

DB2 XMLエクステンダー

第4話

第1話

- XMLエクステンダーとは
- XML文書の保管方法
- DTDの保管
- 製品構成

第2話

- XMLエクステンダー使用のための準備
- XML列の使用例

第3話

- XMLコレクションの概要
- RDB_nodeマッピングによるXMLの分解

第4話

- XML文書合成の例
- RDB_nodeマッピングによる合成
- SQLマッピングによる合成

はじめに

- ▶ いよいよ、このXMLエクステンダーについてのシリーズも最終回になりました。最終回の今回は、前回に引き続き、XMLコレクションについて見ていきます。前回は、XML文書を要素ごとに分解して保管する様子を見ましたので、今回は、分解してリレーショナル表に保管されたデータから、XML文書を合成して取り出すところについて見ていきましょう。
- ▶ 本文中 (Install_Dir) と記述している部分はXMLエクステンダーを導入したディレクトリを示しています。デフォルトでは
 - Windowsの場合: C:¥dxx
 - AIXの場合: /usr/lpp/db2xml_07_01
 - Sunの場合: /opt/IBMdb2xml/V7.1
 - Linuxの場合: /usr/IBMdb2xml/v7.1になります。

XML文書の合成の例

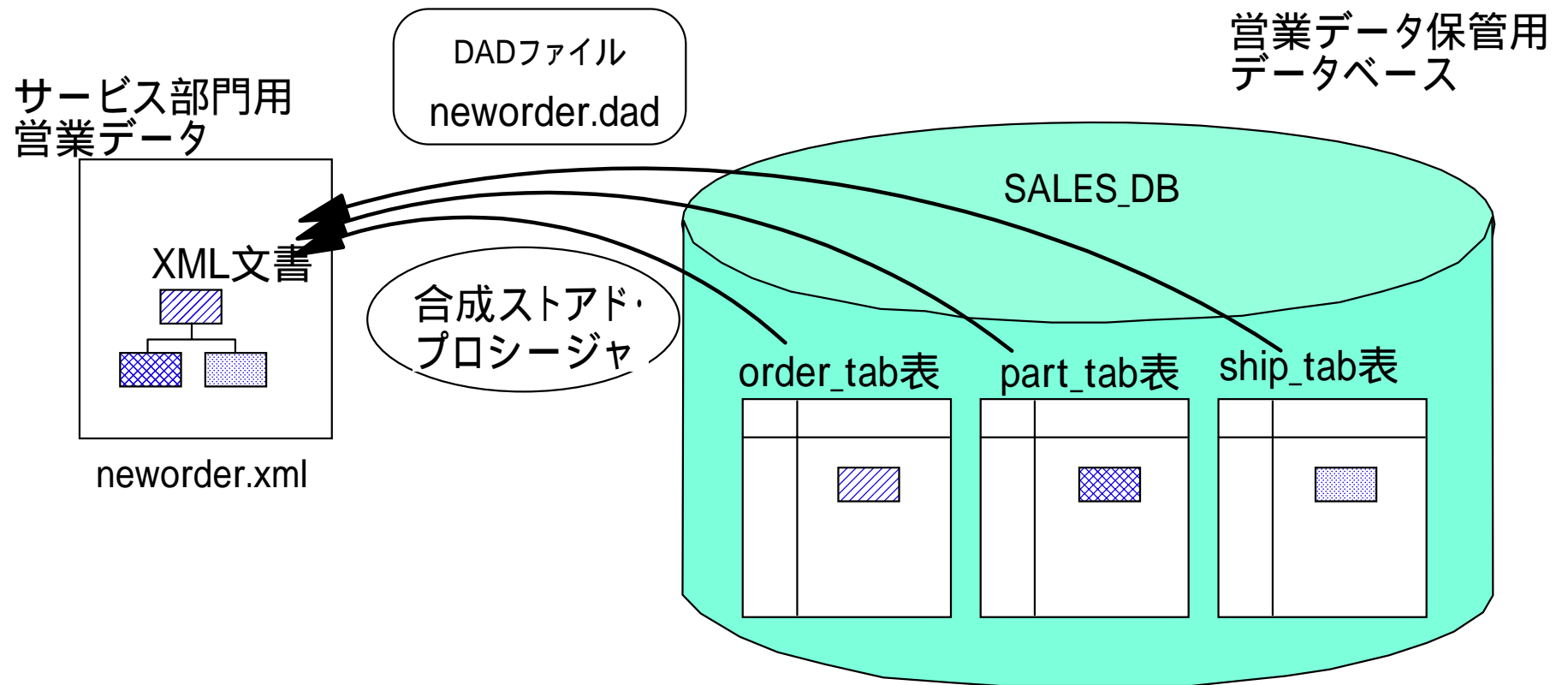


図1 . XML文書の合成の例

解説:XML文書合成の例

- ▶ ここでは後述の「コレクション名を指定するサンプル(RDB_nodeマッピング)」の例を挙げて説明します。
- ▶ XML文書の要素/属性の値を、XMLコレクションを使用してorder_tab表、part_tab表、ship_tab表に保管されている値を基に合成します。
- ▶ 値に条件を設定し、その条件を満たす値のみを使用してXML文書を合成することも可能です。
- ▶ 各要素/属性とその値が保管される表・列のマッピング情報はDADファイル内で定義されます。
- ▶ 合成の方式には以下の2種類あります。それぞれの内容については、後述の「XML文書合成の方式」で触れます。
 - RDB_node マッピング方式
 - SQL マッピング方式

使用するリレーショナル表

- リレーショナル表の内容は以下のとおりです。これらの表はenable_collection コマンドを実行する時にDADファイルの定義に従って、XMLエクステンダーより自動的に作成されますが、すでに同じ名前の表が存在する場合はその表が使用されます。この場合は、DADファイル内の列数、データタイプなどの定義と実際の表の列数、データタイプなどが一致している必要があります。

order_tab表

ORDER_KEY	CUSTOMER_NAME	CUSTOMER_EMAIL
INTEGER	VARCHAR(16)	VARCHAR(16)
1	American Motors	parts@am.com

part_tab表

COLOR	PART_KEY	PRICE	TAX	QTY	O_KEY
CHAR(6)	INTEGER	DECIMAL(10,2)	REAL	INTEGER	INTEGER
red	156	1795.40	+2.00000E-002	17	1
black	68	3485.16	+6.00000E-002	36	1
red	128	3800.00	+7.00000E-002	28	1

ship_tab表

DATE	MODE	COMMENT	P_KEY
DATE	CHAR(6)	VARCHAR(250)	INTEGER
1998-03-13	TRUCK	This is the first shipment to service of AM.	156
1999-01-16	FEDEX	This the second shipment to service of AM.	156
1998-08-19	BOAT	This shipment is requested by a call. from AM marketing.	68
1998-07-23	AIR	This shipment is ordered by an email.	68
1998-12-30	TRUCK	-	128

XML文書の構造と表の関係

- ▶ リレーショナル表の列とXML文書とのマッピングはDADファイル(neworder.dad)の中で下記のように定義されています。
- ▶ <ExtendedPrice>要素の値は、part_tab表のPRICE列の値の中で「2500.00より大きい値のみ」という条件が、<ShipDate>要素の値はship_tab表のDATE列の値の中で、「1966-01-01より大きい値のみ」という条件が設定されています。

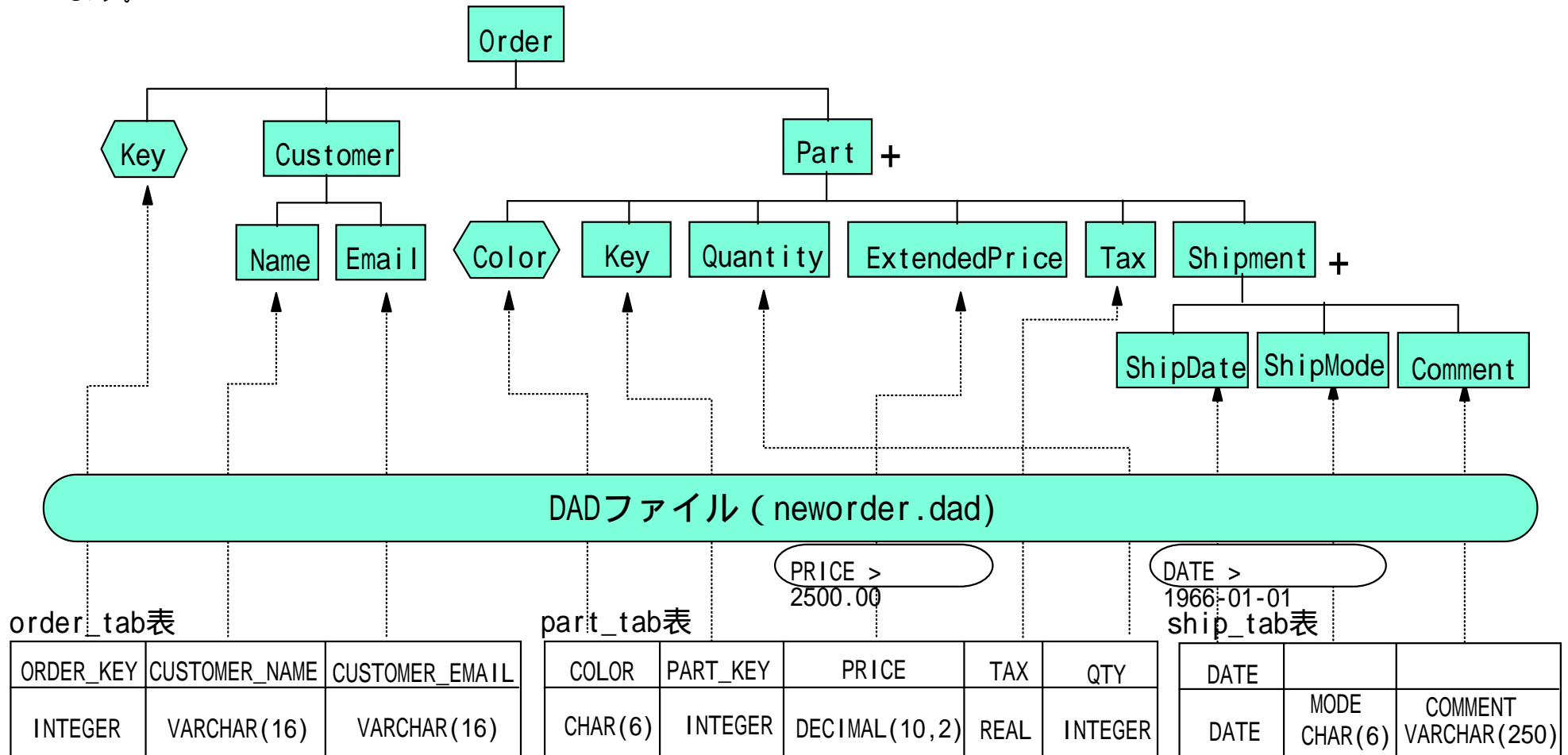


図2 . XML文書の構造と表の関係

合成されるXMLファイル例

- ▶ 合成されるXMLファイルは次のようになります。

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\neworder.dtd">
<Order Key="1">
  <Customer>
    <Name>American Motors</Name>
    <Email>parts@am.com</Email>
  </Customer>
  <Part Color="black ">
    <Key>68</Key>
    <ExtendedPrice>3485.16</ExtendedPrice>
    <Tax>6.000000e-02</Tax>
    <Quantity>36</Quantity>
    <Shipment>
      <ShipDate>1998-07-23</ShipDate>
      <ShipMode>AIR </ShipMode>
      <Comment>This shipment is ordered by an email.</Comment>
    </Shipment>
    <Shipment>
      <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT </ShipMode>
      <Comment>This shipment is requested by a call. from AM marketing.</Comment>
    </Shipment>
  </Part>
  <Part Color="red ">
    <Key>128</Key>
    <ExtendedPrice>3800.00</ExtendedPrice>
    <Tax>7.000000e-02</Tax>
    <Quantity>28</Quantity>
    <Shipment>
      <ShipDate>1998-12-30</ShipDate>
      <ShipMode>TRUCK </ShipMode>
      <Comment></Comment>
    </Shipment>
  </Part>
</Order>
```

XML文書合成の方式

- ▶ XML 文書の合成の方式には、表の列とXML文書の要素/属性のマッピング方式の種類により2種類あります。
- ▶ 一つはRDB_node マッピングと呼ばれる方式です。
 - この方式では要素を指定する<element_node>タグ、または属性を指定する<attribute_node>タグの中に、<RDB_node>タグを指定し、その中でどの表のどの列をマッピングするかを定義します。
 - RDB_node マッピングを定義したDADファイルは、XML文書の合成・分解のどちらにも使用することができます。つまり、XML文書の分解に使用したDADファイルを使用して、もとのXML文書を合成することが可能です。
- ▶ もう一つはSQL マッピングと呼ばれる方式です。
 - この方式は、単一のSQLのSELECT文でXML要素/属性として使用する列の値を取り出し、その各値について、どの要素/属性とマッピングするかを指定する方式です。
- ▶ これらのマッピング情報はDADファイル内に指定します。

- ▶ さらに、これらの各方式について、それぞれ2通りの方式が用意されており、それぞれ異なるストアド・プロシージャを使用します。
- ▶ 一つはコレクション名を指定する方式です。
 - この方式では、enable_collection コマンドによりDADファイルに対応するコレクション名を登録する必要があります。
 - コレクション名を登録するとDADファイルがXML Extender の管理表内に保管されるため、XML文書を合成する際にDADファイルを指定しなくても、最初に指定したXML コレクションの名前を指定するだけで済みます。
 - dxxRetrieveXML()ストアド・プロシージャを使用します。
- ▶ もう一つは、XML文書を合成する際に、DADファイルの内容を指定する方式です。
 - enable_collection コマンドを実行する必要はなく、XML文書の合成時には、コレクション名のかわりにDADファイルの内容を直接指定してストアド・プロシージャを実行します。
 - 合成する条件が頻繁に変更される場合などには、上記の方式よりオーバーヘッドが少ないことがあります。
 - dxxGenXML()ストアド・プロシージャを使用します。

RDB_nodeマッピングによる合成(コレクション名を指定する方法)

- ▶ コレクション名を指定するサンプルは、retrvxml.cmd サンプルファイルに記述されています。このサンプルはRDB_nodeマッピングのサンプルです。
- ▶ サンプルファイルを実行する前に、第3話の「データベースの準備」のステップを実行して、データベースの作成、XML使用可能化を完了していることを確認します。ここでも、第3話と同様、データベースにMYDBが使用されます。
- ▶ サンプルファイル内では、以下のステップが実行されます。
 - a. 合成するXMLファイルの妥当性検査を行う場合(DADファイル内でvalidationをYESに指定した場合は、使用するDTDファイル(neworder.dtd)をDTD_REF表に挿入します。
 - b. 合成されたXML文書を保管するための表 result_tab表を作成します。ここで使用するXMLエクステンダーの合成ストアド・プロシージャ(dxxRetrieveXML)は合成したXML文書を表に保管します。
 - c. enable_collection コマンドをDADファイルを指定して実行し、XMLコレクションに名前をつけて登録します。ここでは、neworder.dadファイルを指定し、XMLコレクションに「abc」という名前をつけて登録しています。このサンプルでは、このステップで合成する元データを保管する表がXMLエクステンダーにより自動的に作成されます。DADファイル内で指定した表の定義と同じ名前の表が既に存在する場合には、その表が使用されます。ただし、列数、データタイプなどはDADファイル内の定義と実際の表の列定義との間で一致している必要があります。続いて、作成された表にデータを挿入します。
 - d. dxxRetrieveXML ストアド・プロシージャを使用してXML文書の合成を行います。
 - *実際には、内部でdxxRetrieveXMLストアド・プロシージャを使用するretrieveというプログラムを使用して、MYDBからのXML文書の合成を行っています。retrieveプログラムの内容は以下のファイルで確認できます。
 - Windowsの場合: (Install_Dir)¥samples¥c¥retrieve.sqx
 - Unixの場合: (Install_Dir)/samples/c/retrieve.sqx
 - e. 合成したデータを確認するために、SELECT文によりresult_tab表の内容を確認します。
 - f. 作成した表をドロップし、DTDファイルのエントリを削除して、作成した環境を削除します。
 - g. disable_collection コマンドにより、XMLコレクションの登録を削除します。
- ▶ ここで使用するDADファイルは以下のファイルになります。
 - ▶ Windowsの場合: (Install_Dir)¥samples¥dad¥neworder.dad
 - ▶ UNIXの場合: (Install_Dir)/samples/dad/neworder.dad
- ▶ なお、サンプルファイルはありませんが、コレクション名を指定する方式で、SQL マッピング方式も可能です。

RDB_nodeマッピング(コレクション名を指定) - サンプル・スクリプト

▶ retrvtxml.cmd ファイルの内容 (Windows版)

```
db2 connect to mydb

db2 "insert into db2xml.dtd_ref values('neworder.dtd',
db2xml.XMLClobFromFile('%DB2DXXPATH%\samples\dtd\neworder.dtd'), 0, 'anita', 'anita','anita')"
```

a

```
db2 "echo ----- Creating result_tab -----"
db2 "create table result_tab(doc db2xml.XMLVarchar)"
```

b

```
rem create collection "abc" with validation check
dxxadm enable_collection mydb abc %DB2DXXPATH%\samples\dad\neworder.dad

rem insert data to tables
db2 "insert into order_tab values(1, 'American Motors', 'parts@am.com')"
```

c

```
db2 "insert into part_tab values('red', 156, 1795.4, 0.02, 17, 1)"

(中略)
retrieve mydb abc result_tab
```

d

```
db2 "echo ----- Displaying the resulting XML documents -----"
db2 "select * from result_tab"
```

e

```
rem cleanup
db2 drop table order_tab
db2 drop table part_tab
db2 drop table ship_tabf
db2 drop table result_tab
```

f

```
dxxadm disable_collection mydb abc
db2 "delete from db2xml.dtd_ref where dtdid='neworder.dtd'"

db2 terminate
```

g

RDB_nodeマッピング(コレクション名を指定) - DADファイル

▶ DADファイル(neworder.dad)の内容(抜粋)

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:¥dxx¥dtd¥dad.dtd">
<DAD>
  <dtdid>neworder.dtd</dtdid>
  <validation>YES</validation>
  <Xcollection>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM "c:¥dxx¥samples¥dtd¥neworder.dtd"</doctype>
<root_node>
  <element_node name="Order">
    <RDB_node>
      <table name="order_tab" key="order_key" />
      <table name="part_tab" key="part_key" />
      <table name="ship_tab" key="date" />
      <condition>order_tab.order_key=part_tab.o_key AND part_tab.part_key=ship_tab.p_key</condition>
    </RDB_node>
    <attribute_node name="Key">
      <RDB_node>
        <table name="order_tab" />
        <column name="order_key" type="integer" />
      </RDB_node>
    </attribute_node>
    <element_node name="Customer">
      <element_node name="Name">
        <text_node>
          <RDB_node>
            <table name="order_tab" />
            <column name="customer_name" type="varchar(16)" />
          </RDB_node>
        </text_node>
      </element_node>
    </element_node>
  </root_node>
</DAD>
```

合成する表の定義

Key属性のマッピング定義

Name要素のマッピング定義

RDB_nodeによる合成(DADファイルの内容を指定する方法)

- ▶ RDB_nodeマッピングでDADファイルの内容を指定する方式のサンプルは、genxml_rdb.cmd サンプルファイルに記述されています。
- ▶ サンプルファイルを実行する前に、第3話の「データベースの準備」のステップを実行して、MYDBデータベースの作成、XML使用可能化を完了していることを確認します。ここでも、第3話と同様、データベースにMYDBが使用されません。
- ▶ サンプルファイル内では、以下のステップが実行されます。
 - a. 元データを保管するための表(order_tab表、part_tab表、ship_tab表)、および合成した結果のXML文書を保管するための表(result_tab表)を作成します。表の列名、列の数、データタイプなどはDADファイル内の指定と一致していなくてはなりません。ここで使用するXMLエクステンダーの合成ストアド・プロシージャ(dxxGenXML)は合成したXML文書を表に保管します。
 - b. dxxGenXML ストアド・プロシージャを使用してXML文書の合成を行います。引数に指定するのはDADファイルと合成結果のXML文書を保管する表の名前です。
 - *実際には、内部でdxxGenXMLストアド・プロシージャを使用するtests2xというプログラムを使用して、MYDBからのXML文書の合成を行っています。tests2xプログラムの内容は以下のファイルで確認できます。
 - Windowsの場合： (Install_Dir)¥samples¥c¥tests2x.sqx
 - Unixの場合： (Install_Dir)/samples/c/tests2x.sqx
 - c. 合成したデータを確認するために、SELECT文によりresult_tab表の内容を確認します。
 - d. 作成した表をドロップして、作成した環境を削除します。
- ▶ ここで使用するDADファイルは以下のファイルになります。
 - ▶ Windowsの場合： (Install_Dir)¥samples¥dad¥order_rdb.dad
 - ▶ UNIXの場合： (Install_Dir)/samples/dad/roder_rdb.dad

RDB_nodeマッピング(DADファイルの内容を指定) - サンプル・スクリプト

▶ genxml_rdb.cmd ファイルの内容 (Windows版)

```
db2 "connect to mydb"

db2 "echo ----- Setting up the database mydb -----"

db2 "create table order_tab(order_key varchar(4), customer_name varchar(16), customer_email varchar(16),
customer_phone varchar(16))"
db2 "create table part_tab(part_key integer, color char(6), qty integer, price decimal(10,2), tax real,
order_key varchar(4))"
db2 "create table ship_tab(date date, mode char(6), comment varchar(128), part_key integer)"
db2 "create table result_tab(doc varchar(2000))"

db2 "insert into order_tab values('1', 'American Motors', 'parts@am.com', '800-AM-PARTS')"
db2 "insert into part_tab values(156, 'red', 17, 17954.55, 0.02, '1')"
(中略)

db2 "echo ----- Calling the RDB-node XML retrieval stored procedure -----"
tests2x mydb %DB2DXXPATH%\samples\dad\order_rdb.dad result_tab

db2 "echo ----- Displaying the resulting XML documents -----"
db2 "select * from result_tab"

db2 "echo ----- Cleaning up the database mydb -----"

db2 "connect to mydb"
db2 "drop table order_tab"
db2 "drop table part_tab"
db2 "drop table ship_tab"
db2 "drop table result_tab"

db2 "terminate"
```

a

b

c

d

RDB_nodeマッピング(DADファイルの内容を指定) - DADファイル

▶ DADファイル (order_rdb.dad) の内容 (抜粋)

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:¥dxx¥dtd¥dad.dtd">
<DAD>
  <dtdid>order_rdb.dtd</dtdid>
  <validation>NO</validation>
  <Xcollection>
  <prolog?xml version="1.0"?</prolog>
  <doctype>!DOCTYPE Order SYSTEM "c:¥dxx¥samples¥dtd¥order_rdb.dtd" </doctype>
  <root_node>
  <element_node name="Order">
    <RDB_node>
      <table name="order_tab" />
      <table name="part_tab" />
      <table name="ship_tab" />
      <condition>order_tab.order_key=part_tab.order_key AND part_tab.part_key=ship_tab.part_key </condition>
    </RDB_node>
    <attribute_node name="Key">
      <RDB_node>
        <table name="order_tab" />
        <column name="order_key" />
      </RDB_node>
    </attribute_node>
    <element_node name="Customer">
      <element_node name="Name">
        <text_node>
          <RDB_node>
            <table name="order_tab" />
            <column name="customer_name" />
          </RDB_node>
        </text_node>
      </element_node>
    </element_node>
  </RDB_node>
</element_node>
</RDB_node>
</DAD>
```

合成する表の定義

Key属性のマッピング定義

Name要素のマッピング定義

SQLマッピングによる合成 - データベースの準備

- ▶ DADファイルの内容を指定する方式で SQL マッピング方式のサンプルは、genxml_sql.cmd サンプルファイルに記述されています。
- ▶ サンプルファイルを実行する前に、第2話で触れました「XML Extender を使用するための準備」の(1)～(3)のステップを実行して、データベースの作成、バインドの実施、XML使用可能化を行う必要があります。
- ▶ SAMPLEデータベースの準備
 - このサンプルではDB2のSAMPLEデータベースが使用されていることに注意してください。サンプルデータベースが作成されていない場合には、コマンド・ウィンドウ (Windowsの場合) またはターミナル (Unixの場合) より`db2samp1`と入力してSAMPLEデータベースを作成することができます。
- ▶ データベースのバインドおよびMYDBのXML使用可能化
 - MYDBを準備したとき同様、第2話で用いた`getstart_prep.cmd`を利用することができます。元の`getstart_prep.cmd`ファイルの中で、データベース名に`SALES_DB`と指定されていた部分をSAMPLEに修正して実行します。

```
db2 "connect to SAMPLE"

cd /d %DB2DXXPATH%\%bnd
db2 "bind @dxxbind.lst"
cd /d %DB2PATH%\%bnd
db2 "bind @db2cli.lst"

db2 "terminate"

cd /d %DB2DXXPATH%\%samples%\cmd
dxxadm enable_db SAMPLE
```

この2ヶ所をSALES_DBからSAMPLEに修正します

SQLマッピングによる合成

- ▶ サンプルファイル内では、以下のステップが実行されます。
 - a. 合成した結果のXML文書を保管するための表(result_tab表)を作成します。ここで使用するXMLエクステンダーの合成ストアド・プロシージャ(dxxGenXML)は合成したXML文書を表に保管します。
 - b. dxxGenXML ストアド・プロシージャを使用してXML文書の合成を行います。引数に指定するのはDADファイルと合成結果のXML文書を保管する表の名前です。
 - *実際には、内部でdxxGenXMLストアド・プロシージャを使用するtests2xというプログラムを使用して、SAMPLEからのXML文書の合成を行っています。tests2xプログラムの内容は以下のファイルで確認できます。
 - Windowsの場合: (Install_Dir)¥samples¥c¥tests2x.sqx
 - Unixの場合: (Install_Dir)/samples/c/tests2x.sqx
 - c. 合成したデータを確認するために、SELECT文によりresult_tab表の内容を確認します。
 - d. 作成した表をドロップして、作成した環境を削除します。
- ▶ ここで使用するDADファイルは以下のファイルになります。
 - ▶ Windowsの場合: (Install_Dir)¥samples¥dad¥department.dad
 - ▶ UNIXの場合: (Install_Dir)/samples/dad/department.dad

SQLマッピングによる合成 - サンプル・スクリプト

▶ genxml_sql.cmd ファイルの内容 (Windows版)

```
@echo off
rem
rem genxml_sql - driver to test SQL-to-XML mapping stored procedure.
rem           Result column type: varchar
rem           Database: sample
rem
rem Notice:
rem     1. make sure that database sample is created via db2sampl
rem     2. make sure that sample database is bonded with XML Extender and CLI
rem     3. see ../c/tests2x.sqc for the way to call dxxGenXML()
rem
rem This file is shipped with XML Extender.

db2 "connect to sample"

db2 "echo ----- Creating result_tab -----"
db2 "create table result_tab(doc varchar(3000))"

db2 "echo ----- Calling the SQL-to-XML mapping stored procedure
tests2x sample %DB2DXXPATH%\samples\dad\department.dad result_tab"

db2 "echo ----- Displaying the resulting XML documents -----"
db2 "select * from result_tab"

db2 "echo ----- Cleaning up"
db2 "drop table result_tab"

db2 "terminate"
```

a

b

c

d

SQLマッピングによる合成 - DADファイル

▶ DADファイル (department.dad) の内容 (抜粋)

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:¥dxx¥dtd¥dad.dtd">
<DAD>
  <dtdid>department.dtd</dtdid>
  <validation>NO</validation>
  <Xcollection>
    <SQL_stmt>SELECT d.deptno, d.deptname, p.projno, p.prstdate, p.prstaff, p.projname, e.empno, e.job, e.firstnme, e.lastname
      FROM department d, employee e, project p
      WHERE e.workdept=d.deptno AND p.deptno=d.deptno AND d.deptname like '%SYSTEMS%'
      ORDER BY deptno, projno, empno
    </SQL_stmt>
  </Xcollection>
</DAD>
</?xml version="1.0"?>
<doctype>!DOCTYPE department SYSTEM "c:¥dxx¥dtd¥department.dtd"</doctype>
<root_node>
  <element_node name="department">
    <attribute_node name="id">
      <column name="deptno" />
    </attribute_node>
    <element_node name="name">
      <text_node>
        <column name="deptname" />
      </text_node>
    </element_node>
    <element_node name="work" multi_occurrence="NO">
      <element_node name="project">
        <attribute_node name="id">
          <column name="projno" />
        </attribute_node>
        <attribute_node name="start">
          <column name="prstdate" />
        </attribute_node>
      </element_node>
    </element_node>
  </root_node>
</doctype>
```

合成するデータを取り出すSQL文の定義

id属性のマッピング定義

name要素のマッピング定義

id属性のマッピング定義

start属性のマッピング定義

最後に

- ▶ 以上、4回にわたりXMLエクステンダーの基本的な機能について一通り概観しました。
- ▶ XMLエクステンダーには、今回触れたもののほかに、主だったものに以下のような機能があります。
 - 今回XMLコレクションで触れたXML文書の合成方法は、合成したXML文書を表 (result_tab表) に出力しましたが、合成した結果をCLOBで返すストアド・プロシージャ(dxxGenXMLClob、dxxRetrieveXMLClob)も用意されています。
 - XMLエクステンダーと弊社MQ Series製品を連携するための一連のストアド・プロシージャも用意されています。これらを用いることにより、MQ Seriesから渡されたXML文書をXMLエクステンダーを用いてDB2 UDBに保管したり、DB2 UDBに保管されているXML列やXMLコレクションに基づくXML文書をMQ Seriesに渡すことができます。
 - これら2点(合成結果をCLOBで返すストアド・プロシージャ、MQ Seriesとの連携に関するストアド・プロシージャ)の詳細については、FixPak4のRelease Notesをご参照ください。
<http://www-3.ibm.com/software/data/db2/extenders/xmlxt/support/fixpak.html>
 - 各種管理を実行できるGUIツールも用意されており、今回サンプル・ファイルの中でコマンドにより行われた、以下のようなことができます。詳細については製品マニュアルをご参照ください。
 - ▶ データベースのXML使用可能化/不可能化
 - ▶ DTDリポジトリへのDTDの保管
 - ▶ XML列、XMLコレクションの定義
 - ▶ XML列の使用可能化/不可能化、XMLコレクションの使用可能化/不可能化
 - ▶ DADファイルの作成/編集
 - テキスト情報エクステンダーと組み合わせることで使用することにより、XML列として保管したXML文書に対して全文検索を行うことが可能になります。
- ▶ 今回はXMLエクステンダーに付属するサンプルを元にその機能を見てきました。サンプルにあるファイルのパス、DADファイルなどを皆様の環境に合わせてカスタマイズすることで、XMLエクステンダーの機能を使用することが可能です。今回のご紹介が皆様への一助となれば、幸いです。