．

図 5 (a) において外部キーの無駄な重複を除去すべき理由と，図 2 の同じ情報の無駄な重複を除去すべき理由を 3 つに整理して以下に示す．

・参照関係の XML 文書中の位置

XML 文書用のフィルタ型のアプリケーションを考える．このアプリケーションは文書順の逐次処理と文書全体をメモリ上に展開しない方式によって，巨大な XML 文書に対する高速な処理を実現する．しかし文書順に格納されていないデータが存在すると，そのつどアプリケーションは XML 文書を再読み込みするなどの対策が必要となり，負荷が大きくなる．XML 文書中の参照関係は参照元と参照先に分離し，それらの位置は文書順とはいえないため，参照先の検索が必要となる．一方，階層関係で表現されたデータは親子の要素に分かれているが，それらは文書順であるため 1 回の読み込みで済み，検索は不要である．したがって階層関係の方がフィルタ型のアプリケーション上での逐次処理に適しているといえる．

・アプリケーションにおける更新時異状

サーバとの問合せの回数の削減のため，一般に受け取った XML 文書をアプリケーション側でキャッシュする．アプリケーションは XML ビューへの追加，削除，修正の問合せを行った後，更新時異状を避けるために，XML ビューから XML 文書を再取得する必要がある．またはアプリケーション側で取得した XML 文書をあらかじめ正規化し更新時異状を避ける．この正規化は，問合せの種類の数が多い場合を想定すると，サーバ側で更新時異状を発生しない XML 文書の生成処理の自動化のために有用であると考えられる．

・XML ビューにおける更新時異状

問合せ処理の応答時間を速くするため，一般に XML データソースをキャッシュする．更新時異状が発生しない XML ビューの場合は，RDB データソースと XML データソースの変更を同時に，または XML データソースの変更を先に実行できる．一方，更新時異状が発生する場合は，XML ビューの更新時異状を避けるため，RDB を変更した後に，XML データソースを再作成する必要がある．以上のことから，更新時異状を発生しない XML データソースは問合せ応答時間を速くできると期待できる．

以上のことから本設計法は，更新時異状と通信コストの 2 つを考慮し，冗長性のない XML ビュー生成の自動化を試みる．その自動化は，RDB の表が第 3 正規形（ただし関係がキーだけの関係の場合は第 4 正規形）に分解されていることを前提にしている．

なお，RDB 上にどのような XML ビューを構築したいかは，本来ユーザの要求で決まるものであるため，冗長性の削減は，必ずしもユーザの要求と一致するとは限らない．そこで本設計法は，RDB から利用する表と階層関係のルートの表はユーザが任意に決定できることにより，XML ビューの自由度を確保できると考えた．導出された XML ビューにおける問合せ処理は今後の課題である．

正規化手続きは一般に関数従属性と多値従属性[9] を利用して行う．しかし本設計法は，主キーと外部キーを利用して正規化を行う．そこでまず 3 章で，主キーと外部キーを利用して RDB の表を XML ビューのどこに配置すべきかについて議論し，そしてその結果を変換規則としてまとめる．その後 4 章で，その変換規則を実現するための本設計法を示し，5 章で，本設計法による XML 文書の削減量について実験結果から考察する．

3. XML ビューの設計方針

この章では，本提案の XML ビューの設計方針を明らかにするため，本設計法の冗長性をなくす基準，本設計中に得られる変換ダイアグラム，その変換ダイアグラムからの自明な従属性の除去，本設計法の変換規則の 4 点について述べる．

3.1 XML ビューの設計基準

この節では本設計法の冗長性をなくす基準について

述べる．

本設計法は，外部キーの無駄な重複を除去するために階層関係を導出し，かつ階層関係と参照関係の集合が Ling の NF–N3[10] の条件を満たすことで冗長性がないことを保障する．NF–N3 とは階層を持つ関係に利用できる正規形であり，NF–N3 の条件を満たす階層を持つ関係には冗長性がない．よって RDB から XML 文書への配置情報としてあらかじめ NF–N3 の条件を満たす XML ビューが定義できれば，その XML ビューを雛型として作成する XML 文書に含まれる関係データにも冗長性がないといえる．

しかし正規形は同じデータの重複をなくすことを目的としているため，正規形だけを利用する正規化手続きでは，前述した 2 種類の無駄な重複を防ぐことはできない．たとえば図 5 (a) のような参照関係だけの場合，同じデータの無駄な重複は存在せず，NF–N3 の条件も満たす．つまり同じデータの無駄な重複を防ぐだけでは，外部キーの無駄な重複を含む可能性がある．ゆえに外部キーの無駄な重複を防ぐためには階層関係も利用する必要がある．本設計法は，2 種類の無駄な重複を防ぐ手法である．

定義 1 ビューが妥当であるとは，冗長性がないことである．その状態のビューを**妥当なビュー**という．妥当なビューとなるには，以下の 2 つの条件を満たす必要がある．

(1) 外部キーの無駄な重複を含まないこと．

(2) 同じデータの無駄な重複を含まないこと． □

なお，本設計法では同じデータの無駄な重複を防ぐために，階層関係と参照関係の集合が NF–N3 の条件を満たすこととする．

NF–N3 は冗長性をなくす以外に，NF–N3 を満たす階層関係を拡張関数従属性[10] で表記できるという特徴を持つ．Ling の拡張関数従属性とは，関数従属性を拡張し多値従属性も表現できるようにしたものである．

A，B が属性の集合で，B は A に拡張関数従属であるとき，次のように表記する．

$$A \Rightarrow B.$$

左辺が決定項（キー）であり，右辺が従属項である．

たとえば，**図 6** に示される階層を持つ関係を拡張関数従属性で表記すると以下のようになる．

顧客 ID ⇒ 顧客名, 受注 (受注 ID, 受注日,
受注明細 (商品 ID, 数量)), 住所, 電話.

上記の ( ) を，繰返しグループ（repeating group）と呼び，開始ブラケットの前に関係名を持つ．この名はオプションである．繰返しグループは，繰返しグループ以外に埋め込み関係（embedded relation），または入れ子関係を許している．また顧客 ID をこの関係の“キー”，受注 ID を“繰返しグループのキー”と呼ぶ．なおキーおよび繰返しグループのキーに，繰返しグループは許されない．

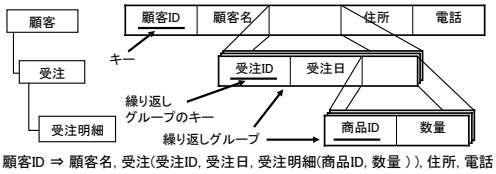

図 6 階層を持つ関係を拡張関数従属性で表記した例
Fig. 6 An example described by extended functional dependency.

次に拡張関数従属性で表現できる関係が NF–N3 であるための条件をあげる[10]．

(1) 関係 R は少なくとも 1 個のキー K を持つこと．

(2) ルートの属性の関係は第 3 正規形（3NF）であり，ルートの属性すべてがキーだけの関係である場合は第 4 正規形（4NF）であること．

(3) 繰返しグループのキー K_G を持つ R の繰返しグループ G ごとに，G のすべての属性と K が NF–N3 である関係を構成できること．このとき $\{K_G\}$ または $\{K, K_G\}$ が関係のキーであること． □

上記の条件は再帰している．つまり (1) と (2) はルートの属性だけを対象としていない．(3) より，$\{K, G\}$ の関係は NF–N3 でなければならない．そのため $\{K, G\}$ の関係も再び (1)，(2)，(3) を満たす必要がある．また (3) は繰返しグループのキーがない状態を許さない．したがって本設計法においても，繰返しグループのキーがない階層関係は妥当なビューとしない．

【NF–N3 の条件 (3) の具体例】

図 6 の階層関係の例を使って NF–N3 の条件 (3) の具体例を示す．

キーとルートの繰返しグループの属性集合の関係 { 顧客 ID, 受注 ID, 受注日, 受注明細 (商品 ID, 数量)} は，{ 受注 ID} をキーとして，次のように拡張関数従属性で表記できる．

受注 ID ⇒ 顧客 ID, 受注日,
受注明細 (商品 ID, 数量).

同様に，キーと繰返しグループの属性集合の関係 { 受注 ID, 商品 ID, 数量 } は，{ 受注 ID, 商品 ID} をキーとして，次のように拡張関数従属性で表記できる．

受注 ID, 商品 ID ⇒ 数量.

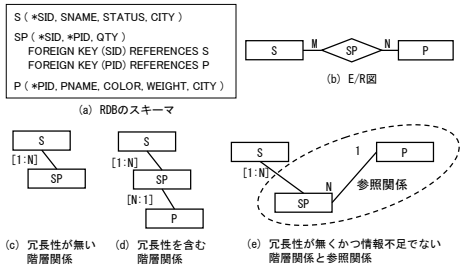

図 7 階層関係と参照関係への変換例 1
Fig. 7 First example of converting from relationship to hierarchical relations and reference relations.

3.2 XML 文書の階層関係と参照関係への変換ダイアグラム

この節では本設計中に得られる XML 文書の階層関係と参照関係への変換ダイアグラムについて説明する．

図 7 (a) に，3 つの表 S，SP，P のスキーマを示す．これは，Date[1] の納入業者・部品データベースを参考にしている．なお主キー制約には列名の前に * を記述し，先頭に FOREIGN KEY のキーワードが記された行は外部キー制約を示している．たとえば SP では SID と PID が主キー制約で，その SID は S を参照する外部キー制約でもある．

図 7 (b) は図 7 (a) を実体/関連図[8]（Entity Relationship diagram; E/R 図）で表している．E/R 図は，概念設計で利用されるデータモデルである．実体型は矩形で表し，これは実社会の個々の実体を意味する．関連型は実体型と実体型の間に菱形で表し，これは実体間の関連を意味する．実体型と関連型を結ぶ辺には，関連における実体の次数を記述する．ただし本設計法は 3 項以上の関連を扱わない．図 7 (b) では S と P の間に多対多の関連があることを表している．図 7 (a) の SP のように RDB スキーマでは関連は主キー制約と外部キー制約の組合せになる．

図 7 (c)–(e) は XML 文書の階層関係と参照関係への変換ダイアグラムである．これらの図は階層関係と参照関係を表現できるように拡張した E/R 図（以降，このダイアグラムを **xE/R 図**という）で示している．矩形は RDB の表から XML 文書の要素への配置を意味する．xE/R 図は関連型を利用しない．すべて矩形で表し，要素として扱う．要素間の辺にはカーディナリティを表記する．階層関係のカーディナリティは [*parent* : *child*] で表記する．これによって関連から階層関係への構造変換をより明確にできる．

図 7 (c)，(d) は S を階層関係のルートとした xE/R 図である．カーディナリティに注目すると，図 7 (c) の階層関係は [1:N] の関係であるが，図 7 (d) の階層関係は [N:1] の関係を含んでいる．[N:1] は，子から見た親のタプルのカーディナリティが 1 対 N であるため同じデータの無駄な重複が発生し，冗長性を必ず含む．このことは，階層関係では子から見た親を複数決定することを表現できないためである．そのために [N:1] の関係は同じデータの無駄な重複の冗長性の原因となる．たとえば，図 2 に示した階層関係には [N:1] の関係が含まれている．よって図 7 (d) は妥当なビューの条件を満たさない．

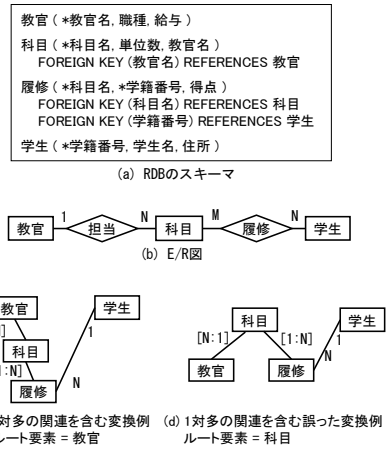

図 8 階層関係と参照関係への変換例 2
Fig. 8 Second example of converting from relationship to hierarchical relations and reference relations.

図 7 (c) は冗長性を含まないが，XML ビューに問い合わせるアプリケーション側で情報が不足する場合が考えられる．たとえば SP から PID は取得できるが，PNAME は取得できない．本設計法は図 7 (e) に示すような，階層関係に冗長性がなく，かつ情報不足とならないビューを導出する．図 7 (e) は関連が 2 つの関係（つまり階層関係と参照関係）に置き換わる例でもある．ここでは P は SP との参照関係となっている．以降，P のように参照される要素を「被参照要素」，SP のように参照する要素を「参照要素」という．

図 8 に別の設計例を示す．図 8 (a) の RDB のスキーマからビューの設計を試みる．図 8 (b) は，増永[19] に記載されている E/R 図を参考にしている．ただし本設計法では 1 対多の関連には属性を持たないこととする．

図 8 (b) は図 8 (a) の E/R 図であり，担当と履修が関連となっている．教官と科目は 1 対多の関連があり，科目と学生は多対多の関連がある．一般に図 7 (a) の SP のように多対多の関連は関連関係（relationship relation）として RDB のスキーマが存在する．一方 1 対多の関連にはスキーマが存在せず，関連における

次数が 1 である実体（以降，1 側の実体という）のスキーマに主キー制約を持ち，そして関連における次数が 1 以外である実体（以降，多側の実体という）のスキーマに外部キー制約を持つ．たとえば図 8 (b) の担当は 1 対多の関連であり，1 側の実体は教官そして多側の実体は科目である．よって図 8 (a) には担当のスキーマは存在せず，教官は主キー制約，科目は外部キー制約を持つ．

図 8 (c) は階層関係のルートを教官とした場合の本設計中に得られた xE/R 図である．科目は教官の子となり，さらに履修が科目の子となる．学生は参照関係となる．本設計において，多対多の場合と異なり，1 対多の関連の変換はさらに別の関連を子に配置できる．

しかし 1 対多の関連の変換は 1 側の実体の子に多側の実体を配置できない．図 8 (d) は誤った変換を示している．その理由は多側の科目の子に教官を配置しているためである．この関係は [N:1] であるから，図 7 (d) と同様，図 8 (d) はデータの無駄な重複を含むため，これは妥当なビューの条件を満たさない．

3.3 xE/R 図からの属性の無駄な重複の削除

前節で説明した xE/R 図において，RDB 中の主キーと外部キーの属性が 1 つの階層関係に配置されている．ところがそれらは階層間で重複する値をとるため，冗長性の原因となる．したがって主キーまたは外部キーの属性のどちらかを削除する必要がある．このデータの無駄な重複は，本来なら参照関係に存在した外部キーの無駄な重複である．以降，xE/R 図において，階層間での主キーまたは外部キーのデータの無駄な重複を「属性の無駄な重複」と呼ぶ．この節では xE/R 図からの属性の無駄な重複の削除について述べる．

まず繰返しグループから属性を削除するための定理を示す．

定理 1 キー A を持つ $\{A, B, (C, (D))\}$ の関係 R において，A，B，C はそれぞれ単一の値を持つ属性の集合であり，D は繰返しグループを含んでもよいとする．そして A はその関係の拡張関数従属性のキーであり，かつ $A \supseteq C$ のとき，C は削除可能である．

（証明） Ling の基礎従属性[10] の定義を用い，R のルートの属性を取り除いた関係 R_G は，$\{A, C, (D)\}$ である．さらに基礎従属性を用い，R_G におけるルートの属性だけの属性集合は，$\{A, C\}$ である．ここで $A \supseteq C$ から $A \rightarrow C$ は自明な関数従属性である．ゆえに C は削除可能である． □

定理 1 は $A \supseteq B$ なら $A \Rightarrow (B)$ を意味し，以降，これを「自明な拡張関数従属性」と呼ぶ．

なお，定理 1 では階層を持つ関係の従属性を見つけるために，基礎従属性を用いている．以下に基礎従属性を記す．

キー K を持つ関係 R において，R の基礎従属性のセット（以降 BD(R)）は次に定義する：

(1) R の単一の値を持つ属性だけを含んだすべての関数従属は，BD(R) にある．
(2) R の繰返しグループ G ごとに，拡張関数従属性 $K \Rightarrow G$ は BD(R) にある．
(3) R の繰返しグループ G ごとに，K と G からなる関係は BD(R) のサブセットである． □

たとえば，定理 1 の関係 R の基礎従属性のセットは次のようになる．

(1) より，$A \rightarrow B$.
(2) より，$A \Rightarrow (C, (D))$.
(3) より，$\{A, C, (D)\}$ の基礎従属性のセット．

次に，自明な拡張関数従属性に基づいた xE/R 図からの属性の無駄な重複の削除手順を以下に示す．

(1) xE/R 図の階層関係を拡張関数従属性で表現する．ただし，拡張関数従属性のキーは xE/R 図のルート要素へ配置した表の主キーとすること．
(2) 繰返しグループを含む場合，繰返しグループごとに，キーとの自明な拡張関数従属性となる属性を削除する．ただし，削除後も繰返しグループのキーが存在していること．
(3) 繰返しグループに繰返しグループを含む場合，繰返しグループのキー K_G を持つ関係 R の繰返しグループ G ごとに，G のすべての属性と R のキー K から構成される関係を拡張関数従属性で表記し，(2) 以降を繰り返す．このとき $\{K_G\}$ または $\{K, K_G\}$ が関係のキーであること．

上記の手順に従い，以下に具体例を示す．

【多対多の関連】

図 7 (c) を使い，多対多の関連から変換した階層関係における属性の無駄な重複の削除手順を説明する．まず，図 7 (c) を拡張関数従属性で表記する．

SID ⇒ SNAME,STATUS,CITY,
SP(SID,PID,QTY).

次に，属性の無駄な重複を削除する．その方法は，従属項の繰返しグループの属性の中で，決定項の属性に含まれる属性がある場合，従属項からその属性を削除する．結果を次に示す．

SID ⇒ SNAME,STATUS,CITY,SP(PID, QTY).

上記の結果は親を参照する子の外部キーの属性を削除することになり，外部キーの無駄な重複を防ぐ．

【1 対多の関連 [1:N]】

図 8 (c) を使い，1 対多の関連から [1:N] へ変換した

関係を含む階層関係における属性の無駄な重複の削除手順を説明する．まず図 8 (c) の階層関係を拡張関数従属性で表記する．

教官名 ⇒ 職種, 給与,
科目 (科目名, 単位数, 教官名,
履修 (科目名, 学籍番号, 得点)).

次に属性の無駄な重複を削除する．

教官名 ⇒ 職種, 給与,
科目 (科目名, 単位数,
履修 (科目名, 学籍番号, 得点)).

さらにキーと繰返しグループの関係を拡張関数従属性で表記し直す．この場合 $\{K_G\}$ が決定項となる．

科目名 ⇒ 単位数, 教官名
履修 (科目名, 学籍番号, 得点).

最後に属性の無駄な重複を削除する．

科目名 ⇒ 単位数, 教官名,
履修 (学籍番号, 得点).

上記の結果は親を参照する子の外部キーの属性を削除することになり，外部キーの無駄な重複を防ぐ．

3.4 関連から階層関係と参照関係への変換規則

この節では本設計法における関連から階層関係への変換規則とその例外について述べる．

本設計法の変換規則を以下に示す．

・階層関係を参照関係よりも優先して決定する．
・各表が配置できる要素は 1 カ所である．
・関連から階層関係への変換においては，親と子が [1:N] の関係となり，子は親の主キーを参照する外部キーを持つ．

以降に，1 対多と多対多の関連からの変換規則の例外について述べる．

【1 対 1 の関連】

1 対 1 の関連から変換した階層関係を拡張関数従属性で表記し，自明な拡張関数従属性を削除すると，繰返しグループにキーがなくなる．したがって NF–N3 の条件 (3) を満たせないと判断し，本設計法においても，1 対 1 の変換は階層関係としない．

【弱実体】

E/R 図には弱実体型があり，二重の矩形で弱実体を表す．実体と弱実体の間の関連を ID 関連型と呼び，二重の菱形で表す（**図 9** (a)）．弱実体は関連する実体の主キーを外部キーとして持ち，その外部キーは主キーの一部となる．そして ID 関連には RDB のスキーマが存在しない（図 9 (b)）．RDB のスキーマに関しては，ID 関連との弱実体は 1 対多の関連の実体と差はないため，ID 関連に関する変換規則に例外は特に設けないこととする．本設計法においては，弱実体を関連する実体の子要素として階層関係に変換する（図 9 (c)）．

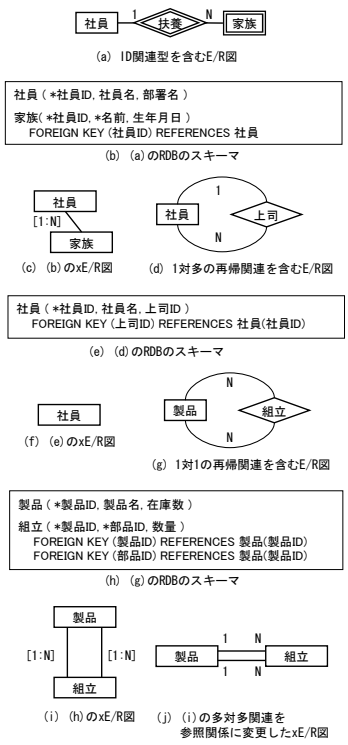

図 9 ID 関連と再帰関連

Fig. 9 ID relationship type and recursive relationship type.

【関連のリング】

表 M が参照する表 P，Q があり，これら 3 つの表を階層関係として変換する場合を考える．P，Q は階層関係では M の親要素となる．階層関係はルートが 1 つであるため，P，Q は必ず共通の親（または祖先）を持つ．その結果，M は 2 つの部分木を結合する．以降，2 つの部分木が同じ要素を子に持つ状態を「関連のリング」と呼ぶ．2 つ以上の表を参照するある表とそれに参照される 2 つ以上の表が 1 つの階層関係に配置できるとき，階層関係中に関連のリングが構成される．

しかし，XML 文書の構造は関連のリングを階層関係で表現できない．石川[17] では，XML 文書の論理的モデルは木構造，複数文書は林構造と位置づけている．特に，XML 文書に含まれる関係データに注目した場合は，図 2 のように XML 文書の部分木に分割して格納する．

XML 文書は部分木間の結合を表現できないため，本設計法では，結合する要素を参照関係として扱う．

【再帰関連】

E/R 図における 1 項関連である再帰関連は，実体

を 1 つだけ扱う．また再帰関連には 1 対多と多対多の 2 種類がある．

1 対多の再帰関連は，1 対多の関連と同様に RDB のスキーマは存在せず，実体のスキーマに主キー制約と外部キー制約を持つ（図 9 (d)，(e)）．1 対多の再帰関連は RDB のスキーマが存在しない．1 対多の関連と同様に変換すると，多側の実体は自分自身であるため，その変換は不要である．また外部キーによる参照先（1 側の実体）も自分自身（多側の実体）であり，その参照先の実体は変換済みと考えられる．よって本設計法において，1 対多の再帰関連の変換は発生しない（図 9 (f)）．

一方多対多の再帰関連は，多対多の関連と同様に RDB のスキーマが存在する（図 9 (g)，(h)）．しかし多対多の再帰関連を実体の子要素として階層関係に配置すると，次の問題点が発生する（図 9 (h)）．子要素に配置する表には 2 つの外部キーがあり，同じ表を参照するため，どちらの外部キーに従って配置すべきか決定できない．そのため本設計法では，多対多の関連を関連のリングと同類と見なし，参照関係として扱う（図 9 (j)）．

本設計法の変換規則の例外を以下に示す．

・1 対 1 の関連の参照と，1 対多の再帰関連の再帰参照を子要素としないこと．
・関連のリングとなる子要素（多対多の再帰関連も含む）は，参照関係とすること．

4. XML ビューの設計法

この章では，本提案の XML ビューの設計法について述べる．

4.1 XML ビューの設計法の概要

本提案の “関係データベース上の階層関係を持つ妥当な XML ビューの設計法” は，冗長性がなく，更新時異常を発生しない階層を持つ XML ビューを設計する．入力条件は次のとおりである．

・ユーザが RDB から利用する表のリストをあらかじめ決定できること．
・ユーザが階層関係のルート要素に配置する RDB 中の表をあらかじめ決定できること．
・RDB のスキーマが利用できること．

なお，ユーザが RDB の表のどの表がルート要素に配置すると決定しても冗長性は表れない．そしてルート要素に配置する表によっては，階層がない XML ビューと比較した削減率が最適なものが選べる．前提条件は次のとおりである．

・RDB 中の表は第 3 正規形（ただし関係がキーだけの関係の場合は第 4 正規形）であること．
・RDB 中の各表は 1 回しか利用しないこと．

ただし，本設計法は次の項目を考慮しない．

・階層関係が複数存在する XML ビュー
・3 項以上の関連を扱う E/R 図から変換した RDB のスキーマ
・1 対多の関連に属性を持つ E/R 図から変換した RDB のスキーマ
・多対多の再帰関連の階層関係の子要素への配置

この設計法は，次の 2 つからなる．

【関係データベース上の階層関係を持つ妥当な XML ビューの設計法】

手順 1 RDB 中の関連を XML の階層関係と参照関係に変換した xE/R 図を作成する（4.2 節）．

手順 2 xE/R 図から XML ビューの配置情報を作成する（4.3 節）． □

なお，本設計法から得られる XML ビューの配置情報が妥当なビューであるという証明を付録 A.1 に示す．

4.2 RDB 中の関連を XML 文書の階層関係と参照関係へ変換した xE/R 図の作成

この節では手順 1 のアルゴリズムについて述べる．

“階層関係と参照関係の探索決定アルゴリズム” は，RDB から利用する表と階層関係のルート要素に配置する RDB 中の表が決定できれば，それをルート要素とする階層関係と参照関係を決定し，その結果を xE/R 図で出力する．このアルゴリズムは 3.2 節とは違い E/R 図を用いない．代わりに正規化された RDB スキーマ中の主キーと外部キー制約を利用し，直接，xE/R 図を導き出す．アルゴリズムの概要を以下に，詳細を**図 10** に示す．なお，step3(1) のアルゴリズムは step2 とほぼ同じであるため，その詳細も省略した．また step2 の decideH の先頭に “※” を記した行は，3.4 節で述べた変換規則やその例外にない規則である．これらの行は外部キーの組合せが閉路となっていた場合，このアルゴリズムが無限ループに陥らないための対策である．たとえば，2 つの表 A，B があり，A が B を，B が A を参照していると閉路となる．以降では，閉路の処理に関して省略している．

【階層関係と参照関係の探索決定アルゴリズムの概要】

入力： RDB から利用する表名のリスト
入力： 階層関係のルート要素にする表名
出力： xE/R 図
階層関係の決定：

ルート要素とする表を外部キーで参照する表 R_C ごとに，ルート要素の子要素 X_C とする．R_C を外部キーで参照する表ごとに，X_C の子要素とす

```
入力　利用する表名のリスト，階層関係のルート要素にする表名
グローバル変数　探索決定表

【step 1 : 探索決定表の準備1】
 N個の表からxE/R図を作成するには，(N+2) × (N+2)マスの探索決定表
が必要になる．ここでNはRDB中の表の数とする．

(1) (N+2) × (N+2)マスの探索決定表を作成し，次のラベルを記入する．
      ・2行目2列目：表
      ・1行目2列目：主キー側
      ・2行目1列目：外部キー側
(2) 2行目の3列目から右方向に利用する表名を順に記入する．
      （表名の順序は任意）
(3) 2列目の3行目から下方向に利用する表名を(2)と同じ順に記入する．
(4) 全ての外部キー制約を探索決定表の対応する欄に「↑」を記入する．
    ただし，外部キー制約が主キー制約でもあるときは記入しない．

【step2 : 探索決定処理1 - 階層関係の探索決定 -】
 decideH(階層関係のルート要素にする表名,1) /* サブルーチンの実行 */

 sub decideH(親，親の階層レベル)
   変数 子，子の階層レベル := 親の階層レベル+1
   if 探索決定表[1行目，親列] ◇ “” then
     探索決定表[1行目，親列] += “M” /* 関連のリング(多対多再帰) */
   else
     探索決定表[1行目，親列] := 親の階層レベル
※   探索決定表[1行目，親列] += “P” /* 探索パスの再帰の防止用 */
     for 3行目の親列から「↑」を探索，下方向に進むこと
       if 「↑」を発見したら then
         子 := 探索決定表[発見した行，2列目]
         if 子◇親 then /* 1対多の再帰関連の確認(※により省略可) */
※          if 探索決定表[1行目，発見した行]の末尾 ◇ “P” then
             decideH(子，子の階層レベル)  /* 自分自身に再帰 */
           end if
         end if
     end if end for
※   探索決定表[1行目，親列]の末尾の “P” を削除
   end if
 end sub

【step3 : 探索決定表の準備2】
(1) 探索決定処理1で決定した階層レベルの数値の後ろに “M” が記入
  されているものは関連のリングの頂点を意味する．その頂点の階層
  レベルの数値を削除，つまり参照関係を意味する “M” に変更し，
  その頂点の子も全て削除する．
(2) 1行目の3列目以降を，1列目の3行目以降に複写する．
(3) 主キー制約でもある外部キー制約を探索決定表の対応する欄に
  「↑」を記入する．

【step4 : 探索決定処理2 - 参照関係の探索決定-】
 for 3行目の2列目から下方向にすべての表を調べる
   if 探索決定表[探索中の行,1列目] ◇ “”
              and 探索決定表[探索中の行,1列目] ◇ “M” then
     for 3列目から「↑」を探索，右方向に進むこと
       if 「↑」を発見したら then
         if 探索決定表[1行目,発見した列] = “” then
           探索決定表[1行目,発見した列] := “M”
         end if
     end if end for
   end if
 end for

【step5 : 探索決定表からxE/R図の作成】
 詳細は本文に記す
```

図 10 階層関係と参照関係の探索決定アルゴリズム
Fig. 10 Decision algorithm for retrieval of hierarchical relations and reference relations.

る．以下同様．ただし 1 対 1 の関連の参照と，1 対多の再帰関連の再帰参照を子要素としないこと．そして関連のリングとなる子要素（多対多の再帰関連も含む）は，参照関係とすること．

参照関係の決定：

階層関係の要素となる表から参照している表を被参照要素とする．ただし階層関係の要素としてその表が利用されていないこと． □

このアルゴリズムは，RDB のスキーマを利用して，階層関係と参照関係を決定している．利用するには 5 つの手順の実行が必要である．処理を順番に実行し，妥当な階層関係とその階層から参照される被参照要素を決定する．その処理とは，以下の step1 から step5 を順に実行することである．

step1：探索決定表の準備 1

利用する表のリストと 1 対 1 を除く関連の情報を用いて，グローバル変数「探索決定表」を初期化する．

step2：探索決定処理 1

階層関係の探索決定を行う．

step3：探索決定表の準備 2

関連のリングを参照関係に変更し，そして 1 対 1 を含む関連の情報の準備を行う．

step4：探索決定処理 2

参照関係の探索決定を行う．

step5：探索決定表から xE/R 図の作成

グローバル変数「探索決定表」から xE/R 図を作成する． □

たとえば，正規化された RDB 中の 3 個の表 A，B，C があり，B には A と C への外部キー制約が存在し，それぞれの外部キー制約だけでは B の主キー制約と等しくないときの，RDB から A，B，C のすべてを利用し，A をルートとした階層を取得する場合を**図 11** を使って説明する．

まず，step1 の探索決定表の準備 1 を実行し，図 11 (a) の探索決定表を作成する．このとき 1 対 1 の関連が階層関係として決定されないように，1 対 1 の関連の外部キー制約は記入しない．ここでは B のそれぞれの外部キー制約だけでは B の主キー制約と等しくないことから，1 対 1 の関連ではないと判断し，B から A と C への外部キーをそれぞれ 4 行目の 3 列目と 5 列目に「↑」を記入してある．なお，多対多の再帰関連の場合は，1 カ所に 2 つの「↑」が現れる．

次に，step2 の探索決定処理 1 を実行する．この処理は入力として，ルート要素にする表名を与える必要がある．ここでは入力が “A” であり，したがってサブルーチン decideH(“A”,1) を実行する．decideH では，図 11 (b) に示すように第 2 引数の親の階層レベルを探索決定表の 1 行目の親列に記入する．親列とは decideH の中で利用する変数であり，これは第 1 引数から導き出す．このとき探索決定表の 2 行目 3 列目から 5 列目を参照するとよい．そして親を参照している外部キーを，探索決定表の親列目の 3 行目から 5 行目まで調べ，それが 1 対 1 の再帰関連の外部キーでなければ再帰を利用し，decideH を再び実行する．結果，

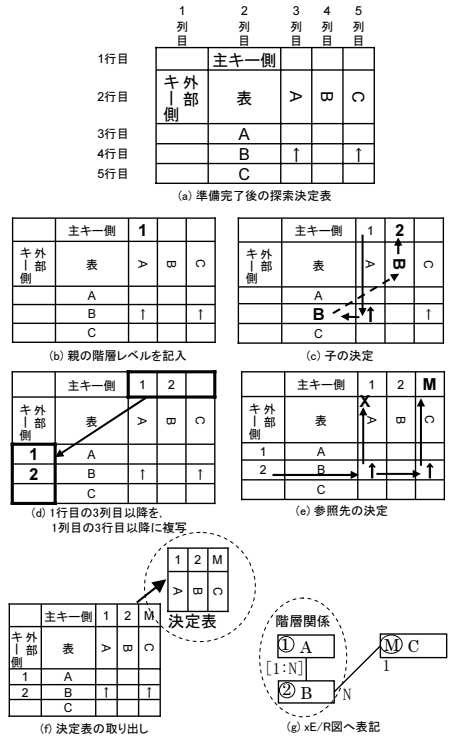

図 11 探索決定表の変化の様子
Fig. 11 Transition of a decision table in retrieving process.

図 11 (c) となる．なお decideH は関連のリングを見つけると，その関連を参照関係として扱うための処理として，階層レベルの後ろに “M” を追加する．

次に，step3 の探索決定表の準備 2 を実行し，図 11 (d) の状態にする．ここではまず，探索決定表の 1 行目の 3 列目以降に記されている step2 の結果を 1 列目の 3 行目以降に複写し，次に step1 で記入しなかった 1 対 1 の関連の外部キー制約を記入する．ただし step2 で決定した階層レベルの数値の後ろに “M” が追加されているものは，関連のリングの頂点を意味しているため，その頂点だけを参照関係を意味する “M” に置き換える．なお参照関係は階層を持たない．本設計法の関連のリングの処理は，2 パス形式となっている．step2 で関連のリングにマークし，step3 で関連のリングを除去する．step3 では，step2 でたどったパスを再び通る．本アルゴリズムはこの方法によりバックトラッキング（backtracking）の発生を防いでいる．

次に，step4 の探索決定処理 2 を実行し，図 11 (e) の状態にする．この処理は参照関係を探索し，参照関係となる表の列の 1 行目に “M” と記入している．ただし step3 までに確定している階層レベルは上書きしない．

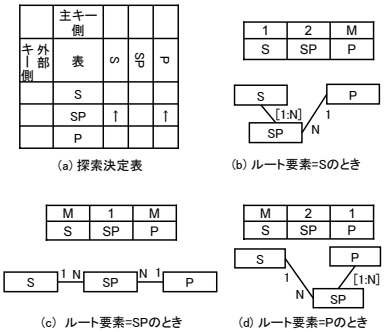

図 12 関連から階層関係と参照関係への変換例 3
Fig. 12 Third example of converting from relationships to hierarchical relations and reference relations.

最後に，step5 の探索決定表から xE/R 図の作成を実行する．これには 2 つの方法がある．1 つは，図 11 (f) に示すように決定表から図 11 (g) の xE/R 図を表記する方法である．決定表とは探索決定表の 1 行目と 2 行目の 3 列目以降のことであり，決定表には親子関係の階層レベルが「1 以上の数値」で，参照先の関係は “M” と示される．もう 1 つは探索決定表をそのまま利用する方法である．たとえば図 11 (f) の探索決定表の 4 行目に注目する．階層レベルが 2 レベルの B は，「↑」が 2 つ記述してある．この記述は関連を 2 つ持つことを意味している．それぞれの「↑」の列の 1 行目と 2 行目に注目すると「階層レベル 1 は A であり，参照関係 C がある」と判断できる．コンピュータに実装するときは，前者を要素の配置位置とし，後者を使って辺を描く．

xE/R 図のカーディナリティは，階層関係では親の要素側が 1 であり，子の要素側が N である．参照関係では，被参照要素側が 1 であり，参照要素側が N である．参照関係の中には 1 対 1 の関連および関連のリングによるものも含むために，RDB スキーマを再び確認する必要がある．

【探索決定アルゴリズムの実行例 1】

図 7 (a) の RDB スキーマにアルゴリズムを適用した結果を**図 12** に示す．図 12 (a) は，step1 の探索決定表の準備 1 を実行した後の探索決定表である．その 4 行目は SP から S と P への外部キーがあることを示している．図 12 (b)，(c)，(d) はアルゴリズムを実行した後の決定表と xE/R 図である．ルート要素の表の違いから，これら 3 つの図はそれぞれ異なる結果となっている．図 12 (b)，(d) は実体をルート要素とした場合であり，多対多の関連が子要素となり，その子要素は参照関係をともなっている．図 12 (c) は関連

関係の表をルート要素とした場合であり，結果として RDB と同様に階層がなく対等な構造となることを示している．

【探索決定アルゴリズムの実行例 2】

関連のリングが発生する例を**図 13** に示す．図 13 (a) は，以前示した図 7 (a) の RDB スキーマに CITY → STATUS の冗長な従属性が含まれていると仮定して，S を，S と CS の 2 つの表に分解したスキーマである．図 13 (b) はその E/R 図である．そして，CS は S，P の 2 つから参照されている．さらに，S，P の 2 つを SP が関連付けている．このとき，ルートを CS とする階層関係を導き出すと，SP が S と P の 2 つの部分木をつなげてしまう．

図 13 (c)–(e) は，アルゴリズム実行中の探索決定表を示している．図 13 (d) は，step2 の探索決定処理 1 を実行した後である．1 行目の 5 列目には，最後に文字 “M” が記されている．このことは，SP を階層関係の要素として変換すると関連のリングの状態になったことを示している．step3 は関連のリングを参照関係へ変更する．その結果，step3 の実行した後の探索決定表は図 13 (d) となる．図 13 (f) は，アルゴリズムの実行により得られた xE/R 図である．なお関連のリングによる参照関係は階層関係中に被参照要素を持つ．

【考察】

上記の 2 つの実行例を考察する．

(1) 親の要素は主キー側の表であり，子の要素は外部キー側の表である．

(2) 実体は 2 つ以上の子を持つ階層関係が存在する．

(3) RDB の表には 2 つ以上の複数の外部キーが存在できるため，階層関係に変換される要素は複数の親を持つ可能性がある．しかし本設計法において，関連のリングはその頂点を参照関係に変更するため，結果として，親は最大 1 つとして導かれる．

【アルゴリズムの時間計算量に関する考察】

アルゴリズムの時間計算量について考察する．

まず，アルゴリズムを Java で実装し，予備実験を行った．実験環境は OS に Windows2000 を用い，CPU は Pentium4 (1.7 GHz)，メモリは 512 MB を用いた．

実験手順は RDB スキーマに None 型，Loop 型，Line 型の 3 つを構築して行った．None 型は，外部キーが存在しないデータベースである．これはどの表をルート要素としても子要素を持つことはない．Line 型は，Loop 型の外部キーを 1 つ削除したものである．たとえば，A，B，C の 3 つの表が存在するデータベースの場合は，A が B を，B が C を外部キーを使って参照している．この構造は，平均すると表の半数が階層関係の要素となる．Loop 型は，すべての表で閉路が構成されたデータベースである．たとえば，A，B，C の 3 つの表が存在するデータベースのときは，A が B を，B が C を，C が A を外部キーを使って参照している．この構造はどれをルート要素として選んでも，すべての表が階層関係に配置された xE/R 図が計算される．**図 14** に RDB 中の表数が N ($N = 100, 200, \ldots 1000$) 個のときの N 個の xE/R 図の計算時間を示す．なお，RDB から利用する表が N 個指定されたとき，ルート要素として配置できる表の数は N 個である．図 14 の計算時間はこの N 通りの計算をすべて行った時間を示している．たとえば，利用する表数が 1,000 個のときは，1,000 個の xE/R 図を計算し，そのときの計算時間がグラフの値となっている．None 型は表の数

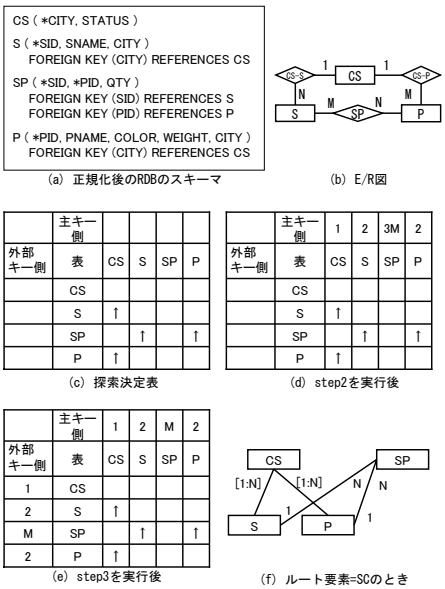

図 13 関連から階層関係と参照関係への変換例 4

Fig. 13 Fourth example of converting from relationships to hierarchical relations and reference relations.

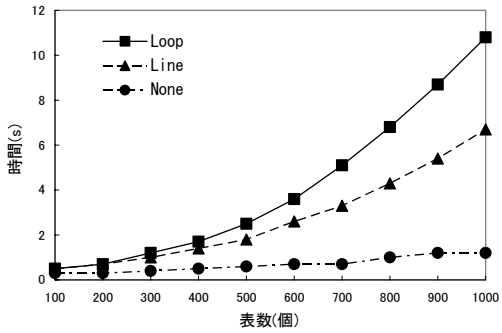

図 14 xE/R 図の計算時間

Fig. 14 A time of computing for xE/R diagrams.

が増えても計算時間は Line 型や Loop 型に比べ増加していないことから，RDB の外部キー数が多いほど計算時間が大きくなることが予想される．

そこで，主処理である step2 で実行するサブルーチン decideH に注目し，探索決定表から外部キーを調べる回数を確認する．decideH の中には for 文によって外部キーを RDB から利用する表の数だけ調べる．そして decideH は xE/R 図の階層関係の要素に決定した表の数だけ再帰する．なお，変換規則の例外処理を実現する decideH の 2 行目の関連のリングの処理などによって，階層関係の要素に決定する表の数は最大で RDB から利用する表の数と等しい．さらにこのアルゴリズムは，関連のリングの処理などが発生しても，要素として決定した表を無効にするなどの処理，つまりバックトラッキングをしない．よって，本アルゴリズムの step2 の時間計算量は，多項式オーダである．

その他の step1，3，4，5 については省略するが，すべて多項式オーダとなっている，ゆえに本アルゴリズムは多項式時間アルゴリズムである．

4.3 xE/R 図から XML ビューの配置情報の作成

手順 2 は XML ビューの配置情報として，手順 1 で作成した xE/R 図を従属性で表記する．このとき階層関係中の子は必ず親への外部キーの属性を持つことを利用する．これについては 4.2 節の考察 (1) で述べた．以下に手順 2 の詳細を示す．

【xE/R 図から XML ビューの配置情報の作成の詳細】

xE/R 図の中の階層関係を，拡張関数従属性で表記する．このとき，属性の無駄な重複の削除として，繰返しグループから親の主キーを参照する外部キーの属性を削除する．

子要素を持たない階層関係および被参照要素（関連のリングによるものは参照要素）は繰返しグループを持たないため，関数従属性で表記する．

5. 本設計法の RDB 上の XML ビューへの適用

本設計法を RDB 上の XML ビューへ適用した評価実験を行った．この章では実験概要を述べ，実験結果について考察する．

【実験概要】

実験では RDB 上の XML ビューとして，Java で開発した変換ツールを用いた．このツールは 4.3 節で述べた本設計法の結果を XML ビューの配置情報として扱い，そして RDBMS と連携して関係データを XML 形式に変換できるように実装している．

RDBMS には図 7 と図 13 で示した Date[1] の 2 種類のデータベースと TPC–H[11] を用いて構築した．図 7 を実験 1 で利用し，図 13 を実験 2 で利用する．TPC-H を実験 3 で利用する．階層のない表現と階層を持つ表現を比較するため，各実験ごとに 2 つのビュー定義を用意した．実験データを以下に示す．

実験 1 S，P，SP のタプル数：5，6，12

View 1-1. 階層関係なし

S : SID → SNAME, STATUS, CITY.
P : PID → PNAME, COLOR, WEIGHT, CITY.
SP : SID, PID → QTY.

View 1-2. 階層関係あり，図 7 (e)

S : SID ⇒ SNAME, STATUS, CITY, SP(PID, QTY).
P : PID → PNAME, COLOR, WEIGHT, CITY.

実験 2 CS，S，P，SP のタプル数：4，5，6，12

View 2-1. 階層関係なし

CS : CITY → STATUS .
S : SID → SNAME, CITY.
P : PID → PNAME, COLOR, WEIGHT, CITY.
SP : SID, PID → QTY .

View 2-2. 階層関係あり，図 13 (f)

CS : CITY ⇒ STATUS, S (SID, SNAME), P (PID, PNAME, COLOR, WEIGHT).
SP : SID, PID → QTY.

実験 3 フィールド数が多いため，ビューの詳細は省略する．

テーブル名 : フィールド数 : タプル数

CUSTOMER : 8 : 150,	LINEITEM : 16 : 6005
NATION : 4 : 25,	ORDERS : 9 : 1500
PART : 9 : 200,	PARTSUPP : 5 : 700
REGION : 3 : 5,	SUPPLIER : 7 : 10

なお XML 形式への変換時において子要素または属性に配置する 2 種類の方式がある．本実験では後者の属性に配置する方式を利用した．そして文字列に対する参照ポインタで文字列を共有することで多くの場合，XML ビューの記憶量を削減できる．しかし本実験では標準的な XML パーサを用いたアプリケーションが受け取る XML 文書を想定しているため，この種の圧縮は前提としていない．

【実験結果と考察】

実験結果を**表 1** に示す．実験 1，2 ともに階層関係

を持つビューの方が，インスタンスのサイズを減少させた．しかし子要素となる数が少ない実験 2 の方が削減率が大きい．これは 1 回の親子関係で減少させる量 G は次式 (1) となるためである．

$$G \approx (L + l + 4) \times n - (P + 2) \times m \quad (1)$$

ただし，

L：キーの属性名の文字数
l ：キー値の平均文字数
n：子のタプル数
P：親の要素名の文字数
m：利用するキー値の種類の数 $(n \geq m)$

とする．なお，式 (1) の 4 は子要素から削除される属性リストの区切り文字「(空白)」，属性名と属性値の間にある「=」，属性値を囲む 2 つの引用符「“”」の合計 4 文字である．そして 2 は親要素の開始タグから削除される「/」の 1 文字と追加される終了タグの「</」，「>」の合計 3 文字の差 2（$= 3 - 1$）文字である．

要素名や属性名の文字数は固定であるとすると，第 1 項からキーのサイズまたは子のタプル数が増加すると削減量が増加し，第 2 項から，利用するキー値の種類が増加，つまり同じキーの利用が少ないと削減量が減少する．親の要素名が長い $(L+l+4) < (P+2)$ のとき，逆にインスタンスのサイズが大きくなることがあるが，その増加量は 0 に近い値である．たとえば，m は最大で n と等しい場合，次式 (2) となる．

$$G \approx (L + l + 2 - P) \times n \quad (2)$$

しかし，本設計法では [1:1] の階層関係を許さないため，m が n と等しくなることは [1:n] の階層関係では，可能性は低く，そのようになるのは子のタプル数が小さいときであるといえる．

よって，本設計法は XML 文書のサイズを減少させることができると考える．

次に，実験 3 について考察する．8 つの表を各々ルートとしてビューの設計を繰り返し，XML 文書を作成した．その結果は，削減率は 0〜6.2%（削減量は 0〜181,058 byte）であり，ルートに SUPPLIER を選んだときが最も高く，LINEITEM のときが最も低かった．表 1 の実験 3 には，実験結果の削減率の最も高い結果を記した．実験 3 は実験 1，2 より，タプル数が多くかつフィールド数も多いが，削減量は 0 を下回ることはなかった．

また，本設計法はルートの選び方によって削減率が異なる結果となったが，削減率が高い表を選ぶべきであるか否かについて考える．ユーザはビューとして見たい情報についてある目的を持っていると考えられる．たとえば，SQL 文の SELECT 文では，FROM 句が RDB から利用する表のリストに相当する．同様に，XML 問合せから見たい情報が定まるはずである．実際，削減率を高めてルートを選んでしまうと，ユーザの目的に合わない可能性がある．XML ビュー設計の自動化は，ある程度，自由度が必要と考えられる．

表 1 XML 文書のサイズ
Table 1 Size of an XML document.

	表の数	階層関係なし	階層関係あり	削減率
実験 1	3	1,042 byte	947 byte	9.1
実験 2	4	1,103 byte	971 byte	12.0
実験 3	8	2,742,423 byte	2,923,329 byte	6.2

6. 関連研究との比較

RDB 上の XML ビューの冗長性をデータ構造変換時と階層構造変換時の 2 つに分類し，そして RDB データソースから XML 文書へのデータ構造変換における冗長性の発生について指摘してきた．この章では本設計法と関連研究を比較する．

【XML ビューの妥当性の確認に関する関連研究】

XML ビューの設計に関する研究として，Chen らの先行研究[14] がある．XQuery[13] などを用いた XML 問合せは，従属性を変更する可能性がある．そのため従属性を維持することが妥当なビューであるとし，妥当な XML ビューの設計手法を提案している．この手法は，データモデルに ORA-SS[2] を利用し，ビュー定義言語と問合せ処理に XQuery を利用する．そして XML ビューの作成のために XML 問合せを選択，集計，結合，交換の 4 つに分類し，それら 4 種類の XML 問合せに対応する XML ビューの妥当性を確認するアルゴリズムを提案している．

本論文の妥当性の確認は，RDB データソースから XML データソースへのデータ構造変換時の確認である．一方 Chen ら[14] の妥当性の確認は，XML データソースから XML 文書への XML 問合せにおける階層構造の変更の確認である点が異なる（**表 2**）．

【正規化に関する関連研究】

本設計法は無駄なデータの重複の確認のため NF–N3 の条件を利用する．しかし本論文では NF–N3 だけでは外部キーの無駄な重複が発生することを指摘し

表 2 冗長性の分類
Table 2 A classification of redundancy.

区分	要因	先行研究
階層構造変更時（XML 問合せ）	従属性を満たさない可能性	Chen ら[14]
データ構造変換時（RDB → XML）	階層関係と参照関係の選択を誤る可能性	

表 3 関係データを含む XML 文書の正規化手続きの比較
Table 3 Comparison between normalization procedures for an XML document.

	本提案手法	Du らの手法	XNF の手法
入力	RDB スキーマ 関係データ 従属性の情報	関係データ 従属性の情報	DTD 従属性の情報
基準	NF-N3	なし	BCNF(,NNF)
中間形式	xE/R 図	ORA-SS	なし
出力	配置情報 （従属性の情報）	XML Schema	DTD

た．さらに外部キーの無駄な重複となる冗長性は，拡張関数従属性のキーとの自明な従属であることを示し，それを XML ビューの設計法に適用した．よって外部キーの無駄な重複の判定基準に応用している点が Ling[10] と異なる．

関係データを含む XML 文書の正規化手続きの比較を**表 3** に示す．関係データを含む XML 文書の正規化手続きの関連研究として，Du ら[12] と XNF[7] の先行研究がある．Du ら[12] は，正規化されていない関係データから ORA-SS へ変換し，属性間における従属性を利用し，この ORA-SS から冗長性を取り除く正規化処理を行い，XML Schema[3] を得る手法が研究されている．XNF[7] では，関数従属性を入れ子関係に適用可能にしたものを利用し，任意の文書型定義（DTD）[4] から XNF を満たす DTD を得る正規化手続きが研究されている．XNF は BCNF を利用し，入れ子関係の正規形 NNF[16] の条件も満たす正規形である．

Du ら[12] には変換手順は示されているが，（正規形などの）基準が示されていないため，変換手順の結果が良い設計であるか確認ができない．一方，本設計法は妥当性の基準を設定し，この設計法から得られる従属性がつねに妥当なビューの条件を満たすことを確認している点が異なる．

そして，本設計法は正規化された RDB スキーマ，特に主キーと外部キーの制約を利用し，階層がない関係データから階層を持つ関係データへ正規化を行う．本正規化手続きは，すでに正規化されている関係データからのビュー設計法である．それは RDB を構築するときに行う正規化を再利用し，階層を持つ構造の正規形を得る手順である．Du ら[12] および XNF[7] の正規化手続きは意味的な従属性を抽出する作業は人間が行わなくてはならない．一方本設計法が利用する RDB スキーマにはこの情報が機械入力可能な形式で抽出済みであるという大きな利点がある．

さらに，XNF の正規化手続き[7] は，図 1 (a) のような冗長性を除去できるが，正規化した結果は外部キーの無駄な重複を含む図 5 (a) を導出する．5 章の実験結果が示すように，階層関係を持つビューを導出する本設計法の方が XML 文書のサイズを減少させることができる利点がある．

XML 文書の関係データを RDB の格納に写像する研究として，RRXS の先行研究[15] がある．この研究は本論文と補間的であり，XML から RDB へのデータ構造変換を行う．RRXS[15] は，XNF[7] と同様，XML へ適用できる関数従属性を定義し，XML 文書中の関係データを正規化しながら RDB へ格納する手法を提案している．RRXS の正規化手続き[15] は，自動的にとらえることができない意味的な関数従属性を事前情報として入力する必要があるが，DTD や XML 文書から自動的にとらえることができる関数従属性と組み合わせ，XML 文書中の関係データを第 3 正規形（3NF）へデータ構造変換を行い，RDB 中のデータサイズの削減を実現している．

RRXS の手法によって正規化された XML 文書中の関係データは，正規化された RDB に格納される．したがってこの手法によって格納された関係データが本設計法の RDB データソースとして利用できる．格納した関係データを XML 文書として取り出すとき，通信コストなどの削減のために本設計法が活用できると考える．

7. ま と め

本論文では RDB 上の XML ビューから冗長性のない XML 文書を取り出すために，関係データベース上の階層関係を持つ妥当な XML ビューの設計法を提案した．この本提案の設計法のため，次の 3 点について述べた．

1 つ目に，XML ビューの妥当性という概念を導入した．そのため従来の冗長性以外に，外部キーの無駄な重複の冗長性を，本設計法が拡張関数従属性に基づいて除去することを示した．

2 つ目に，RDB データソースから XML 文書への配置情報を作成する “階層関係と参照関係の探索決定アルゴリズム” を開発し，さらに，このアルゴリズムを利用する設計法から得られる従属性が妥当なビューの条件を満たすことを証明した．この証明によって本設計法による XML ビューは更新時異状を発生しないことを保障した．

3 つ目に，本設計法の実験結果は，XML 文書サイズが階層を用いない方法より減少することを示した．したがって本設計法が通信コストを低減させる利点を持つことが明らかになった．

今後は，3 項以上の関連，属性を持つ 1 対 N の関連の変換，複数の階層関係の導出も考慮した XML ビュー設計法の研究を行う予定である．

参 考 文 献

1) Date, C.J. (著) 藤原 譲 (監訳)：原書 6 版データベースシステム概論，丸善 (1997).
2) Dobbie, G., Wu X.Y., Ling, T.W. and Lee, M.L.: ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data, Technical Report TR21/00, School of Computing, National University of Singapore (2000).
3) Thompson, H., Beech, D., Maloney, M. and Mendelsohn, N.: XML Schema Part 0: Primer (May 2001). http://www.w3.org/TR/xmlschema-0/
4) Bosak, J., Bray, T., Connolly, D., Maler, E., Nicol, G., Sperberg-McQueen, C.M., Wood, L. and Clark, J.: W3C XML Specification DTD. http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm
5) Shanmugasundaram, J., Tufte, K., He, G., DeWitt, D. and Naughton, J.: Relational Databases for Querying XML Documents: Limitations and Opportunities, *The VLDB Journal*, pp.302–314 (1999).
6) Fernandez, M., Tan, W.C. and Suciu, D.: Silkroute: Trading between relations and xml, *9th International World Wide Web Conference* (2000).
7) Arenas, M. and Libkin, L.: A normal form for XML documents, *Proc. PODS 2002*, Madison, Wisconsin (June 2002).
8) Chen, P.P.: The Entity-Relationship Model: Toward a Unified View of Data, *ACM TODS*, Vol.1, No.1 (1976).
9) Fagin, R.: Multivalued Dependencies and a New Normal Form for Relational Databases, *ACM TODS*, pp.262–278 (Sep. 1977).
10) Ling, T.-W.: A Normal Form for Sets of Not-Necessarily Normalized Relations, *Proc. 22nd Hawaii International Conference on System Sciences*, pp.578–586. IEEE Computer Society Press (1989).
11) http://www.tpc.org/
12) Du, W., Lee, M.-L. and Ling, T.W.: XML Structures for Relational Data, *WISE (1) 2001*: 151–160, 3–6 December 2001, Kyoto, Japan (2001).
13) XQuery 1.0. http://www.w3.org/TR/xquery/
14) Chen, Y., Ling, T.W. and Lee, M.L.: Designing Valid XML Views, *21st International Conference on Conceptual Modeling (ER'2002)*, October 7–11, Tampere, Finland, pp.463–477 (2002).
15) Chen, Y., Davidson, S., Hara, C. and Zheng, Y.: RRXS: Redundancy reducing XML storage in relations, *Proc. 29th VLDB Conference*, Berlin, Germany (2003).
16) Ozsoyoglu, Z.M. and Yuan, L.Y.: A New Normal Form for Nested Relations, *ACM TODS*, Vol.12, No.1 (1987).
17) 石川 博：解説 XML とデータベース—交換から格納・収納へ，IPSJ Magazine, Vol.41, No.1 (2000).
18) 川田 純，石川佳治，北川博之：リレーショナルデータベース上の XML ビューに対する外部関数を考慮した問合せ処理，情報処理学会論文誌：データベース，Vol.43, No.SIG 12 (TOD 16) (2002).
19) 増永良文：リレーショナルデータベース入門［改訂版］，サイエンス社 (2003).

付 録

A.1 証 明

ここでは関係データベース上の階層関係を持つ妥当な XML ビューの設計法から得られる XML ビューの配置情報が妥当なビューであることを示す．

まず，妥当なビューの条件 (1) を確認する．

外部キーの無駄な重複が存在するには，ある参照関係を同じデータの無駄な重複がない階層関係で表現できるときのみである．つまり，外部キーの無駄な重複は，参照関係で表現された階層関係中の参照要素と階層関係外の被参照要素の間で発生する以外にはない．つまり，外部キーの無駄な重複は，参照関係で表現された階層関係中の参照要素と階層関係外の被参照要素の間で発生する以外にはない．以下を証明する．

「命題 階層関係中の要素から参照関係にある被参照要素は，同じデータの無駄な重複がない階層関係で表現できない．」

ただし，上記証明においては以下は外部キーの無駄な重複に該当しないものとした．

1) 被参照要素どうしでの外部キーの無駄な重複
2) 被参照要素をルート要素とした配置
3) 関連のリングによる参照関係
4) 階層関係に変換するとカーディナリティが [1:1] のとき

1) は，本設計法が階層関係が複数存在することは考慮しないとしているからで，2) は，ユーザがルートに決定できることと矛盾するからで，3) は，XML 文書に変換ができないからで，4) は NF–N3 に基づいた変換規則に反するためである．

（証明） 階層関係中の要素から参照関係にある被参照要素は，同じデータの無駄な重複がない階層関係で表現できると仮定する．被参照要素 A と階層関係中の参照要素 B の参照関係が存在するとする．この参照関係は A の主キーと B の外部キーの関連を利用している．よって A を B の子として配置した階層関係は，[1:1] あるいは [N:1] である．これは子側が主キーであるので 1 となり，親側の外部キーは主キーになりうるので 1 または N となるためである．変換規則より，同じデータの無駄な重複がない階層関係で表現するには，親子のカーディナリティが [1:N] である必要があり，主キー側の表を子に配置すると，そのカーディナリティは [1:1] あるいは [N:1] となってしまう．これは仮定と矛盾する．ゆえに命題は証明された． □

次に，妥当なビューの条件 (2) を確認する．NF–N3 を満たす妥当なビューとなるには，階層関係と参照関係の集合が NF–N3 となっていることを示す必要がある．まず，以下を証明する．

「正規化された RDB 中の表は 3NF（ただし，表がキーだけの場合は 4NF）であり，RDB 中のそれぞれの表に必ず主キーが存在している場合，“拡張関数従属性に基づく階層関係を持つ妥当なビューの設計法”から得られる拡張関数従属性で表現された階層関係は NF–N3 の正規形である．」

Ling の提案する，拡張関数従属性で表現できる関係が NF–N3 であるための条件をあげる．

(1) 関係 R は少なくとも 1 個のキー K を持つこと．
(2) ルートの属性の関係は第 3 正規形（3NF）であり，ルートの属性すべてがキーだけの関係である場合は第 4 正規形（4NF）であること．
(3) 繰返しグループのキー K_G を持つ R の繰返しグループ G ごとに，G のすべての属性と K が NF–N3 である関係を構成できること．このとき $\{K_G\}$ または $\{K, K_G\}$ が関係のキーであること． □

手順 2 から階層関係は，以下の理由で上記の条件を満たす．

(1)’ 階層関係 R のルート要素となる表の主キーが，R のキー K である．

(2)’ それぞれの表は，正規化された RDB 中の表である．よって，ルート要素は 3NF（表がキーだけの場合は，4NF）である．

(3)’ 手順 2 の xE/R 図から拡張関数従属性への変換において，親要素となる表の主キーと等しい属性（本設計法では必ず外部キー FK である）を，子要素から削除している．その親要素は，ここではルート要素を意味する．よって，子要素から削除した属性（FK）は，(1)’ より K と等しい．

R の繰返しグループ G ごとに，構成される階層関係 $R_G = \{G, K\}$ は，手順 2 で削除した属性を復元する．よって，R_G のルート要素は RDB 中の表と等しいといえるため，R_G は NF–N3 である関係を構成できる．

このとき R_G のルート要素となる表の主キーは，FK と互いに素である場合は $\{K_G\}$ と等しく，また FK の真部分集合の場合は $\{K, K_G\}$ と等しい．よって $\{K_G\}$ または $\{K, K_G\}$ が R_G のキーである． □

同様に，それぞれの参照関係の被参照要素（関連のリングによるものは参照要素）も NF–N3 である．これについては明らかであるため，証明は省略する．また，設計法の前提条件として，RDB 中の各表は多くとも 1 回しか利用しないことから，各 NF–N3 である関係の集合での重複する従属性は存在しないことは明らかである．

したがって，関係データベース上の階層関係を持つ妥当な XML ビューの設計法から得られる XML ビューの配置情報は妥当なビューである．

(平成 16 年 6 月 20 日受付)
(平成 16 年 10 月 8 日採録)

（担当編集委員 都司 達夫）

武川 肇（正会員）

1991 年職業訓練大学校情報工学科卒業．雇用促進事業団（現雇用・能力開発機構）入社．2003 年より職業能力開発総合大学校勤務．東京都立大学研究生として XML データベースの研究に従事．

太田 学（正会員）

1994 年東京大学工学部電気工学科卒業．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）．同年学術情報センター研究開発部リサーチ・アソシエイト．2000 年より東京都立大学大学院工学研究科助手となり，現在に至る．情報検索，データマイニングとその Web への応用に興味を持つ．電子情報通信学会，日本データベース学会各会員．

片山 薫（正会員）

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．日本データベース学会会員．

石川 博（正会員）

東京都立大学大学院工学研究科教授．東京大学理学部情報科学科卒業．富士通研究所を経て 2000 年より現職．東京大学博士（理学）．著書に “Object-Oriented Database System”（Springer Verlag），“Database and Data Communication Network Systems: Techniques and Applications”（共著，Elsevier），『e–ビジネス技術入門教科書—ビジネスモデルと情報技術（IT）IT TEXT』(CQ 出版）等．国際論文誌 ACM TODS, IEEE TKDE，国際学会 VLDB，IEEE ICDE 等学術論文多数．1994 年情報処理学会坂井記念特別賞，1997 年科学技術庁長官賞（研究功績者）受賞．現在，情報処理学会データベースシステム研究会主査．情報処理学会論文誌：データベース共同編集委員長．International Journal Very Large Data Bases Editorial Board．日本データベース学会理事．仏国ナント大学招聘教授．電子情報通信学会，ACM，IEEE 各会員．