XMLスキーマ拡張手法の評価と選択基準

新川 香　大川 昌弘　山根 英彦　濱野 泰男　松浦 桂子　下佐粉 昭

Evaluation and Selection Criteria for XML Schema Evolution Methods

Kaoru Shinkawa, Masahiro Ohkawa, Hidehiko Yamane,Yasuo Hamano, Keiko Matsuura and Akira Shimosako

XMLが情報交換の標準として広く使用されるようになるにつれ，XMLをデータ交換のための一時的なフォーマットとして使うだけではなく，XMLデータそのものを蓄積して活用していくケースが増えている．XMLデータが多く蓄積された状態で要件を変更する場合，すなわち，XMLスキーマを拡張する場合，新旧XMLスキーマの管理，既存のXMLデータやXMLを操作するアプリケーションに対する影響を考慮する必要がある．本論文ではXMLスキーマを拡張する手法を，XMLスキーマの互換性と，それに伴う既存のXMLデータの対応方法の観点から3つに分類し，XMLスキーマの管理，既存XMLデータの対応，既存アプリケーションの拡張に関するメリット・デメリットを述べ，それぞれの選択基準や考慮点を示す．その基準や考慮点を示すことによって，XMLスキーマ拡張方法選択の一助とする．

As the use of XML as a standard for information exchange becomes standard, there are increasing needs regarding the utilization of XML as not only as a temporary medium of information exchange but also as stored data. If requirements change—in other words, if the XML schema change—for large amounts of XML data, consideration will need to be given to the management of old and new XML schema and the impact on existing XML data or relevant applications. In this paper, we categorized XML evolution methods into three categories based on XML schema and XML data processing, and provided criteria for the selection of these methods, including their respective advantages and disadvantages with regard to XML schema management, existing XML data processing and the development of existing applications.

Key Words & Phrases：XML, XML Schema, XPath, XQuery, DB2®, Oracle, XSLT, XMLスキーマ・エボリューション
XML, XML schema, XPath, XQuery, DB2, Oracle, XSLT, XML schema evolution

1. はじめに

近年，XML［1］の活用範囲は拡大し，XMLデータ・フォーマットとして採用するアプリケーションの増加に加え，企業における電子商取引などでも利用されるようになっている．XMLデータを共通フォーマットとして使用する場合，データ構造（スキーマ）を定義する必要がある．XML Schema（XMLスキーマ）は，標準化されたXML用のスキーマ言語（構造定義言語）であり，W3Cにより標準化され［2］，すでに広く利用されている．**表1**に標準化された例を示す．

このようにXML Schemaを用いてXMLデータが管理されることで，アプリケーション間や企業間で，正確なデータを，共通の認識のもと，やり取りすることが可能になる．XMLは，構造が柔軟に記述でき，さまざまな用途に使用できることから，ビジネス環境の変化に応じて最適な構造に拡張してゆくことが求められている．そのXMLの構造を拡張する時，既存のXMLデータの扱い方やそれらをアクセスするアプリケーションの対応も考慮しなければならない．

XML Schemaを採用している標準においても，拡張は必然的なものとしてその考え方が定義されており，その定

提出日：2010年5月7日　再提出日：2010年9月7日

表1. XML Schemaを採用した標準の例

名前	用途
XBRL［3］	財務帳票を電子媒体として流通させるための標準規格
FIXML［4］	証券取引で標準プロトコル（FIX）をXML形式にしたもの
ACORD［5］	非営利団体ACORDによって作成された保険業界の標準データ・フォーマット
NewsML［6］	ニュース素材をインターネット経由で配信するためのフォーマット
流通BMS［7］	流通業界における新しいEDIのガイドライン．データ・フォーマットにXMLを使用している
ebXML［8］	インターネット経由の企業間電子取引のための標準

義に基づきバージョンアップが行われているものもある.

例えば NewsML は，神戸新聞・中国新聞・京都新聞の統合データベースのフォーマットとして利用されており［9］，NewsML バージョン 1.0 を使用している．現在，次世代の要件を盛り込んだ NewsML G2 が発表されており，v1.0 と G2 ではスキーマに互換性がない（互換性については 2 章参照）ため，G2 への移行に際しては，既存の XML データの扱いを検討し，それに応じて既存のアプリケーションの対応を検討する必要がある.

また，流通 BMS（Business Message Standard）のマイナー・バージョンアップはスキーマの互換性を保ち，そのバージョンアップはその時点の最適な構造が定義できるようにスキーマの互換性を保たなくても良いと定義されている．現時点では，v1.1，v1.2，v1.3 が策定されており，スキーマの互換性を保ったままバージョンアップされている．そのため，既存データを変換することなく新規データと混在して使用することが可能であるが，その分，スキーマへの変更は小さいものに押えられている.

このようにスキーマの拡張方法によって，アプリケーションや XML データに影響を与えることが課題となっている .

本論文ではスキーマの拡張方法を 3 つの手法に分類し，それらのメリット・デメリットをアプリケーションの拡張の観点も含めて明らかにする．また，3 つの手法を適切に選択するための基準および考慮点を提案する．さらに，XML データを格納する XML データベースの例として DB2 や Oracle における XML Schema の対応例について述べる.

2. XMLデータの妥当性検査と互換性

XML Schema を拡張する際の検討事項となるスキーマの互換性について述べる.

XML データが特定の XML Schema のルール［2］に従っている状態を「妥当（valid）」と呼び，妥当かどうかを判断する作業を「妥当性検査（validation）」と呼ぶ［10］.

XML Schema を拡張する際，拡張前の XML Schema に妥当な XML データが，新しい XML Schema の妥当性検査に成功する場合，拡張後の XML Schema は拡張前の XML Schema と互換性がある．逆に，妥当性検査に失敗する場合，拡張後の XML Schema は拡張前の XML Schema と互換性がない［11］．例えば，拡張前の XML Schema にオプションとして要素を追加した場合，その要素はオプションのため，拡張前の XML Schema に妥当な XML データは，拡張後の XML Schema にも妥当となる．この場合，拡張後の XML Schema は拡張前の XML Schema に互換性がある.

XML Schema の互換性を維持するかしないかを手法の分類項目とする.

3. 手法の分類

スキーマの拡張手法を，以下の視点で分類する.

- スキーマの拡張に当たり，前述の互換性を維持するかしないか（「スキーマ互換性あり」「スキーマ互換性なし」）
- 拡張前のスキーマで妥当な XML データを拡張後のスキーマで妥当となるように変換するかしないか）「既存 XML データの変換あり」「既存 XML データの変換なし」）

表 2 に,XML Schema を拡張する手法の分類を示す.

表 2. XML Schema を拡張する手法の分類

	既存 XMLデータの変換あり	既存 XMLデータの変換なし
スキーマ互換性あり	該当なし	手法1
スキーマ互換性なし	手法2	手法3

「スキーマ互換性あり」の場合，既存の XML データを変換しなくても拡張された XML Schema の互換性が維持されることから，**表 2** の左上は「該当なし」となる.

よって，残りの 3 つの枠に当てはまる手法ですべて網羅されていることが分かる.

3.1 手法1: XML Schemaの拡張（互換性あり）

手法 1 は，XML Schema を拡張前の XML Schema と互換性を維持するように要素や属性を追加する．この手法では，互換を持たせるために，既存の要素や属性に対しては，その制約を緩和する以外の変更はしない．そして，新たに追加した要素や属性はオプションとして定義する（そのほかの条件については［12］を参照）.

この手法は,DB2［13］や Oracle［14］などでサポートされており，管理している XML Schema を互換性のある XML Schema と置き換えることが可能である.

3.2 手法2: XML Schemaの拡張（互換性なし）と既存データの変換あり

手法 2 は，XML Schema を互換性のないように拡張し，既存の XML データは，拡張された XML Schema に対して妥当となるように変換する.

互換性のない XML Schema の拡張は，さらに以下の

2 つに分類できる.

A) 新たに追加した要素や属性により互換性が失われる

B) 既存の要素や属性の変更・削除・移動などにより互換性が失われる

この手法も DB2 や Oracle などでサポートされている.

Oracle の場合は, データベースの 1 つの XML 列を 1 つの XML Schema のみで管理するため, XML Schema の拡張と XML データの変換を同時に行う. DB2 では, 1 つの XML 列を複数の XML Schema で管理することが可能なため, XML Schema の拡張と XML データの変換を別々に行うことができる.

3.3 手法3: XML Schemaの拡張(互換性なし)と既存データの変換なし

手法 3 は, XML Schema を互換性がないように拡張し, 既存の XML データは変換せず, 今まで通りの XML Schema を使用し続ける.

データ変換が行えない例としては, 発注履歴データなどをアーカイブして読み取り専用で管理している場合が挙げられる. データ変換可能であっても, 発注履歴データを変換すると, データの原本性維持の観点から好ましくない.

この手法も手法 2 の A, B と同様に分類される.

手法 3 では, 複数(新旧)の XML Schema を扱うため, XML データにアクセスするアプリケーションは, XML Schema のバージョン管理を行い, XML データに妥当な XML Schema に応じてアクセス・ロジックを変える必要がある(B の場合). DB2 では, データごとにどの XML Schema で妥当性検査が行われたかの情報を取得することが可能なので, その情報を元にアプリケーションが XML データによって XML Schema を切り替えて動作することができる.

既存の要素や属性を変更しない A の場合, 新たに追加した要素や属性に対する処理以外は同じロジックが使用可能である. さらに, 新たに追加した要素や属性の処理についても, その存在をチェックして処理することで, 異なる XML Schema に妥当な XML データの処理を 1 つのアプリケーションで実装してゆくことも可能である.

4. 手法の評価・比較

3 つの手法のメリット・デメリットをそれぞれ提示し, それらの評価と比較を行う.

4.1 手法1のメリット・デメリット

手法 1 によるスキーマの拡張は, 互換性を保つため, 旧スキーマで妥当性検査された既存の XML データを修正し, 再度妥当性検査する必要がなく, 新しい XML Schema に互換となる. そのため, 単一のスキーマのみを管理すればよいので, アプリケーション改修の影響が最小限であることが最大のメリットである. また, DB2 や Oracle では, 旧スキーマを互換性のある新スキーマに置き換えることが可能であり, 単一のスキーマとして扱うことができる.

手法 1 のデメリットは, スキーマの拡張の自由度が小さいことである. 例えば, 必須要素・属性の追加, 名前空間の変更, 要素の移動などが不可能である. 必須要素・属性の追加が必要で, 手法 1 を選択したい場合には, スキーマ定義では必須とせず, アプリケーション側で該当要素・属性の有無をチェックすることも技術的には可能である. この場合アプリケーションと XML Schema が密結合となるため, 運用保守上の注意が必要である.

4.2 手法2のメリット・デメリット

手法 2 によるスキーマの拡張のメリットは, スキーマの互換性を保持しないため, 変更の自由度が手法 1 と比較して高いことである. また, 既存の XML データを拡張後の XML Schema に妥当となるように変換するため, 手法 1 のように単一の XML Schema を管理するだけで良いこともメリットとして挙げられる.

デメリットは, 旧スキーマで妥当性検査された既存 XML データに変換作業が発生することである. 拡張前に必須でなかった, あるいは存在しなかった要素や属性が新スキーマでは必須となった場合, 必須となった要素や属性を既存の XML データに付加する必要がある. 付加する値は, 例えば, 同じ表中のリレーショナル列から取得したり, 参照する値がない場合は, デフォルトの値を定めたりするか, 値を空にするなどして対応する. このような既存 XML データを新スキーマに対応させる変換作業において, 変換ルールが定まらないようなケースでは手法 2 を実現するのは困難である.

3.2 節において述べたように, この手法は A と B に分類できる. A の場合は, 既存の要素・属性部分は変更がないので, その部分に対しての既存アプリケーションへの影響は手法 1 と同じで, 最小限に抑えられるといえる. B の場合は, スキーマの構造自体が変更されるので, 既存アプリケーション内の XML データ処理ロジック(例: DB から XML データ取得に利用する XQuery ,XML データ取得後に特定の要素や属性を取得するための XPath)を変更しなくてはならず, A の場合よりもアプリケーションへの影響度は大きい.

4.3 手法3のメリット・デメリット

手法 3 によるスキーマの拡張のメリットは，スキーマの互換性を保持せず，かつ既存の XML データの修正も必要ないため，変更の自由度が 3 つの手法の中で最も高いことである．

その代わりに複数（新旧）のスキーマを管理しなければいけないというデメリットが生じる．なぜならば，スキーマ，データ共に互換性がないため，旧スキーマで妥当性検査された既存の XML データを今後も保持し，それに対応する旧スキーマを保持する必要があるためである．

アプリケーションに関しても，新旧それぞれの XML Schema のバージョンに応じた XML データ・アクセス，処理ロジックが必要となる．実装方法としては 1 つのアプリケーション内で XML Schema のバージョンを判断し，処理を分岐させる方法と，XML Schema のバージョンごとにアプリケーションを独立させる方法があるが，いずれの方法でもアプリケーション保守性の観点で手法 1，2 と比較してデメリットとなる．

この手法も手法 2 と同様に A と B に分類できる．

A の場合，DB2 を使用すれば，アプリケーションへの影響度を手法 1 や手法 2 の A と同程度にすることが可能である．例として，XML で管理された顧客情報の XML Schema を A に基づいて拡張するケースを考える．顧客がシステムにアクセスした際に，保持している既存の XML データから顧客情報を表示させ，新規の XML Schema によって必須となった項目の値を保持していない場合，すなわち旧 XML Schema によって管理されているデータの場合，その項目の値の入力を促す画面を表示する．入力された顧客情報を新しい XML Schema で妥当性検査して更新することにより，データは新しい XML Schema に妥当となるように変換される．顧客情報を表示させるための照会はどの XML Schema を使用しているか意識する必要がなく，更新や追加時は新しい XML Schema を使用するため，アプリケーションの拡張は手法 1 や手法 2 の A と同様に容易である．

Oracle の場合，1 つの列では 1 つの XML Schema に妥当な XML データしか保持できないため，複数の XML Schema のデータを扱う場合，別の列あるいは別の表を用いて管理する必要がある．そのため A の場合でも，アプリケーションの拡張は複雑となる．

XML データの追加や更新時に，DB2 では XML Schema の妥当性検査をトリガーで行うことができる．それを使用することで，XML データを追加したり更新したりする際に，どの XML Schema で妥当性検査するかアプリケーションで指定する必要がなくなり，アプリケーションの品質に頼ることなく XML データの信頼性を高めることができる［11］．

4.4 手法の評価

これまでに述べた各手法のメリット・デメリットをまとめたものを**表 3** に示す．

それぞれの手法のメリット・デメリットを理解し，変更要件にあった手法を検討していくことが必要である．

5. 手法の適用基準

4 章では，XML Schema の拡張手法のメリット・デメリットを明確にしたが，さらに，ユーザーの要件を照らし合わせ，さまざまな視点からどの拡張手法を適用するか検討する必要がある．本章では，3 つの手法のメリット・デメリットを基に，要件に合った手法を選択する際の基準について，さまざまな視点から考察し，提示する．

5.1 データ特性に応じた選択

XML データを，その特性によってマスター・データとトランザクション・データに分類する．マスター・データは，顧客情報や製品情報などの静的で複数の業務から共通に使用されるデータと定義する．トランザクション・データは，発注情報や入荷情報など，業務の結果として生成される動的なデータと定義する．

マスター・データに対して新しい情報を付加する場合，すでに登録されているデータにも新しい情報を付加することが望まれる．例えば，顧客情報に新たな情報を付加する場合，既存顧客に対してもその情報を付加したい．従って，マスター・データの場合は，1 つの XML Schema で管理し，同じ情報をすべてのデータに持たせることが望ましい．ゆえに，複数バージョンのスキーマに従う XML データが存在しない，手法 1 または手法 2 を選択するのが妥当である．

表 3. 3 手法の相対的評価

	手法1	手法2	手法3
スキーマ変更の自由度	低（互換性を維持）	中（既存データを新スキーマに適合）	高
スキーマ管理の容易性	高（1つのスキーマを管理）	高（1つのスキーマを管理）	低（複数のスキーマを管理）
アプリケーション拡張・保守の容易性	高	A：高 B：中（旧スキーマ部分の変更が必要）	A：高～中※ B：低（スキーマごとにアプリケーションの保守が必要）

※ DB2を用いて，かつ，追加・更新時は新しいXML Schemaで妥当性検証する場合，容易性は高い

一方，トランザクション・データに対しては，すでに登録されているデータに対して情報を付加することが改ざんに当たる可能性がある．この場合，新しい XML Schema に合わせて変換することは望ましくない．ゆえに,既存データの変更が生じない，手法 1 または手法 3 を選択するのが妥当である．

5.2 データ変換の負荷に応じた選択

文書格納システムのように，格納されている XML データが多数かつ文書サイズの大きいものである場合，データ変換の負荷が大きくなるため，手法 2 を選択する場合はデータ変換の負荷を考慮する必要がある．データ変換は，変更前と変更後のスキーマのマッピング（図 1）が可能な場合，SQL UPDATE 文から XQuery［15］式を呼び出すか，ツール（InfoSphere Data Architect［16］, Altova MapForce［17］など）で XSLT［18］や XQuery を生成して使用することによって，データ変換を自動で行うことも可能である．DB2 の SQL UPDATE 文の場合，変更した要素のみを置き換えることが可能である（図 2）．ただし，名前空間を使用しており，その変更も行う場合，すべての要素が別のものとなるため，すべての要素をマッピングする必要がある（図 3）．

スキーマに新たに必須要素が追加された場合，変換後のデータに何のデフォルト値を割り当てるかなどマッピングを作成する段階で考慮する必要もある．リレーショナル・データベースと XML のハイブリッド・データベースの場合は，ほかのリレーショナル表からデータを取得してデフォルト値として使用するなどのシナリオも考えられる．

自動でデータ変換可能な場合でも変換中のエラーやミスも想定しなくてはならない．また，変換後のデータの妥当性検査でエラーになる場合も想定される．例えば，上記で述べたようにリレーショナル表からデフォルト値をセットしたときに,その値が桁あふれなど起こす場合など考えられる．データ変換対象の XML データが多数あり，変換後のデータに互換性でのエラーがあると，それぞれのデータに対してエラー解析が必要になるため，データ変換の負荷と共に考慮が必要である．負荷が大きく，データ変換が含まれる手法 2 を選択できない場合は，手法 1 や 3 を選択し，既存データを変更せずにスキーマ拡張することを検討する．

5.3 業界標準スキーマのポリシーに応じた選択

業界標準スキーマの更新管理は，標準策定団体（国際新聞電気通信評議会［19］など）により行われることが一般的である．業界標準スキーマに準じた XML データを扱うシステムでは，標準策定団体のポリシーに従い，

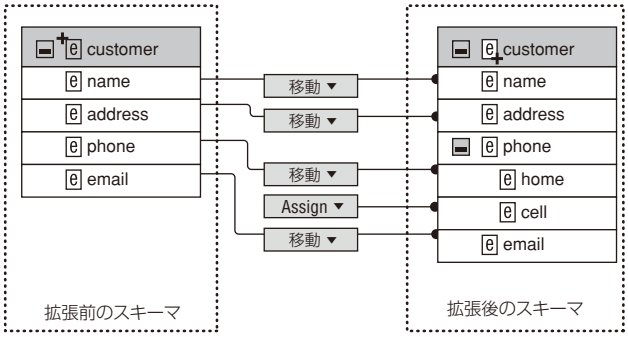


図1. スキーマ・マッピングの例

```
UPDATE T1
SET XMLDATA=XMLQUERY(
 'declare namespace xsi="http://www.w3.org/2001/XMLSchema-instance";
  copy $new := $XMLDATA
  modify (
    do replace $new/customer/phone with
      <phone>
        <home>{$new/customer/phone/text()}</home>
        <cell>{$new/customer/cell-phone/text()}</cell>
      </phone>,
    do delete $new/customer/cell-phone )
                return  $new')
```

図2. SQL UPDATE文の例（名前空間なし）

```
UPDATE T1
SET XMLDATA=XMLQUERY(
 'declare namespace xsi="http://www.w3.org/2001/XMLSchema-instance";
  declare namespace cust="http://www.sample.com/customer2";
  copy $new := $XMLDATA
  modify (
    do replace $new/customer with
      <cust:customer xsi:schemaLocation="http://www.sample.com/customer2
cust2.xsd">
        <cust:name>{$new/customer/name/text()}</cust:name>
        <cust:address>{$new/customer/address/text()}</cust:address>
        <cust:phone>
          <cust:home>{$new/customer/phone/text()}</cust:home>
          <cust:cell>{$new/customer/cell-phone/text()}</cust:cell>
        </cust:phone>
        <cust:email>{$new/customer/email/text()}</cust:email>
    </cust:customer> )
                return  $new')
```

図3. SQL UPDATE文の例（名前空間の変更あり）

最適な手法を選択する．

例えば，ニュースなどを配信する標準フォーマットである NewsML では，1.0 から 1.1, 1.2 といったマイナー・バージョンアップでは互換性を保証しており［20］，手法 1 が選択できる．2.0 へのメジャー・バージョンアップでは互換性がないと明記されており手法 1 は選択できない．

また，マイナー・バージョンアップでは，新しく追加された要素は旧バージョンのアプリケーションで無視すべきという記述があり，このガイドに従って実装されていれば，アプリケーションの変更は必要ないが，XML データの妥当性検査をしている場合，手法 1 に従って XML Schema を拡張する必要がある．

一方，流通 BMS では，XML テクニカルガイド［21］にマイナー・バージョン，メジャー・バージョンの互換性の記述に加えて，並行運用時のバージョン判断方法などの記述もあり，手法 3 に近い方法を想定している．

このように，策定団体により記述の詳細度はさまざまであるため,ポリシーをよく理解し手法を選択することが望ましい．

5.4 スキーマの更新度合いとシステム運用面の制約に応じた選択

スキーマの更新度合いとシステム運用面の制約も，最適な手法を判断する基準となりうる．

構造を大きく変えるようなスキーマの拡張（メジャー・バージョンアップ）が発生するが，長時間のメンテナンスが可能なシステムでありデータ変換負荷が問題になりにくい場合，手法 2 を用いてスキーマ拡張の都度，データを変換し，常に最新のアプリケーション，スキーマ，データの断面を持つことで,シンプルなバージョン管理の構成を取ることができる．

メンテナンス可能な時間の確保が困難なシステムに対して構造を大きく変えるような拡張を行う場合は，データ変換時の負荷の大きい手法 2 や構造を変更できる範囲が小さい手法 1 は選択が難しい．この場合は手法 3 を選択し，バージョン情報や XML データとの対応などメタ・データを管理するスキーマ管理システムを外部に構築しておくことが望ましい．XML Schema の拡張回数は多いが，各拡張での構造変化が少ない場合（マイナー・バージョンアップ）は，互換性を維持して手法 1 を選択することで，アプリケーションやデータの変換，システムのメンテナンスを最小限にとどめることができる．

普段はメンテナンス時間が確保できないが，年や期といった大きな単位でメンテナンス時間が設定されるようなシステムでは，手法 1 と手法 2 を組み合わせることで，スキーマの柔軟性とアプリケーションの保守の容易性という 2 つのメリットを享受できる．

5.5 開発規模に応じた選択

人員やステップ数などの開発規模が小さい場合，開発者が変更点を把握しやすいため，毎回アプリケーションの変更とデータの変換を行う手法 2 を選択し，スキーマの管理を容易とすることができる．開発規模が大きい場合は，将来の構造変化や，開発手法の標準化に備え手法 3 を選択し，スキーマ拡張の管理をシステム化しておくことが望ましい．

5.6 開発局面に応じた選択

開発サイクル内で,スキーマの拡張を行うことを考えると，開発の各局面において，スキーマの拡張がアプリケーション開発側へ与える影響やデータ・モデル設計の状況に応じた最適な手法を選択する必要があることが分かる．

一例として，ウォーターフォール・モデルでは，詳細設計局面でデータベースのスキーマ決定することが多くなるので，XML Schema についても同様のタイミングで全体的な決定を行う必要がある，データ・モデル設計に伴う構造変化が一段落するマイクロ設計の完了までは手法 3 で拡張し，構築フェーズ以降はアプリケーション側への影響を考慮し，手法 1 や手法 2（A）で実現できるような互換性の高い XML Schema の拡張が行われることが望ましい．

6. まとめと今後の課題

本論文では，XML Schema を保守するための手法を 3 種類に分類し，それぞれの特性を示し , さらに，さまざまな視点から拡張手法を選択する方法を提示した．実際の拡張要件やシステムの運用状況に合わせて適切な手法を選択することが重要である．ここで示した特性や考慮点をもとに，スキーマ拡張方法選択の際の一助になれば幸いである．

今後の課題として，手法 2 におけるデータの変換のパフォーマンスの比較や，手法 3 におけるスキーマとデータのバージョン管理方法の確立が挙げられる．これらについては，今後も議論を行い，考察を深め，提案を行いたいと考えている．

謝辞

本論文の執筆にあたっては，高橋 賢司氏，大沼 啓希氏に多くの助力をいただきました．改めてここに深謝の意を述べさせていただきます．

参考文献

［1］ W3C Extensible Markup Language (XML), http://www.w3.org/XML/
［2］ W3C XML Schema part1&part2, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/ , http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/
［3］ XBRL International, Extensible Business Reporting Language (XBRL) 2.1 (2003.12), http://www.xbrl.org/Specification/XBRL-RECOMMENDATION-2003-12-31+Corrected-Errata-2008-07-02.htm
［4］ FIXML Schema Version for FIX 4.4. Revision 1, http://www.fixprotocol.org/specifications/fix4.4fixml
［5］ ACORD XML Naming and Design Rules Candidate Recommendation V1.0.1, http://www.acord.org/standards/downloads/Pages/CDPublic1.aspx
［6］ JIS X 7201:2005「ニュース用マーク付け言語（NewsML）」(2005).
［7］ 流通システム標準普及推進協議会, 流通 BMS, http://www.dsri.

jp/ryutsu-bms/standard/standard01.html (2010).
［8］ ISO/TS 15000, 2005 Electronic business eXtensible Markup Language (ebXML) (2005).
［9］ 高乗昌人（京都新聞社）:「NewsML実装事例 神戸・中国・京都3社による統合データベース・システム」JST Global ID 200902214988055176 (2008.11.01).
［10］大川昌弘：DB2 による XML スキーマの管理と XML データの妥当性検証, developerWorks Japan, http://www.ibm.com/developerworks/jp/data/library/db2/j_d-xmlschemavalidation/index.html (2009).
［11］大川昌弘：DB2 による XML スキーマの進化と XML データの管理 , developerWorks Japan, http://www.ibm.com/developerworks/jp/data/library/db2/j_d-xmlschemaevolution/index.html (2009).
［12］IBM DB2 9.7 Information Center, “XML スキーマの展開のための互換性要件,” http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.xml.doc/doc/r0051287.html
［13］IBM DB2 9.7 Information Center, XML スキーマの展開, http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.xml.doc/doc/t0050684.html
［14］Oracle Technology Network, インプレース XML スキーマ・エボリューションの実行, http://www.oracle.com/technology/global/jp/obe/11gr1_db/datamgmt/xmldb2_a/xmldb2_a.htm
［15］XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/
［16］IBM InfoSphere Data Architect, http://www.ibm.com/software/data/optim/data-architect/
［17］Altova: Altova MapForce, http://www.altova.com/mapforce.html (2010).
［18］IBM Japan: “データ交換での XSLT の使用” : IBM DB2 Database for Linux, Unix and Windows インフォメーション・センター , http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.xml.doc/doc/c0051504.html (2010.02).
［19］International Press Telecommnucations Council, http://www.iptc.org
［20］NewsML 1.2 Guidelines 1.01, http://www.iptc.org/std/NewsML/1.2/documentation/NewsML_1.2-doc-Guidelines_1.01.pdf
［21］流通ビジネスメッセージ標準 XML スキーマ解説書 , http://www.dsri.jp/ryutsu-bms/standard/standard01_1.html

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
スタッフ・ソフトウェア・エンジニア

新川 香 Kaoru Shinkawa

［プロフィール］
2001年,日本IBM入社.ソフトウェア開発研究所においてInformation Management 製品開発に従事. 現在は, IBM InfoSphere™ Information Server 製品の DataStage Connectivity 開発を担当.

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
アドバイザリー・ソフトウェア・エンジニア

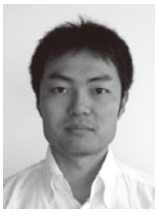
大川 昌弘 Masahiro Ohkawa

［プロフィール］
1989 年, 日本 IBM 入社. 入社以来, 各種アプリケーション開発, ミドルウェア製品開発, データベース関連製品開発などに従事. 現在は, IBM Optim™ 製品群の品質保証および支援業務を担当.

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
ソフトウェア・ラボ・サービス
IT スペシャリスト

山根 英彦 Hidehiko Yamane

［プロフィール］
2003 年ソフトウェア製品のベータ・プログラムを推進するエンジニアとして日本 IBM 入社. 現在 IT スペシャリストとして, Information Management 製品に関連したサービスのデリバリーを担当.



日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
ソフトウェア・ラボ・サービス
IT スペシャリスト

濱野 泰男 Yasuo Hamano

［プロフィール］
2006 年, 日本 IBM 入社. ソフトウェア開発研究所において DB2, solidDB などのデータベース関連サービスに従事. 情報処理学会会員. IT スペシャリスト.

日本アイ・ビー・エム株式会社
ソフトウェア事業部
主任 IT スペシャリスト

松浦 桂子 Keiko Matsuura

［プロフィール］
1997 年, 日本 IBM 入社. 製造業のお客様中心に Web アプリケーション構築プロジェクトに参画後, 2008 年よりソフトウェア事業部に異動. 現在はビジネス・パートナー様へのソフトウェア技術支援に従事.

日本アイ・ビー・エム株式会社
ソフトウェア事業部
IT アーキテクト

下佐粉 昭 Akira Shimosako

［プロフィール］
2001 年, 日本 IBM 入社. ソフトウェア事業において DB2 の技術支援業務に従事. 現在は, IT アーキテクトとしてビジネス・パートナー向け技術支援, 提案活動を行っている.