Oracle XML DBに見る統一スキーマの実現課題

鈴木　俊宏，秋本　尚吾

本稿では，Oracleデータベースの持つネイティブ XMLデータベース機能であるOracle XML DBをご紹介する．リレーショナルモデルとオブジェクト・リレーショナル・データベースの機能によって統一スキーマを実現するXMLモデルにどのように対応させるかを記す．最後に，統一スキーマ実現への課題を記す．

キーワード：Oracleデータベース，XML Schema，構造化マッピング，クエリ・リライト

1. はじめに

近年，B2B，B2CおよびEAIアプリケーションのデータ交換フォーマットの標準としてXML文書が受け入れられてきている．これは，XML文書の持つクロス・プラットホームでのデータ交換を円滑にし，データの抽象化を行うことのできる性質や，インターネット環境の発達によるものである．

XML文書でデータ交換を行うためには，受信側と送信側の意思疎通のためにXML用のスキーマ言語を利用したXML文書のフォーマット（以降，統一スキーマ）の制定は不可欠な要素である．このため，業界ごとに統一スキーマの制定が急ピッチで進められている．

本文書では，リレーショナルモデルとXMLモデルの関係を相互補完するOracle XML DBを例に取りながら，統一スキーマの実現可能性とその際の課題を議論する．

2. 統一スキーマとXML Schema

2.1 スキーマ言語

スキーマには，一般に概要，大意，図式といった意味がある．例えばデータベースにおいては，データの論理的構造の定義のことを指す．XMLのスキーマもこれとほとんど同じであるが，対象となるデータがXMLであるという点が異なる．つまり，XMLのスキーマとは，「XMLデータの論理構造の定義」である．

XML文書は，柔軟性が高くどのような構造を持った文書も格納できる反面，意図しない構造を持った不正なXML文書の格納も許してしまうという問題がある．したがって，企業間のデータ交換や電子商取引などでXML文書を使用している場合において，配送途中で壊れるなど何らかの理由で間違ったXML文書が与えられた際に，人間の手を介さずにチェックすることがないため，不向きである．

この「XML文書がどのような構造を持っているか」や「作成者がどのような意図を持ったXML文書を想定しているか」を表すために使用されるのがスキーマ言語である．スキーマ言語によって，XML文書の取り得る構造を記述することが可能となる．

2.2 DTDとXML Schema

XML用のスキーマ言語の主なものとして，DTDとXML Schemaを挙げることができる．

XMLの前身となったSGML（Standard Generalized Mark-up Language）で使用されていたスキーマ言語がDTD（Document Type Definition）[1]である．HTMLの定義などにもDTDが使用されてきたが，データ型が定義できない点や，文法がXMLと無関係である点などの欠点があった．この欠点は，XMLをSGMLと同様にドキュメント記述用に利用する場合にはあまり問題がなかったが，データ交換などのためのデータ記述用に利用する場合には非常に問題となっており，より強力でXMLの仕様と整合性のあるスキーマ言語の開発が待望された．

XML Schema[2]は，2001年5月にW3Cにより勧告になったスキーマ定義言語で，DTDの持つ欠点を補うべく，次のような機能が提供され，XML文書の内容に関して厳密に定義することが可能となった．

・要素や属性の値のデータ型の指定
・新しいデータ型の作成

すずき　としひろ，あきもと　しょうご
日本オラクル㈱
〒102-0092 千代田区紀尾井町4-1 ニューオータニガーデンコート

・データに対する制約の指定
・異なる名前空間の利用

2.3 統一スキーマの必要性

XML Schemaの持つ特徴から，現在ではXBRLなど多くの統一スキーマがXML Schemaを用いてスキーマ定義を作成している．

誰もが好き勝手にスキーマを表現すると問題が生じるのは言うまでもない．XMLデータを交換する2者間でスキーマに関する共通の理解がなく，要素名，属性名，出現回数，データ型がバラバラなデータをやり取りしても意味はない．したがって，スキーマの表現を共通化したルールに基づいて記述する必要があるのだ．これが統一スキーマの意義である．

3. Oracleデータベース

3.1 リレーショナル・データベース

リレーショナル・データベースは，1970年にIBMのEdgar Frank Coddが「A Relational Model of Data for Large Shared Data Banks」という関係モデルの概念に関する論文を発表したことがきっかけで，急速に広まったデータベースである．情報を表（テーブル）としてまとめ，その表の集まりをデータベースとしている．

関係モデルであるがゆえに，ツリー構造を持つXML文書との相性は通常あまりよくない．そのため，リレーショナル・データベースのベンダーによってXML文書に対する様々なアプローチが取られている．本文書では，リレーショナル・データベースとして1979年に世界初の製品化を行ったOracle Corporation（当時はRelational Software Inc.）のOracleデータベースのアプローチについて考えてみる．

3.2 オブジェクト・リレーショナル

Oracleデータベースは，もともとはリレーショナル・データベースとしての機能のみを提供していたが，世の中のニーズの変化に対応し，現在は単なるリレーショナル・データベースとしてではなく，様々な機能を提供している．

オブジェクト指向の考え方が普及した頃には，オブジェクト指向データベースの重要性が問われ，Oracleデータベースは，1997年にOracle 8のリリースとともにオブジェクト指向データベースに対応した．これによって，多段ネストした構造や独自のメソッドを持つユーザ定義のデータ型を作成し，リレーショナル・データのようにSQL文で検索するといったことを可能とした．

このオブジェクト指向データベースの機能はリレーショナル・データベース上に実装されているので，オブジェクト・リレーショナル・データベースと呼ばれている．

4. Oracle XML DB

4.1 Oracle XML DBとは

近年はXMLデータベースの必要性が問われているため，Oracleデータベースは，オブジェクト・リレーショナル機能を強化し，2000年末にOracle 9*i* Databaseリリース1において，XML文書を格納するためのXMLTypeデータ型（以降はXMLTypeと表す）を実装し，XMLデータベースとしての基本的な機能を搭載した．

Oracle XML DBは，Oracle 9*i*Databaseリリース2より搭載された本格的なネイティブXMLデータベース機能である．なお，本稿では，2005年3月現在の最新リリースであるOracle Database 10*g*リリース1をベースとして議論を進める．

OracleデータベースのXML対応の特徴として，あくまでもリレーショナル・データベースの特徴を生かしていることを挙げられる．Oracle XML DBの機能も，実際の処理はリレーショナル・データベース上で動作している．その理由は，Oracleデータベースがリレーショナル・データベースで培った高い信頼性とパフォーマンスを損なわないまま，ユーザに利便性を提供するためである．

4.2 Oracle XML DBにおけるスキーマ言語の利用

Oracle XML DBでは，スキーマ言語としてDTDおよびXML Schemaをサポートしている．これらのスキーマ言語を用いることにより，XMLTypeに格納するXML文書が，特定の構造に従っていることを保証することができる．

XMLTypeはオブジェクト型のデータとして保存され，格納方式によって構造化マッピング方式と非構造化マッピング方式に分類される．非構造化マッピング方式の場合にはXML文書は内部的にCLOB型の

表1 スキーマ言語と格納方法

スキーマ言語	格納方法	
	非構造化	構造化
なし	✔	
DTD	✔	
XML Schema	✔	✔

データとして保存され，構造化マッピング方式の場合にはXML文書は細分化されて，データ（要素のテキストおよび属性値）のみがオブジェクト型の表または列の一部に格納されている．

スキーマ言語を使用しない場合およびスキーマ言語としてDTDを使用した場合には，格納方式には非構造化マッピング方式のみしか利用できないが，スキーマ言語としてXML Schemaを使用した場合には，構造化マッピング方式と非構造化マッピング方式の両方が利用可能となっている（表1）．

5. 構造化マッピング

5.1 XML Schemaの登録とSQLオブジェクト型の生成

Oracle XML DBにXMLスキーマ（本文書では，XML Schemaの仕様に従って記述された実際のスキーマをXMLスキーマと表す）を登録すると，その構造に対応したSQLオブジェクト型をOracleデータベースの持つオブジェクト指向データベース機能を用いて作成する．これら一連の処理は，Oracle XML DBによって自動的に実施され，ユーザは気にする必要はない．

リスト1およびリスト2に本文書中で使用するXMLスキーマとその登録時に生成および実行されるSQLオブジェクトを示す．なお，SQLオブジェクトはOracleデータベース内部で実行されるSQL文を整形したものである．

Entry 182_Tオブジェクト型は，Entry複合型の構造を元に作成されている．Entry 183_COLL可変配列（VARRAY）はEntry複合型が複数回出現（maxOccurs="10000"と1より大きい）することから作成されている．

DayEntry 181_Tオブジェクト型は，DayEntry複合型の構造を元に作成されており，そのメンバには上記で作成したEntry 183_COLL可変配列が含まれる．

5.2 単純型（simpleType）

リスト2のEntryId，TitleやBody要素に対応して作成されたSQLデータ型のように，XML Schemaで定義された単純型をSQL 99のSQLオブジェクト型にマッピングしている．次は代表的な例（表2）だが，XML Schema Part 2[3]で事前に用意されたデー

リスト1　XMLスキーマ dayentry.xsd

```
<?xml version="1.0" encoding="Shift_JIS"?>
<xsd:schema
    targetNamespace="http://xmlns.oracle.co.jp/dayentry"
    xmlns="http://xmlns.oracle.co.jp/dayentry"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
 <xsd:element name="DayEntry">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="Date" type="xsd:date"/>
    <xsd:element name="Entry" maxOccurs="10000">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="EntryId" type="xsd:integer"/>
       <xsd:element name="Title">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="80"/>
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="Body" type="xsd:string"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

リスト2　実行されたSQL文

```
CREATE OR REPLACE TYPE "Entry182_T"
AS OBJECT (
  "SYS_XDBPD$" "XDB"."XDB$RAW_LIST_T",
  "EntryId"       NUMBER(38),
  "Title"         VARCHAR2(80 CHAR),
  "Body"          VARCHAR2(4000 CHAR)
) FINAL INSTANTIABLE;

CREATE OR REPLACE TYPE "Entry183_COLL"
AS VARRAY(10000) OF "Entry182_T";

CREATE OR REPLACE TYPE "DayEntry181_T"
AS OBJECT (
  "SYS_XDBPD$" "XDB"."XDB$RAW_LIST_T",
  "Date"          DATE,
  "Entry"         "Entry183_COLL"
) FINAL INSTANTIABLE;
```

表2　XML Schema ビルトインデータ型と SQL オブジェクト型

XML Schema	SQL	
	デフォルト	指定可能
string	VARCHAR2	CHAR, CLOB
float	NUMBER	FLOAT, DOUBLE, BINARY_FLOAT
dateTime	TIMESTAMP	TIMESTAMP WITH TIME ZONE, DATE
date	DATE	TIMESTAMP WITH TIME ZONE
boolean	RAW(1)	VARCHAR2
anyType, anySimpleType	VARCHAR2(4000)	CHAR, CLOB

タ型（ビルトインデータ型）が44個である一方，Oracle データベースの持つ SQL オブジェクト型は22個であるため，対応は一対一ではなく，最も近い SQL オブジェクト型にマッピングしている点に注目できる．

表2で指定可能とされている SQL オブジェクト型を使用するように指定するには，XML DB 名前空間 http://xmlns.oracle.com/xdb を使用する．XML DB 名前空間の SQLType 属性を使用したい SQL データ型の名称として指定する．例えば，リスト1の一部を次のように変更することで，Date 単純型の値を SQL データ型の DATE 型ではなく，TIMESTAMP WITH TIME ZONE 型で格納するようマッピングする．

```
<?xml version="1.0" encoding="Shift_JIS"?>
<xsd:schema
    targetNamespace="http://xmlns.oracle.co.jp/dayentry"
    xmlns="http://xmlns.oracle.co.jp/dayentry"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    xmlns:xdb="http://xmlns.oracle.com/xdb">
...
    <xsd:element name="Date" type="xsd:date"
xdb:SQLType="TIMESTAMP WITH TIME ZONE"/>
```

このように，通常は変更しなくとも良いが，Oracle XML DB の持つデフォルトの動作を変更したい場合には，XML DB 名前空間の機能で指定することとなる．

5.3　複合型（complexType）

リスト2の DayEntry 181_T オブジェクト型や Entry 182_T オブジェクト型のように，XML Schema で定義された複合型は次のように SQL 99 の SQL オブジェクト型にマッピングする．

・複合型内で宣言された XML 属性は，オブジェクト属性にマップされる．XML 属性を定義する単純型によって，対応する属性の SQL データ型が決定される．

・複合型内で宣言された XML 要素も，オブジェクト属性にマップされる．XML 要素を定義する単純型または複合型によって，オブジェクト属性のデータ型が決定される．

また，複合型の継承は，SQL オブジェクト型の持つ継承機能により実現している．

5.4　コレクション

リスト2の Entry 182_T オブジェクト型のように，複合型の maxOccurs 属性が1より大きい場合には，SQL のコレクションにマップする．コレクションには，XML DB 名前空間を利用した次のいずれかの方法を指定することが可能となっている．

・可変配列を LOB に格納（デフォルト）

・可変配列を OCT（Ordered Collections in Tables）に格納

▶XML スキーマの schema ルート要素に XML DB 名前空間の storeVarrayAsTable 属性を true に設定する

・ネストした表に格納

▶対象となる複合型の XML DB 名前空間の SQLInline 属性を false に設定する

次に，リスト1の Entry 複合型を OCT に格納する場合の変更箇所および変更した際にリスト2以外に実行される SQL 文を挙げる．

```
<xsd:schema
    targetNamespace="http://xmlns.oracle.co.jp/dayentry"
    xmlns="http://xmlns.oracle.co.jp/dayentry"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    xmlns:xdb="http://xmlns.oracle.com/xdb"
    xdb:storeVarrayAsTable="true">
```

```
CREATE TABLE "SYS_NTFXeVR3axTPCE4ZhjY7vSuA=="
OF "Entry183_COLL"
  (PRIMARY KEY (NESTED_TABLE_ID, ARRAY_INDEX))
  ORGANIZATION INDEX OVERFLOW;
```

Entry 183_COLL 可変配列のコレクションをすべて

リスト3　表の作成

```
CREATE TABLE xmltable of XMLType
  XMLSCHEMA "http://xmlns.oracle.co.jp/dayentry.xsd"
  ELEMENT    "DayEntry";
```

ネストした表“SYS_NTFXeVR3axTPCE4ZhjY7vSuA==”として格納する．OCTを利用することで，多くの可変配列を一つのCLOBに格納した場合のパフォーマンスへの影響を改善することが可能となる．

このように，Oracleデータベースが豊富なオブジェクト対応データベース機能を持つことにより，XML Schemaという複雑な構造をも表現可能なスキーマ言語へと対応している．

5.5　表の作成とXML文書の格納

構造化マッピングによってXML文書を格納するためのXMLTypeデータ型で必要なメタデータが作成されました．Oracle XML DBに登録したXMLスキーマに従ったXML文書を格納するには，まず，オプションに使用するXMLスキーマおよびルート要素を指定したCREATE TABLE文で表を作成する（リスト3）．

次のXML文書をリスト3で作成した表に格納すると，格納された際の構造は表3のようになる．

```
<?xml version="1.0" encoding="Shift_JIS"?>
<DayEntry
  xmlns="http://xmlns.oracle.co.jp/dayentry"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.oracle.co.jp/dayentry
                      http://xmlns.oracle.co.jp/dayentry.xsd">
 <Date>2005-03-05</Date>
 <Entry>
  <EntryId>1001</EntryId>
  <Created>2005-03-05T08:02:17</Created>
  <Title>XMLベースの基幹業務アプリケーション</Title>
  <Body>Temenos社による銀行業務アプリケーションについての資料です。</Body>
 </Entry>
 <Entry>
  <EntryId>1002</EntryId>
  <Created>2005-03-05T13:01:23</Created>
  <Title>はじめての統合アプリケーション・サーバー</Title>
  <Body>初心者向け講座の連載を掲載しました。</Body>
 </Entry>
</DayEntry>
```

表3　xmltable表

SYS_XDBPD$	Date	Entry			
…	2005/03/05	SYS_XDBPD$	EntryId	Title	Body
		…	1001	XML…	…
		…	1002	はじめ…	…
…	…	SYS_XDBPD$	EntryId	Title	Body
		…	…	…	…

5.6　DOM再現性

一般的に，XML文書をリレーショナル・データベースにマッピングすると，元の構造を必要な箇所だけ抜き出して分解するために，XML文書の再現性を保つことが難しくなる．

Oracle XML DBでは，DOM再現性を提供するためにメタデータをシステムで定義したバイナリ属性SYS_XDBPD$を使用してSQLオブジェクト型レベルで保持する．この属性を位置指定記述子（Positional Descriptor，以降PD属性）という．PD属性は，あくまでもOracleデータベース内部で使用される属性で，ユーザがこの列に対する直接の操作はできない．

他の属性に格納できない情報のすべてがこのPD属性によって格納され，Oracle XML DBに格納されたすべてのXML文書のDOM再現性が保証される．この属性によって格納される情報には，次のものがある．

- コメント
- 処理命令（Processing Instructions）
- 名前空間の接頭辞および位置
- 空ノード
- デフォルト値の存在
- substitution関連情報
- 要素の順序
- XMLSchema-instance名前空間
  例）xsi：nilやxsi：Typeなど
- 混合内容（Mixed content）

6.　検索

XML Schemaに基づいたXML文書を，SQLオブジェクト型にマッピングし格納する大きな理由が，検索（および更新）の際のパフォーマンス向上を狙ったものである．

検索は，Oracle XML DBが用意したXPath式を

利用した各種関数を利用することになる．例えば，ExtractValue関数はXML文書のスカラ値をXPath式で検索するための関数である．これらの関数を利用した検索時に，Oracle XML DBは実際に実行するSQL文をより高速に処理できるように変更するクエリ・リライト機能を利用し，検索対象のSQLオブジェクト型のみにアクセスする．

例えば，Date要素の値が"2005/03/05"であるXML文書を検索するための

```
SELECT * FROM xmltable x
WHERE ExtractValue(value(x),
  '/DayEntry/Date') = '2005/03/05';
```

というSQL文は，次のSQL文に変更される．

```
SELECT * FROM xmltable x
WHERE x."Date" = '2005/03/05';
```

さらに高速に処理を行うためには，その用途に応じて索引（Bツリー索引，ファンクション索引，全文検索索引）を使い分けて使用することになる[4]．

7. 課題

最近では，多くの業界が統一スキーマ定義のために，スキーマ言語にXML Schemaを選択している．しかし，いくつかの統一スキーマではこれまでの経緯からDTDを利用している．これらは「2. 統一スキーマとXML Schema」で示した理由からXML Schemaへの早い段階での移行が望まれる．

また，XML Schemaはあらゆる構造を定義できる反面，その仕様を理解することが難しく，統一スキーマで誤った解釈を行った定義を行う場合がある．XML Schemaの仕様を正しく理解した上での実装の必要性と，簡単に利用でき，仕様を正しく実装したリファレンスと言えるGUIツールの誕生が望まれる．さらにXML Schemaをどのように利用すれば，有効な統一スキーマを作成できるか，ベストプラクティスの蓄積が重要である．

Oracle XML DBには，近くW3Cで勧告となると思われるXML Query[5]へのネイティブ対応が望まれる．XML Queryに対応した機能をOracle XML DB自身が提供することにより，高い信頼性とパフォーマンスを持つXML Queryに対応したシステムの構築が可能となる．

8. おわりに

Oracle XML DBによって，XML文書に対してOracleデータベースのSQLオブジェクト型の機能でXML Schemaのデータモデルを実現し，Oracleデータベースがリレーショナル・データベースで培った高い信頼性とパフォーマンスを損なわないまま，ユーザに利便性を提供している．それにより，統一スキーマの実現が加速することは言うまでもない．

参考文献

[1] The W3C Document Type Definition, http://www.w3.org/TR/REC-xml/

[2] The W3C XML Schema Standard, http://www.w3.org/XML/Schema

[3] XML Schema Part 2: Datatypes, http://www.w3.org/TR/xmlschema-2/

[4] 日本オラクル株式会社 (2004)『Oracle XML DB開発者ガイド10gリリース1 (10.1)』, http://otn.oracle.co.jp/document/

[5] The W3C XML Query, http://www.w3.org/XML/Query