IBM社のリレーショナル・データベース DB2におけるXMLデータの実装方法

菅原　香代子

IBM社のリレーショナル・データベースであるDB2 UDB（Universal Database）の現時点での最新バージョンであるV8.2におけるXMLデータの実装方法について解説する．DB2ではXMLデータをとりあつかうためにXML Extenderというフィーチャを無料で提供している．XML Extenderを使ってXMLデータとRDB表をマッピングする二つの方法，XML COLUMNとXML COLLECTIONについてその特徴を解説する．

キーワード：DB2，XML Extender，XML COLUMN，XML COLLECTION

1. はじめに

XMLデータは2000年前後に新しい技術として非常に脚光を浴びたことがある．ただし，ある意味ではその当時はまだ学問的な興味が半分という感じであった．またその当時すでにXMLデータのツリー構造を保管するデータベースとしてXML専用データベース製品も数社から市販されていたが，やはりまだまだ一般のユーザは試験的に使ってみるというのが大多数であったように思う．しかし，筆者自身最近になって，ようやくXMLデータが本番業務に実際に使用されつつあるのではないかと感じている．実際昨年来多くのお客様からXMLデータベースについての問い合わせが増えているのが事実である．例えば，XMLデータを使った業務として，新聞業界におけるNewsML，医薬業界におけるMedXML，旅行業界におけるTravelXML，金融分野におけるXBRLなど，XMLデータを使ったシステム構築がいよいよ始動しつつあるようである．MXLデータを扱うシステムを構築する際に，キーとなるファクタは多々あるが，やはりその中でも特に重要なのがデータストアの要素であるということに異論を唱える技術者はいないだろう．そこで，XMLデータをどのようにデータベースに保管および効率よく取り出すかという技術の重要性がますます増しているように思われる．

また最近はXMLデータはSOA（Service Oriented Architecture）という面でも非常にその存在が重要になりつつある．SOAというのは，ビジネス・プロセスをサービスという要素にわけ，そのサービス群をビジネス要件の変更にあわせて柔軟にダイナミックに組み合わせて，システムをすばやく構築するという考え方であるが，そのようなプロセス統合，アプリケーション統合を考えるときに忘れがちではあるが，システム間にまたがるデータをどのように処理/管理するのかということが重要である．また，そのようなデータの重複やどちらが最新であるかというようなこと，つまりメタデータまたはメタメタデータの（データ自身のビジネスおよびIT的な意味）管理も今脚光を浴びつつあるといえる．当然そのようなメタデータ管理はXMLで，と考えるのが自然である．

XMLデータはもともとトランザクション・データというよりは，ドキュメントという性質から出発したという経緯があり，そのようなコンテンツ管理業務向けに，IBM社はContents Managerという製品を市販している．また，XMLデータを含むいろいろなデータ・フォーマットの情報統合という点から，WebSphere Information Integratorという製品も提供しているが，本論文では紙面の都合上，トランザクション・データとしてのXMLデータの処理方法を考えるという点からリレーショナル・データベース製品であるDB2でのXML実装機能を解説する．

2. DB2でのXMLデータの取り扱い

2.1 DB2とは？

DB2は，IBM社が1982年にIBM社のメインフレーム上で動くリレーショナル・データベース管理製品として発表したのがその始まりである．その後1993

すがはら　かよこ
日本アイ・ビー・エム㈱
〒150-0043 渋谷区道玄坂1-12-1

年に，UNIX/Windowsのオープン・プラットフォーム上でも稼動する製品も発表され，同様にOS/400上で稼動するDB2/400とあわせて，DB2ファミリ製品として現在にいたっている．現在の最新バージョンはV8であるが，本論文ではその製品群の中でも，UNIX/Linux/Windows上で動くDB2の機能を解説する．DB2はいうまでもなく，リレーショナル・データベースであるため，すべてのデータを表形式で表現する．そのデータ値としてDB2がサポートしているデータ型は，文字型のCHAR，VARCHAR，Graphic，数値型のINTEGER，BIGINT，FLOAT，DBCIMAL，日付け型のDATE，TIME，TIMESTAMP，その他に，マルチメディア・データや大きなテキストなどを保管するためのBLOB，CLOB，DECLOB等がある．残念ながら現時点ではDB2はXMLのツリー構造をそのままの形で保管する，いわゆるXMLネイティブ・データ型というものは提供していない．その代わり，XMLデータとリレーショナル表の間の橋渡しをする機能を提供している．それがXML Extenderと呼ばれるフィーチャである．

2.2 XML Extender

XML Extenderは，図1のようにユーザ定義データ型（UDT）とユーザ定義関数（UDF）とストアードプロシージャから構成されている．ユーザ定義データ型というのは，DB2が提供するデータ型を元にして特別な性質を持つデータ型をユーザ自身が作成できるというものである．そこでその機能を使ってXML ExtenderではCLOB型をもとにしたXMLCLOB，VARCHAR型を元にしたXMLFile，XMLVarcharを提供してこのデータ型にXMLドキュメントを保管できるようにしている[1]．また，ユーザ定義関数というのは，DB2が製品として提供しているSQL関数に付け加えてユーザが独自にプログラムを作成して，独自の関数を作ることができる機能であるが，XMLエクステンダではXPathを使うようなXMLデータ特有の処理を行うために，保管/検索/更新/変換/検証のための特殊な関数を提供している．

その他にXMLドキュメントをELEMENT/ATTRIBUTEの単位の要素に分解して，RDB表の複数の列に保管するためのストアードプロシージャも提供している．その反対にRDB表の複数の列から

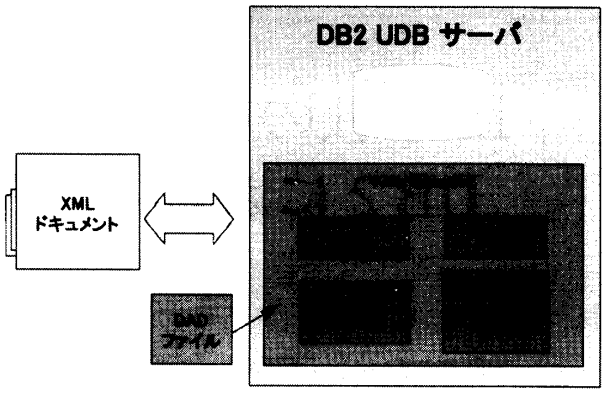

図1 XML Extenderのアーキテクチャ

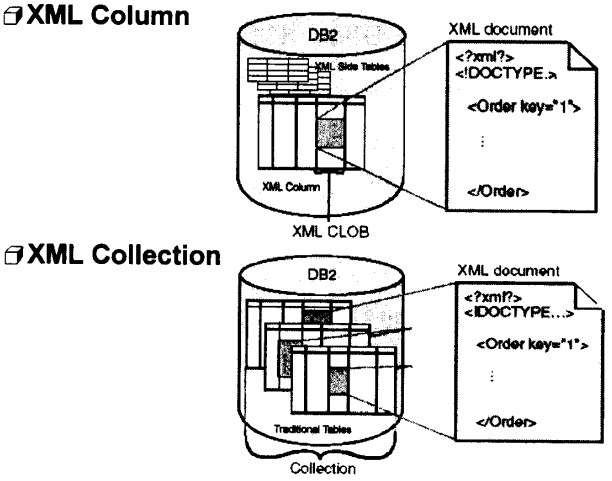

図2 XML Extenderの二つのアクセスとストレージ方法

XMLドキュメントを構築するストアードプロシージャも提供している．この際のRDB表の列は特別なUDTではなくて，DB2が提供しているCHAR型やINT型のような標準型である．このようにXMLドキュメントをRDB表に列に分解/構築するような方式を一般にShredding方式と呼ぶ．XML Extenderでは，ユーザはXMLのツリー構造とRDB表をマッピングするためにDAD（Data Access Definition）と呼ばれるXMLファイルをあらかじめ用意しておく必要がある．DADについては後で詳しく解説する．

さて，実際のXMLデータの格納方法であるが，XML Extenderでは，XMLデータをRDB表のデータ・ストレージに保管するためにXML COLUMNとXML COLLECTIONと呼ばれる図2のような二つのアクセス/ストレージ方法がある．次にその二つの使用方法，目的，およびそれらの違いについて詳しく述べることにする．

[1] CLOB型とはバイナリ・データを保管するBLOB型にテキストを保管するためのコード・ページの情報を持つデータ型のことである．

3. XML Extenderの二つの方法

3.1 XML COLUMNの実装方法

XML COLUMNとは，XMLドキュメントを丸ごと，RDB表の一つの列に保管して，またそこからXML Extenderが提供するSQL関数を使って取り出す機能のことである．その時保管するために使われるデータ型は前述したXMLCLOB，XMLFile，XMLVarcharである．いったんRDB表に保管された後は，extractInteger()やextractVarchar()などのUDFを使って条件式でXPATHにより，指定したELEMENTやATTRIBUTEの値を簡単に取り出すことができる．またXSLTransformTofile()関数を使って指定したスタイルシートでXMLドキュメントを変換したりということも可能である．ただし，この方式では，そのまま何も工夫しないでRDB表の列に保管するのであれば，後にXPATHで任意のATTRIBUTE/ELLEMENTを条件とした検索では，高いパフォーマンスが期待できない．そこでXML Extenderでは図3のDADと呼ばれるXMLファイルで，あらかじめ後の検索で高いパフォーマンスが要求されると予想されるELEMENT/ATTRIBUTEを指定することにより，サイド表と呼ばれる一種のINDEXを自動的に作成することができる（図3，図4）．これによって，XMLドキュメントの検索や更新で高いパフォーマンスを保障している．

XML COLUMN方式は，XMLドキュメントをそのままの形で保管したい場合に適している．XML列にドキュメントを保管した後，これをXPATHによる条件で取り出すことがSQL文で簡単にできるという利点があるが，ただし，あらかじめ検索の条件式でよく使用されるELEMENT/ATTRIBUTEをサイド表に作成しておくことがパフォーマンスの観点から必要なので，いろいろなXPATHを指定して，随意にデータの検索を行ったり，またXMLドキュメントを部分的に更新したりという場合には向いていないといえる．しかしながら，DB2で提供するNet Search Extenderというフィーチャと組み合わせて使うことにより，より高度な全文検索（Nグラム検索のような）をすることも可能である．

3.2 XML COLLECTIONの実装方法

XML COLUMNがXMLドキュメントをXML形式のまま，RDB表の一つの列に保管/取り出しを行うのに対して，XML COLLECTIONはXMLドキュメ

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:¥dxx¥samples¥dtd¥dad.dtd">
<DAD>
<dtdid>c:¥dxx¥samples¥dtd¥getstart.dtd</dtdid>
<validation>YES</validation><Xcolumn>

<Xcolumn>
   <table name="order_side_tab">
      <column name="order_key"
         type="integer"
         path="/Order/@key"
         multi_occurrence="NO"/>
      <column name="customer"
         type="varchar(50)"
         path="/Order/Customer/Name"
         multi_occurrence="NO"/>
   </table>
   <table name="part_side_tab">
      <column name="price"
         type="decimal(10,2)"
         path="/Order/Part/ExtendedPrice"
         multi_occurrence="YES"/>
   </table>
   <table name="ship_side_tab">
      <column name="date"
         type="DATE"
         path="/Order/Part/Shipment/ShipDate"
         multi_occurrence="YES"/>
   </table>

</Xcolumn>
</DAD>
```

図3　XML COLUMNのDADの例

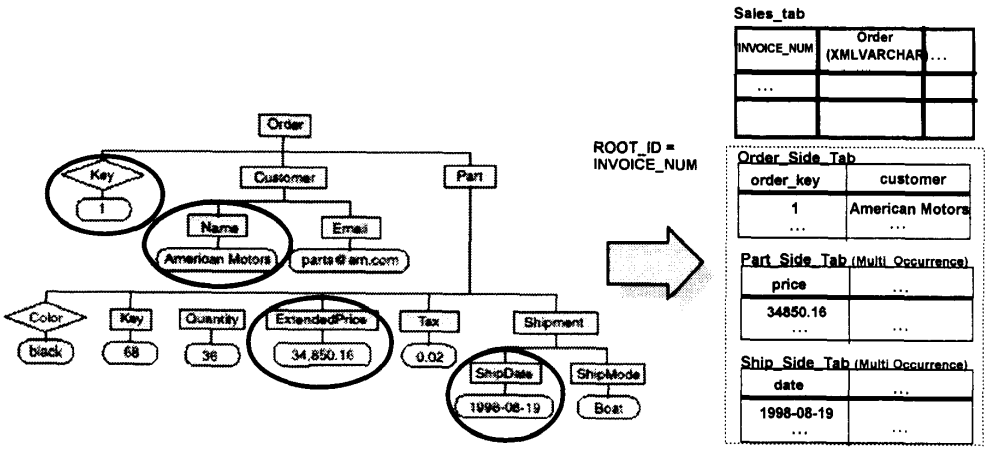

図4　XML COLUM—サイド表

Order_tab

Order_Key	Customer_name	Customer_email	Customer_phone
1	American Motors	parts@am.com	800-AM-PARTS

Part_tab

PART_KEY	COLOR	QUANTITY	PRICE	TAX	ORDER_KEY
156	red	17	17954.55	2.000000e-2	1
68	black	36	34850.16	6.000000e-2	1
128	red	28	38000.00	7.000000e-2	1

Ship_tab

DATE	MODE	COMMENT	PART_KEY
1998-03-13	TRUCK	This is the first shipment to service of AM	156
1999-01-16	FEDEX	This is the second shipment to service to AM	156
1998-08-19	BOAT	This shipment is requested by a call from AM marketing	68
1998-08-19	AIR	This shipment is ordered by an email	68
1998-12-30	TRUCK		128

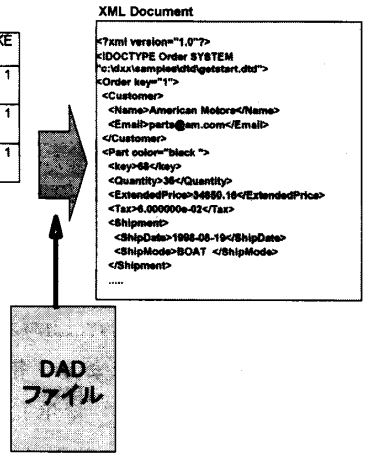

図5　XML COLLECTION

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
<validation>NO</validation>
<Xcollection>
<SQL_stmt>select book_id, price_date, price_text
from book_table ORDER BY book_id</SQL_stmt>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE book SYSTEM "c:\dtd\book.dtd"</doctype>
<root_node>
<element_node name="book">
 <attribute_node name="id">
  <column name="book_id"/>
 </attribute_node>
 <element_node name="price">
  <attribute_node name="date">
   <column name="price_date"/>
  </attribute_node>
  <text_node><column name="price_text"/></text_node>
 </element_node>
</element_node>
</root_node>
</Xcollection>
</DAD>
```

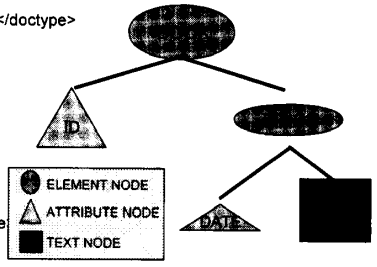

図6　XML COLLECTION-SQL Node Mapping の DAD の例

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
<dtdid>c:\dtd\book.dtd</dtdid>
<validation>YES</validation>
<Xcollection>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE book SYSTEM "c:\dtd\book.dtd"</doctype>
<root_node>
<element_node name="book">
 <RDB_node>
  <table name="book_table" key="book_id"/>
 </RDB_node>
  <attribute_node name="id">
  <RDB_node>
   <table name="book_table"/>
   <column name="book_id" type="integer"/>
  </RDB_node>
 </attribute_node>
 <element_node name="price">
  <attribute_node name="date">
   <RDB_node>
    <table name="book_table"/>
    <column name="price_date" type="date"/>
   </RDB_node>
  </attribute_node>
<text_node>
    <RDB_node>
     <table name="book_table"/>
     <column name="price_text" type="decimal(10,2)"/>
    </RDB_node>
  </text_node>
 </element_node>
</element_node>
</root_node>
</Xcollection>
</DAD>
```

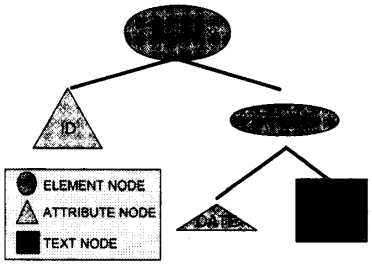

図7　XML COLLECTION-RDB Node Mapping の DAD の例

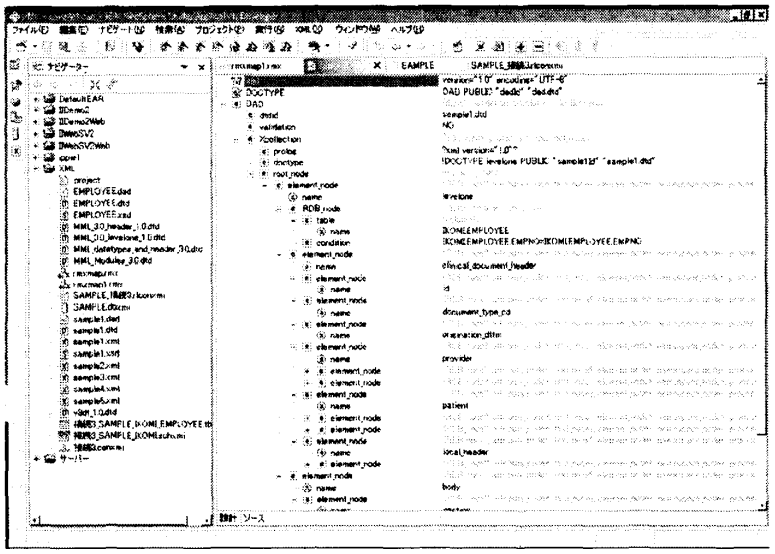

図8　WebSphere Studio Application Developer

ントをELEMENT/ATTRIBUTEごとにばらばらにして，一つ以上の表の複数の列に保管する方法である．または反対にすでに存在しているRDB表の複数の列からXMLドキュメントを構成するということもできる．XML COLUMNと違って，RDB表には特別なUDTを必要としない．このXMLドキュメントの分解/構築はXML Extenderが提供するストアード・プロシージャを使って行われる．その際にELEMENT/ATTRIBUTEと表，列とのマッピングの定義を行うのが今度もDADである．ただし，前述のXMLCOLUMNのDADとは少々構文が違っている．図5はRDB表の列からXMLドキュメントを構築する例である．

XML COLLECTIONのDADには実は2種類ある．SQLステートメントでXMLのフォーマットを定義する方法（SQL Node Mapping）と，直接RDB表の列とXMLのELEMENT/ATTRIBUTEを対応づけてマップする方法（RDB Node Mapping）である．二つの例を次に示す（図6，図7）．

XML COLLECTIONはすでにRDB表が存在し，そのデータからXMLドキュメントをDTD/XML Schemaに基づいて作成したい場合や，XMLドキュメントの中の一部分が頻繁に更新され，そのパフォーマンスが重要である場合に適している．つまり，この方式では，データはRDB表に保管されるので，いつの時点でも任意の列に自由に索引を作成することが可能であるし，更新/削除といった処理が早いという利点がある．ただし，XMLドキュメントの分解および再構築の際のパフォーマンスのオーバヘッドは考慮しなければならない点である．また，これはリレーショナル・データベースの特徴であるが，RDB表では重複した値を許さないという正規化という基本概念がある．これにより，一つのXMLドキュメントに同じELEMENTが繰り返しあらわれるマルチオカレンスの場合，RDB表にマッピングするためには，複数の表間の参照整合性などを考慮する必要があるし，可逆性という問題も，少々複雑であるといえる．またXMLスキーマが変更になった場合，表の再設計が必要になる場合も考慮しなければならない．

3.3　DAD表の作成

前述のDADはText Editorなどを使って作成すると複雑そうにみえるが，開発GUIツールであるIBM社のWebSphere Studio Application Developer（現Rational Application Developer）を使うとマウス操作で簡単にDADを作成できる（図8）．これを使うと，DB2にあまり詳しくないユーザでも，DB2の表からXML ELEMENT/ATTRIBUTEへマッピングするDADファイルが簡単に作成できる．

3.4　DTDとXML Schema

XML ExtenderにはXML文書の妥当性検査を行うためのリポジトリを提供している．それにあらかじめDTD/XML　Schemaを登録しておくことによりXMLドキュメントの保管/取り出しの際に自動的に妥当性検査を行うことが可能となる．

4.　SQL/XML

SQL/XMLはISO国際標準ISO/IECで，IBMを含めたSQLX Informal Group of Companiesで開発

(実行例)
SAMPLEデータベース
のDEPARTMENT表を
XML化して返す

DEPTNO	MGRNO	ADMRDEPT
.....	.....	.....
D01	(NULL)	A00
.....	.....	.....

SQL文

```
SELECT REC2XML(1.0,'COLATTVAL','row',deptno,mgrno,admrdept)
            FROM department WHERE deptno = 'D01'
```

実行結果 (1行分:実際の結果には改行はありません)

```
<row>
  <column name="DEPTNO">D01</column>
  <column name="MGRNO" null="true"/>
  <column name="ADMRDEPT">A00</column>
</row>
```

図9　REC2XML()関数

- 使用例

```
SELECT XML2CLOB(
   XMLELEMENT(NAME "FULLNAME",
      firstnme || '.' || midinit || '.' || lastname)) from employee
```

- 実行結果 (この1行が結果セットの1行分となる)

```
<FULLNAME>CHRISTINE.I.HAAS</FULLNAME>
<FULLNAME>MICHAEL.L.THOMPSON</FULLNAME>
<FULLNAME>SALLY.A.KWAN</FULLNAME>
.....
```

図10　XMLELEMENT()関数

されたXMLドキュメント作成・変換のためのUDF群のことである．XML Extender以外にも，DB2では，これらの関数を提供している．例えば図9のREC2XML()関数を使うとRDB表に保管されているデータを通常のSQLステートメントで簡単にXML化することができる．

また，図10のXMLELEMNT()関数を使うことによりタグを生成することも可能である．

5. まとめ

DB2では本論文で説明してきたように，XMLデータをRDB表のデータにマッピングして高速に保管/取り出しをするためにXML Extenderという非常に魅力的な機能を提供している．このXML Extenderを使用することにより，ほとんどのXMLアプリケーションの要求は満たせるはずである．ただし，XMLのツリー構造を100%利用できているかという観点ではまだまだ満点でないのも事実である．そこで，DB2の更なるXML機能拡張はこれからも続いていく予定である．将来的には，リレーショナル・データベースとXMLのさらなる融合も見据えている．その段階では，既存のSQLアプリケーション開発者も，新しいXMLアプリケーション開発者もなんら意識することなく自由に簡単に高速にデータのやりとりを行うことができるだろう．そのような“進化”したDB2を早く使ってみたいと，筆者もDB2を愛するエンジニアの一人として切に望むものである．