

IDUG Solutions Journal

IDUG Solutions Journal

Copyright © 2010 International DB2 Users Group

Table of Contents

From The Editor's Desk	1
Using DB2 pureXML and ODF Spreadsheets	2
Abstract	2
Introduction	2
The ODF specification	2
DB2 pureXML	2
The ODF document	2
A bit of hacking	4
Scheme of this article	6
Inserting XML document into DB2 table	6
IMPORT statements	6
Converting the content.xml into row-column format	7
Getting the cell values from the XML file	7
The XPath expression	7
The required query	8
Shopping prices	11
Building the list of shopping prices	11
View with list and cost of items	13
XQuery Update to build content.xml document	14
A quick side note on XQuery Update	14
Generating the content.xml for the shopping prices	15
Building the ><table:table-row> elements	15
Next steps	18
Resources	19
Biography	20
Giving You A Reason	21
Should you upgrade to DB2 10?	21
When and how should I upgrade to DB2 10?	21
DB2 9: Robust, Scalable, Available and Easily Manageable	22
DB2 10: Cut costs and improve performance	22
Questions for you	22
What version are you running?	23
DB2 and Storage Management	24
DB2 Storage Basics	24
Important DB2 for z/OS Storage Issues	25
A Little Bit About Modern Disk Arrays	26
Cache Versus Buffer	27
A Little Bit About DFSMS	27
What About Extents?	28
Another Thing to Think About	29
Best Practices	29
Summary	30
.....	31

List of Tables

1. XML files of an ODF document	3
2. Prefix and namespace URI used in content.xml document	8

List of Examples

1. DB2 table to hold content.xml file of the ODF spreadsheet	6
2. Contents of flat file to be used for loading DOCCONTENTXML table	6
3. IMPORT statement to load content.xml into DOCCONTENTXML table	7
4. XPath expression to get first row, second column value	7
5. Query to extract cell values from content.xml into tabular format	9
6. View on the content.xml document	10
7. Result of SELECT on the view V_CONTENT	10
8. DDL for table to hold the shopping prices	11
9. Query to obtain list and cost of items with grand total	12
10. Result of query for obtaining list and cost of items	13
11. View with list and cost of items in shopping list	14
12. A sample <table:table-row> element	16
13. Table to hold generated <table:table-row> elements	16
14. Insert generated <table:table-row> elements with <dummy> parent	17
15. Updated content.xml document	18

From The Editor's Desk

Philip Nelson

Welcome to the new look IDUG Solutions Journal, developed in response to your feedback received over the last few months.

Our main feature article this month is a case study in producing and consuming documents in ODF (Open Document Format) using DB2 pureXML.

We are also pleased to have a full complement of columns from our regular contributors.

Using DB2 pureXML and ODF Spreadsheets

Abstract

ODF (Open Document Format) is a file format for office documents. At a high level, the ODF specification requires five mandatory XML documents and other optional items. This article will describe how DB2 pureXML could be used to handle ODF spreadsheet documents. Other ODF documents such as word documents, presentations, formulas, etc. could also be handled by DB2 because the ODF specification refers mainly to an archive of XML documents. However, for the sake of this article, we are considering only spreadsheets because it is easier to co-relate the rows and columns of the spreadsheet document and that of a DB2 table. This article can even be further extended into an on-line document editing software with .ods as file format and DB2 pureXML as the database.

Introduction

The ODF specification and DB2 pureXML are introduced below.

The ODF specification

The Open Document Format specification was originally developed by Sun [<http://www.sun.com/>] and the standard was developed by OASIS Open Document Format for Office Applications (Open Document) TC – OASIS ODF TC. This standard is based on XML format originally created and implemented by the OpenOffice.org office suite. In addition to being a free and open OASIS standard, it is published (in one of its version 1.0 manifestations) as an ISO/IEC international standard, ISO/IEC 26300:2006 Open Document Format for Office Applications (OpenDocument) v1.0 [http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43485].

DB2 pureXML

DB2 pureXML [<http://www-01.ibm.com/software/data/db2/xml/>] is IBM software for management of XML data. It eliminates much of the work typically involved in the management of XML data. From validation of XML documents, to generation of documents and support for querying XML data in the form of XQuery [<http://www.w3.org/TR/xquery/>] and traditional SQL, DB2 pureXML is a complete solution to manage XML data in enterprise-scale databases.

The ODF document

An ODF document is a zipped file of five mandatory XML files and other optional components. This can be seen by running your favorite un-zip utility to list the components. Here is the result after running the unzip command on an .ods file from command line to list the files. The XML files are highlighted below.

```

nsubrahm@NSUBRAHM:~/Desktop/dw$ unzip -l Shop.ods
Archive:  Shop.ods
  Length      Date    Time    Name
-----
         46  05-13-09  09:44  mimetype
          0  05-13-09  09:44  Configurations2/statusbar/
          0  05-13-09  09:44  Configurations2/accelerator/current.xml
          0  05-13-09  09:44  Configurations2/floater/
          0  05-13-09  09:44  Configurations2/popupmenu/
          0  05-13-09  09:44  Configurations2/progressbar/
          0  05-13-09  09:44  Configurations2/menubar/
          0  05-13-09  09:44  Configurations2/toolbar/
          0  05-13-09  09:44  Configurations2/images/Bitmaps/
       7931  05-13-09  09:44  content.xml
       6482  05-13-09  09:44  styles.xml
        777  05-13-09  09:44  meta.xml
       5459  05-13-09  09:44  Thumbnails/thumbnail.png
       7228  05-13-09  09:44  settings.xml
       1896  05-13-09  09:44  META-INF/manifest.xml
-----
      29819                  15 files

```

Here is a short description of XML files as seen above.

Table 1. XML files of an ODF document

File name	Usage	Location in zipped file
manifest.xml	Information about the files contained in the package	/META-INF
styles.xml	Styles used in the document content and automatic styles used in the styles themselves	/ (root)
settings.xml	Application-specific settings, such as the window size or printer information	/ (root)
content.xml	Document content and automatic styles used in the content	/ (root)
meta.xml	Document meta information, such as the author or the time of the last save action	/ (root)

For this article, we are interested in the content.xml file which will hold the contents of the document. The document used for this article is a simple shopping list named Shop.ods. Below is the spreadsheet as it appears in OpenOffice, with the first row highlighted.

	A	B	C
1	Potato	2.0	kg
2	Onion	1.0	kg
3	Cucumber	0.5	kg
4	Capsicum	0.5	kg
5	Carrot	0.5	kg
6	Green chilli	0.25	kg
7	Tomato	0.25	kg
8	Curd	0.5	L
9	Banana	2	doz
10	Milk	2	L
11	Groundnut Oil	5	L

Example ODF spreadsheet (row highlighted)

And below we have part of the equivalent content.xml file, with the XML markup for the row highlighted.

```
-<office:document-content office:version="1.2">
  <office:scripts/>
  +<office:font-face-decls></office:font-face-decls>
  +<office:automatic-styles></office:automatic-styles>
  -<office:body>
    -<office:spreadsheet>
      -<table:table table:name="Sheet1" table:style-name="ta1" table:print="false">
        <table:table-column table:style-name="co1" table:default-cell-style-name="Default"/>
        <table:table-column table:style-name="co2" table:default-cell-style-name="Default"/>
        <table:table-column table:style-name="co3" table:default-cell-style-name="Default"/>
        -<table:table-row table:style-name="ro1">
          -<table:table-cell office:value-type="string">
            <text:p>Potato</text:p>
          </table:table-cell>
          -<table:table-cell table:style-name="ce1" office:value-type="float" office:value="2">
            <text:p>2.0</text:p>
          </table:table-cell>
          -<table:table-cell office:value-type="string">
            <text:p>kg</text:p>
          </table:table-cell>
        </table:table-row>
      +<table:table-row table:style-name="ro1"></table:table-row>
```

XML representation of ODF spreadsheet (row highlighted)

A bit of hacking

Now that we know a bit about an ODF document, let us try to create one without actually running a standard software dealing with ODF documents. As noted above, the content.xml file holds the data entered while creating the spreadsheet. So, if this file is changed and zipped along with the rest of the files in the original archive (that is, the .ods file), we will be having a “legal” ODF spreadsheet. To test this, follow the steps below.

1. Open the content.xml file in a regular text editor
2. In ???some figure???, the element <table:table-row> has been highlighted. Type in the contents of the highlighted section immediately as a sibling of another <table:table-row> element.
3. In the pasted element, make a change in any of the table:table-cell/text:p element say, something like this -

```

- <table:table-row table:style-name="ro1">
- <table:table-cell office:value-type="string">
  <text:p>Onion</text:p>
</table:table-cell>
- <table:table-cell table:style-name="ce1" office:value-type="float" office:value="1">
  <text:p>1.0</text:p>
</table:table-cell>
- <table:table-cell office:value-type="string">
  <text:p>kg</text:p>
</table:table-cell>
</table:table-row>
<table:table-row table:style-name="ro1">
- <table:table-cell office:value-type="string">
  <text:p>Coffee Powder</text:p>
</table:table-cell>
- <table:table-cell table:style-name="ce1" office:value-type="float" office:value="250">
  <text:p>250</text:p>
</table:table-cell>
- <table:table-cell office:value-type="string">
  <text:p>gms</text:p>
</table:table-cell>
</table:table-row>

```

Edited content.xml showing added row

- Save the edited content.xml file in the same folder where the Shop.ods file was unzipped into, replacing the older one.
- Select all the contents (that is, the ones mentioned in the manifest.xml file) of the unzipped folder and zip them. Name the zipped file as NewShop.ods.
- Now, open NewShop.ods in OpenOffice or IBM Lotus Symphony and confirm that the hacking is successful. Note that OpenOffice successfully opens a file with a .zip extension, so long it is a valid ODF document whereas Symphony requires the extension to be .ods.

	A	B	C	D
1	Potato	2 kg		
2	Onion	1 kg		
3	Coffee Powder	250 gms		
4	Cucumber	0.5 kg		
5	Capsicum	0.5 kg		
6	Carrot	0.5 kg		
7	Green chilli	0.25 kg		
8	Tomato	0.25 kg		
9	Curd	0.5 L		
10	Banana	2 doz		
11	Milk	2 L		
12	Groundnut Oil	5 L		

??? insert suitable caption here ???

It is now clear that, to “read” an existing ODF spreadsheet in its simplest form, we have to navigate the path of the content.xml XML document. Similarly, to “modify” an existing ODF spreadsheet, we should generate the content.xml XML document correctly. One can, of course, extend this to create and/or modify other XML documents in the .ods archive. But, for the sake of this article, we will concentrate only on the creation of content.xml XML document.

Scheme of this article

With this knowledge of XML documents in an ODF document archive, we will now proceed to demonstrate the usage of DB2 pureXML with ODF documents. We consider the scheme below.

1. We consider an ODF spreadsheet of a simple shopping list where the item name, unit of measure (UOM) and the units required are provided.
2. The content.xml document of this ODF spreadsheet is imported into a DB2 table with an XML column.
3. For the sake of this article we will use the LOAD statement in DB2 to load the content.xml document and ignore any other methods of inserting XML documents into the DB2 table.
4. We will refer to another traditional table which stores the price list.
5. Using the table for price list and the table with the XML column for content.xml, we will generate another content.xml XML document that will hold the items with their corresponding prices.

Inserting XML document into DB2 table

We will consider a table to demonstrate DB2 pureXML capabilities. The first table will be keyed via an ID column to differentiate multiple documents. The second column of the table will hold the content.xml file of the ODF spreadsheet document in a XML column. The DDL for this table is reproduced here:

Example 1. DB2 table to hold content.xml file of the ODF spreadsheet

```
-- DOCCONTENTXML table to hold the content.xml file of a ODF document.  
-- The table is keyed on DOCID a running sequence number.  
CREATE TABLE ODF.DOCCONTENTXML  
(  
    DOCID          INTEGER NOT NULL,  
    DOCCONTENT XML NOT NULL  
) ;
```

IMPORT statements

Inserting the XML document into the table can be done using a simple INSERT SQL statement or using IMPORT statement. However, since we have a considerably sized XML document, we will use the IMPORT statement from the command line. To use the IMPORT statement, first, save the content.xml in a folder as say, ODSFolder and create a flat file with a single record. Save the flat file as load.txt. This record will have two values separated by a comma as shown below.

Example 2. Contents of flat file to be used for loading DOCCONTENTXML table

```
1,<XDS FIL='content.xml' />
```

Example 3. IMPORT statement to load content.xml into DOCCONTENTXML table

```
IMPORT FROM load.txt OF DEL
XML FROM ODSFolder
INSERT INTO ODF.DOCCONTENTXML;
```

Alternatively, IBM Data Studio Developer can be used to load into XML column as it offers a convenient GUI.

Converting the content.xml into row-column format

Getting the cell values from the XML file

To get to the cell values of the spreadsheet, we will start with the required XPath expression. Then, we will develop a working query around the XPath expression to get the results.

The XPath expression

Going back to Table 2, the left column shows the first row of the spreadsheet highlighted. The right column is a part of the generated content.xml file and some of the elements collapsed. The relevant part are put in red boxes. The first box shows the table:table element with an attribute table:name whose value is "Sheet1". It is the name of the sheet in the complete spreadsheet. This element is repeated as many times as there are sheets in the spreadsheet with the table:name attribute set to the names as set by the user. The second box is the representation of the highlighted row in the left column. Thus, to get to, say, column B of the first row, the path to be navigated (or, the XPath expression) from the root of XML document would be as below.

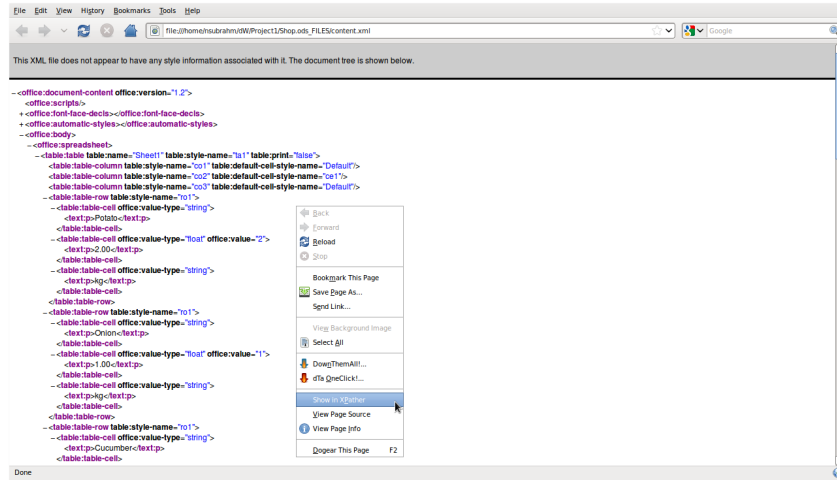
Example 4. XPath expression to get first row, second column value

```
/office:document-content/office:body/office:spreadsheet/table:table[1]/table:
```

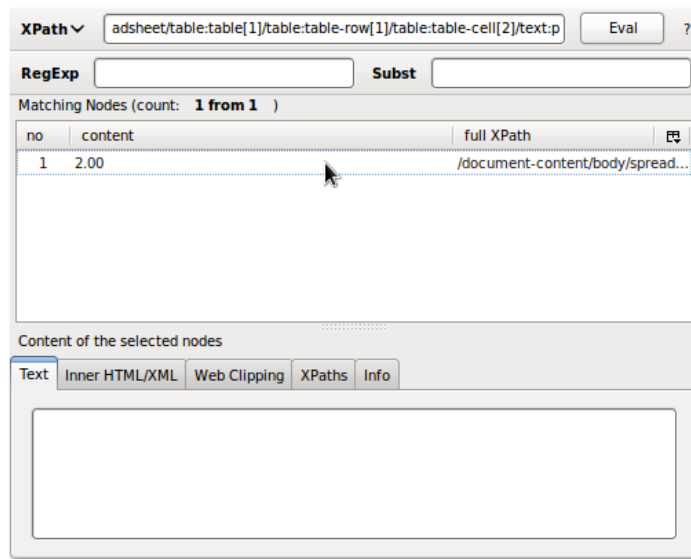
In other words, we start from the office:document element to office:body and then to office:spreadsheet to look for the first table:table element. The first element represents the first sheet of the spreadsheet document. Then, to get to the second column of first row we navigate to table:row[1]/table:cell[2] and finally to text:p to get the actual text of the cell.

With Firefox Mozilla browser, the above XPath expression can be easily tested using the add-on Xpather (see Resources below). We follow the steps below :-

1. Open the content.xml in Firefox Mozilla browser
2. Right-click on the page and select 'Show in Xpather'



3. The XPath dialog box opens up. Enter the XPath expression in the dialog box and click 'Eval'



The required query

To wrap the XPath expression into a query, we first note the XML namespaces used. The content.xml file has the namespaces defined as shown below :

Table 2. Prefix and namespace URI used in content.xml document

Prefix	Namespace
office	"urn:oasis:names:tc:opendocument:xmlns:office:1
table	"urn:oasis:names:tc:opendocument:xmlns:table:1
text	"urn:oasis:names:tc:opendocument:xmlns:text:1.0

The query shown below will use the XML namespaces and XPath expression that we produced in the previous sections to extract the cell values from the loaded content.xml file. The highlighted text shows the XPath expression used to extract cell values.

Example 5. Query to extract cell values from content.xml into tabular format

```
SELECT
X.*
FROM
XMLTABLE(
XMLNAMESPACES('urn:oasis:names:tc:opendocument:xmlns:office:1.0' AS
"office",
'urn:oasis:names:tc:opendocument:xmlns:table:1.0' AS "table",
'urn:oasis:names:tc:opendocument:xmlns:text:1.0' AS "text"),
'db2-fn:sqlquery("SELECT
DOCCONTENT
FROM
ODF.DOCCONTENTXML
WHERE DOCID
= 1")
/office:document-content/office:body/office:spreadsheet/table:table[1]/table'
) AS X;

COLUMNS
"ITEMNAME" CHARACTER (20) PATH
'table:table-cell[1]/text:p',
"QUANTITY" DECIMAL (4,2) PATH
'table:table-cell[2]/text:p',
"UOM" CHARACTER (5) PATH
'table:table-cell[3]/text:p'
```

To see the content.xml document in traditional rows and columns format, we create a view using the query above.

Example 6. View on the content.xml document

```
CREATE VIEW ODF.V_CONTENT AS
(
SELECT
X.*
FROM
XMLTABLE(
XMLNAMESPACES(
'urn:oasis:names:tc:opendocument:xmlns:office:1.0' AS "office",
'urn:oasis:names:tc:opendocument:xmlns:table:1.0' AS "table",
'urn:oasis:names:tc:opendocument:xmlns:text:1.0' AS "text"
),
'db2-fn:sqlquery("SELECT
DOCCONTENT
FROM
ODF.DOCCONTENTXML
WHERE
DOCID = 1"
)/office:document-content/office:body/office:spreadsheet/table:table[1]/table
COLUMNS
"ITEMNAME" CHARACTER (20) PATH 'table:table-cell[1]/text:p',
"QUANTITY" DECIMAL (4,2) PATH 'table:table-cell[2]/text:p',
"UOM" CHARACTER (5) PATH 'table:table-cell[3]/text:p'
) AS X
);
```

A simple SELECT query on this view shows that we have successfully 'converted' the content.xml document in the ODF archive for spreadsheet into a row-column format.

Example 7. Result of SELECT on the view V_CONTENT

```
SELECT * FROM ODF.V_CONTENT ;
```

ITEMNAME	QUANTITY	UOM
Potato	2.00	kg
Onion	1.00	kg
Cucumber	0.50	kg
Capsicum	0.50	kg
Carrot	0.50	kg
Green chilli	0.25	kg
Tomato	0.25	kg
Curd	0.50	L
Banana	2.00	doz
Milk	2.00	L
Groundnut Oil	5.00	L

11 record(s) selected.

Shopping prices

We will now build the content.xml for an ODF spreadsheet that will hold the shopping list – discussed above – with an additional column for cost of items in the shopping list. We will also provide an additional row for the grand total of the items in shopping list.

Building the list of shopping prices

We create a simple table that holds the unit price of shopping items with the DDL shown below.

Example 8. DDL for table to hold the shopping prices

```
--Table to hold the unit price of shopping items
CREATE TABLE ODF.SHOPPER
(ITEMNAME CHARACTER(50),
UNIT_PRICE DECIMAL (4,2));
```

This table is inserted with values such that all the item names are available in the SHOPPER table. To obtain the list of item names and the cost of items, we use the following query. Note that, this query uses the view V_CONTENT. We could have used the base table directly. Usage of the view makes the query 'compact' and easier to follow.

Example 9. Query to obtain list and cost of items with grand total

```
SELECT
COALESCE(P.ITEMNAME, ' TOTAL')
ITEMNAME,
SUM(P.QUANTITY) QTY,
COALESCE(P.UNIT_PRICE, 0) UNIT_PRICE,
SUM(P.COST) COST
FROM
(SELECT
A.ITEMNAME,
A.QUANTITY,
B.UNIT_PRICE,
A.QUANTITY*B.UNIT_PRICE COST
FROM
ODF.V_CONTENT A,
ODF.SHOPPER B
WHERE
UPPER(A.ITEMNAME) = B.ITEMNAME
) P
GROUP BY
ROLLUP(P.ITEMNAME,P.QUANTITY,P.UNIT_PRICE)
HAVING
(P.ITEMNAME IS NULL OR
(P.ITEMNAME IS NOT NULL AND
P.QUANTITY IS NOT NULL AND
P.UNIT_PRICE IS NOT NULL AND
SUM(P.COST) IS NOT NULL AND
SUM(P.QUANTITY) IS NOT NULL
)
)
ORDER BY COALESCE(P.ITEMNAME, ' ') DESC;
```

The result of the above query is shown below.

Example 10. Result of query for obtaining list and cost of items

ITEMNAME	QTY	UNIT_PRICE	COST
-----	-----	-----	-----
Tomato	0.25	6.00	1.50
Potato	2.00	5.50	11.00
Onion	1.00	4.00	4.00
Milk	2.00	40.00	80.00
Groundnut Oil	5.00	30.00	150.00
Green chilli	0.25	8.00	2.00
Curd	0.50	20.00	10.00
Cucumber	0.50	10.00	5.00
Carrot	0.50	15.00	7.50
Capsicum	0.50	20.00	10.00
Banana	2.00	6.00	12.00
TOTAL	14.50	0.00	293.00

View with list and cost of items

Now that we have the result of computation of cost of items, we will take this into a view. This is purely for convenience. We can proceed without using this view.

Example 11. View with list and cost of items in shopping list

```
CREATE VIEW ODF.V_SHOP_COST AS
(
  SELECT
    COALESCE(P.ITEMNAME, ' TOTAL') ITEMNAME,
    SUM(P.QUANTITY) QTY,
    COALESCE(P.UNIT_PRICE, 0) UNIT_PRICE,
    SUM(P.COST) COST
  FROM
    (
      SELECT
        A.ITEMNAME,
        A.QUANTITY,
        B.UNIT_PRICE,
        A.QUANTITY*B.UNIT_PRICE COST
      FROM
        ODF.V_CONTENT A,
        ODF.SHOPPER B
      WHERE
        UPPER(A.ITEMNAME) = B.ITEMNAME
    ) P
  GROUP BY
    ROLLUP(P.ITEMNAME, P.QUANTITY, P.UNIT_PRICE)
  HAVING
    (P.ITEMNAME IS NULL OR
     (P.ITEMNAME IS NOT NULL AND
      P.QUANTITY IS NOT NULL AND
      P.UNIT_PRICE IS NOT NULL AND
      SUM(P.COST) IS NOT NULL AND
      SUM(P.QUANTITY) IS NOT NULL
     )
    )
);
```

XQuery Update to build content.xml document

At this point, we have 'shredded' the shopping list (from the content.xml of ODF spreadsheet), joined with the relational table having the unit prices of items and generated a relational output of the cost of items in shopping list with the grand total. Our aim is now to convert this relational output into the format of content.xml so that the result is a complete ODF spreadsheet.

One way of building the content.xml is to start from scratch and building the whole of the document. This approach would be quite tedious. A better way would be to generate the relevant part in the content.xml part and then insert into an otherwise empty content.xml document. This approach will also allow us to make use of the XQuery Update facility in DB2 pureXML.

A quick side note on XQuery Update

XQuery is a query and functional programming language designed to query XML documents. It became a W3C Candidate Recommendation on 23rd January, 2007. XQuery Update facility is an extension to XQuery that allows update (insert, modify and delete) of the XML documents. It became a W3C Candidate Recommendation on 14th March, 2008.

Generating the content.xml for the shopping prices

To start with, we will create a dummy content.xml that is complete and ODF compliant in all respects except it would resemble an empty spreadsheet. Next, we will generate XML sequences that is correct for a row and its columns of a spreadsheet. Then, we will collect all of these XML sequences and insert into the dummy content.xml document.

To create the dummy document, we simply open the spreadsheet for shopping list, delete all rows and columns and save it with another name. Then, we unzip this document and refer the content.xml document. Here is a screenshot of how this document would look.

```
-<office:document-content office:version="1.2">
  <office:scripts/>
  + <office:font-face-decls></office:font-face-decls>
  + <office:automatic-styles></office:automatic-styles>
  -<office:body>
    -<office:spreadsheet>
      -<table:table table:name="1" table:style-name="ta1" table:print="false">
        <table:table-column table:style-name="co1" table:default-cell-style-name="ce1"/>
        <table:table-column table:style-name="co2" table:default-cell-style-name="ce1"/>
        <table:table-column table:style-name="co3" table:default-cell-style-name="Default"/>
        <table:table-column table:style-name="co4" table:default-cell-style-name="Default"/>
        <table:table-column table:style-name="co5" table:default-cell-style-name="Default"/>
      </table:table>
    </office:spreadsheet>
  </office:body>
</office:document-content>
```

content.xml for an empty spreadsheet

Insert this dummy document into the ODF.DOCCONTENTXML table with DOCID as zero using the IMPORT method described before. Our aim now is to update this document such that, <table:table-row> elements are added as children of <table:table> and as siblings of <table:table-column>.

Building the <table:table-row> elements

In our example, each row of spreadsheet would have four columns. Therefore, each row element, as required in content.xml, would have four children for the number of columns. A sample of <table:table-row> is shown below. The XML namespace declarations have been omitted for the sake of brevity.

Example 12. A sample <table:table-row> element

```
<table:table-row  
table:style-name="rol">  
<table:table-cell  
office:value-type="string"><text:p>Tomato</text:p></table:table-cell>  
<table:table-cell  
office:value-type="string"><text:p>.25</text:p></table:table-cell>  
<table:table-cell  
office:value-type="string"><text:p>6.00</text:p></table:table-cell>  
<table:table-cell  
office:value-type="string"><text:p>1.5000</text:p></table:table-cell>  
</table:table-row>
```

To build the <table:table-row> elements we need to generate <table:table-cell> for each column of the row in the relational table and then aggregate them to a single <table:table-row> element. We will use the XMLAGG function to build the elements and store in a table the generated elements so that it is easier to use while doing the update of XML document. The table to hold the generated elements is defined as shown below.

Example 13. Table to hold generated <table:table-row> elements

```
CREATE TABLE ODF.SHOP_COST_XML  
(SHOP_COST XML);
```

Example 14. Insert generated <table:table-row> elements with <dummy> parent

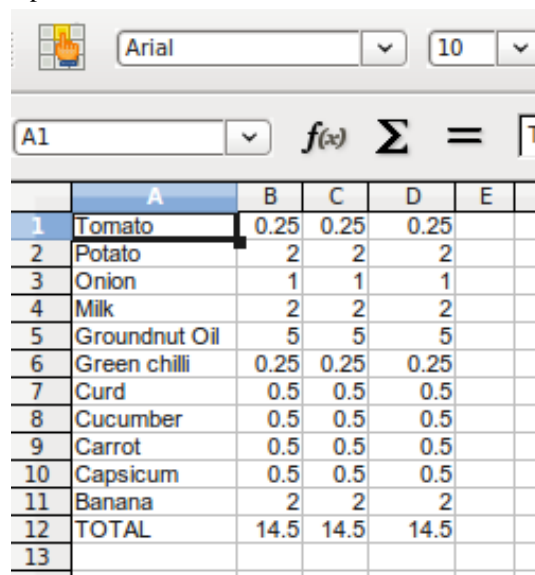
```
INSERT INTO ODF.SHOP_COST_XML
(SHOP_COST)
(SELECT
XMLDOCUMENT(
  XMLELEMENT(NAME "dummy",
    XMLNAMESPACES(
      'urn:oasis:names:tc:opendocument:xmlns:office:1.0' AS "office",
      'urn:oasis:names:tc:opendocument:xmlns:table:1.0' AS "table",
      'urn:oasis:names:tc:opendocument:xmlns:text:1.0' AS "text"
    ),
    XMLAGG(
      XMLELEMENT(NAME "table:table-row",
        XMLATTRIBUTES('rol' AS "table:style-name"),
        XMLELEMENT(NAME "table:table-cell",
          XMLATTRIBUTES('string' AS "office:value-type"),
          XMLELEMENT(NAME "text:p", STRIP(O.ITEMNAME))
        ),
        XMLELEMENT(NAME "table:table-cell",
          XMLATTRIBUTES('string' AS "office:value-type"),
          XMLELEMENT(NAME "text:p", O.QTY )
        ),
        XMLELEMENT(NAME "table:table-cell",
          XMLATTRIBUTES('string' AS "office:value-type"),
          XMLELEMENT(NAME "text:p", O.UNIT_PRICE )
        ),
        XMLELEMENT(NAME "table:table-cell",
          XMLATTRIBUTES('string' AS "office:value-type"),
          XMLELEMENT(NAME "text:p", O.COST)
        )
      )
    )
  ORDER BY O.ITEMNAME DESC
)
)
)
FROM ODF.V_SHOP_COST O
);
```

Finally, we are now ready to update our dummy XML document. The query to do the update is shown below. Note that, this query is not inserting the resulting into any table. It is left to the reader to route the output to a table column, file, message queue, etc.

Example 15. Updated content.xml document

```
VALUES XMLQUERY(  
  'declare namespace office="urn:oasis:names:tc:opendocument:xmlns:office:1.0";  
  declare namespace table="urn:oasis:names:tc:opendocument:xmlns:table:1.0" ;  
  transform  
  copy $dummy := db2-fn:sqlquery("SELECT DOCCONTENT FROM ODF.DOCCONTENTXML WHE  
  $rows := db2-fn:sqlquery("SELECT * FROM ODF.SHOP_COST_XML")  
  modify do  
  insert $rows/dummy/* as last  
  into $dummy/office:document-content/office:body/office:spreadsheet/table:tab  
  return $dummy'  
);
```

To test that our update has worked successfully run, we route the output of query above to a file called content.xml. Replace the content.xml file in the unzipped archive of our original shopping list spreadsheet with the one created now. Zip the files back and save it with name as ShopPrices.ods and open with OpenOffice. Here is the screenshot.



The screenshot shows an OpenOffice spreadsheet window. The toolbar at the top includes a font dropdown set to 'Arial' and a size dropdown set to '10'. Below the toolbar is a formula bar showing 'A1' and a function dropdown set to 'f(x)'. The spreadsheet grid has columns A through E. Row 1 is highlighted. The data in the grid is as follows:

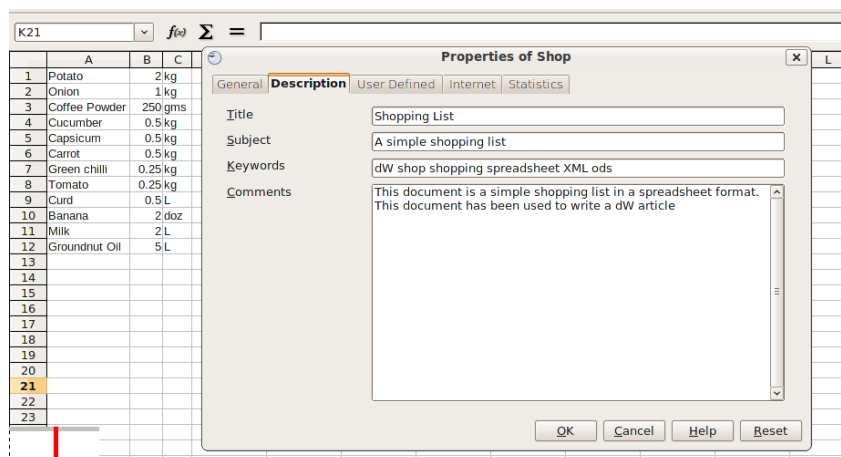
	A	B	C	D	E
1	Tomato	0.25	0.25	0.25	
2	Potato	2	2	2	
3	Onion	1	1	1	
4	Milk	2	2	2	
5	Groundnut Oil	5	5	5	
6	Green chilli	0.25	0.25	0.25	
7	Curd	0.5	0.5	0.5	
8	Cucumber	0.5	0.5	0.5	
9	Carrot	0.5	0.5	0.5	
10	Capsicum	0.5	0.5	0.5	
11	Banana	2	2	2	
12	TOTAL	14.5	14.5	14.5	
13					

ShopPrices.ods

And, we are done !

Next steps

Further enhancements may involve manipulating other XML documents in a ODF compliant archive. For example, the meta.xml document holds information about the document itself. This information can be accessed and set by going to File -> Properties... dialog box. For example, for the document in this article (Shop.ods), this is how the dialog box looks like.



Document properties

The corresponding meta.xml document (partial snapshot) is shown below.

```
--<office:meta>
  <meta:creation-date>2009-05-29T17:39:45</meta:creation-date>
  <meta:editing-duration>PT00H01M16S</meta:editing-duration>
  <meta:editing-cycles>2</meta:editing-cycles>
--<meta:generator>
  OpenOffice.org/3.0$Linux OpenOffice.org_project/300m15$Build-9379
</meta:generator>
<meta:initial-creator>Nagesh Subrahmanyam</meta:initial-creator>
--<dc:description>
  This document is a simple shopping list in a spreadsheet format. This document has been used to write a dW article
</dc:description>
<meta:keyword>dW shop shopping spreadsheet XML ods</meta:keyword>
<dc:subject>A simple shopping list</dc:subject>
<dc:title>Shopping List</dc:title>
<dc:date>2009-05-29T17:39:47</dc:date>
<dc:creator>Nagesh Subrahmanyam</dc:creator>
<meta:document-statistic meta:table-count="3" meta:cell-count="36" meta:object-count="0"/>
<meta:user-defined meta:name="Info 1"/>
<meta:user-defined meta:name="Info 2"/>
<meta:user-defined meta:name="Info 3"/>
<meta:user-defined meta:name="Info 4"/>
</office:meta>
```

meta.xml (partial)

Thus, while generating the spreadsheet, one can write queries to generate the meta.xml document with correct values for creator, creation time-stamp, etc.

Resources

Important resources :

DB2 Express-C [<http://www-01.ibm.com/software/data/db2/express/download.html>]

– Version of DB2 which is free to download, develop, deploy and distribute.

DB2 v9 pureXML [<http://www.ibm.com/db2/9>]

All information about DB2 V9 pureXML

DB2 v9 Product Manuals [http://www.ibm.com/support/docview.wss?rs=71&uid=swg27009552]	Especially, refer the ones for XQuery and XML.
IBM Data Studio [http://www-01.ibm.com/software/data/studio/]	Govern, design, develop and deploy databases and data driven applications.
ODF Specification [http://www.oasis-open.org/committees/office/]	Link to the complete specification
OpenOffice.org XML essentials [http://books.evc-cit.info/book.php]	– Nice explanation of XML data formats.
Thinking XML : The open office file format [http://www.ibm.com/developerworks/xml/library/x-think15/]	An earlier developerWorks article discussing the file formats of OpenOffice.
Manage ODF and Microsoft Office 2007 documents with DB2 9 pureXML [http://www.ibm.com/developerworks/db2/library/techarticle/dm-0705gruber/]	An earlier developerWorks article on using ODF and Microsoft Office documents with DB2 9 pureXML.
XPather add-on [https://addons.mozilla.org/addon/1192]	Nifty tool for Firefox Mozilla users to quickly check an XPath expression
Update XML in DB2 9.5 [http://www.ibm.com/developerworks/db2/library/techarticle/dm-0710nicola/%5C]	Nice article on developerWorks explaining update of XML documents

Biography

Nagesh Subrahmanyam is a Software Developer in IBM Global Services, India.....

Giving You A Reason

Compelling Reasons to Upgrade DB2

Roger Miller



Is your current bowl getting tight? What is limiting you? Is it CPU? Virtual storage? Latching? Catalog and directory? Utilities? Are you currently running DB2 9? V8? V7? Upgrading to a bigger bowl may be just what you need.

Should you upgrade to DB2 10?

To 10, or not to 10, when and how are the questions. Whether 'tis nobler in the mind to suffer the slings and arrows of outrageous limits, Or to take arms against a sea of troubles, And by upgrading, end them? To die: to sleep; No more; and by a sleep to say we end the heart-aches and the thousand natural shocks that old versions are heir to. 'tis a consummation devoutly to be wished. To die, to sleep. To sleep: perchance to dream: ay, there's the rub; For in that sleep of death what dreams may come when versions have shuffled off this mortal coil, must give us pause: There's the respect that makes calamity of too long life for old versions. [With abject apologies to the Bard and to Hamlet act 3 scene 1]

The answer to upgrading to 10 is a definite Yes. The question is not so much whether to upgrade as when and how to upgrade. If you are running DB2 9 today, then DB2 10 is in your near future, giving you more room to grow, with higher limits, lower costs, and more for less. If you are running DB2 V8 today, then you have a choice of jumping to DB2 9 or directly to DB2 10. So the key question is ...

When and how should I upgrade to DB2 10?

As of early August 2010, DB2 10 is in beta. Some of the key information for making this decision is known, but some is not yet. DB2 for z/OS V8 end of service is set for April 2012, 21 months from now. The unknown information includes the date for DB2 10 general availability, V8 extended service, and pricing, which will come in later announcements.

While DB2 10 is expected to be better than prior versions, it will have maturity, stability, and service delivery similar to other software and versions, with more defects at first, then fewer as the software matures. Determining when the software is ready for a specific customer and when the customer is ready for the software depends upon the specific customer resources, prior experience, and the value for the improvements versus the need for stability. Many customers depend upon tools or other software, and having that software that works with DB2 is a prerequisite. When this information is known, we can help answer the question. This web page can help. <http://www.ibm.com/support/docview.wss?uid=swg21006951> Content of the two versions is available, although the details of DB2 10 are still not public. Here is a summary of the two versions -

DB2 9: Robust, Scalable, Available and Easily Manageable

DB2 9 delivers CPU reductions for utilities that are generally in the range of 20% to 30%. Customers report saving terabytes of disk space using index compression. More CPU time is shifted to zIIP processors, reducing costs. Security is improved with more flexible trusted contexts and roles. Resilience is improved as more changes can be made while applications keep running. One table can be replaced quickly with a clone. Indexes and columns can be renamed. Many more utilities can be online.

DB2 9 delivers seamless integration of XML and relational data with pureXML and makes big strides in SQL for productivity and portability of applications. A new storage structure is introduced for large tables. Today's complex applications include both transactions and reporting, so performing both well is required. The key improvements for reporting are optimization enhancements to improve query and reporting performance and ease of use. More queries can be expressed in SQL with new SQL enhancements. Improved data is provided for the optimizer, with improved algorithms. Improved CPU and elapsed times can be achieved with the FETCH FIRST clause specified on a subquery. The INTERSECT and EXCEPT clauses make SQL easier to write.

DB2 10: Cut costs and improve performance

DB2 10 for z/OS provides the best reduction in CPU for transactions and batch in over 20 years. We expect most customers to reduce CPU times between 5% and 10% initially, with the opportunity for much more. Applications which can take advantage of additional benefits, such as hash access, can have larger CPU and memory reductions.

Scalability is the second major benefit, with the ability to run five to ten times as many threads in a single subsystem by moving 80% to 90% of the virtual storage above the bar. Schema evolution or data definition on demand enhancements improves availability. SQL and pureXML improvements extend usability and application portability for this platform. Productivity improvements for application developers and for database administrators are very important as data grows in scale and complexity.

Questions for you

The right answer is not "One size fits all." If we know the key factors for you, we can help you make a better choice. Here are some of the key objectives. Which ones are most important for you?

- Performance improves in both DB2 9 and 10, with larger CPU reductions in DB2 10.
- Scalability is improved a little in DB2 9 and a lot in DB2 10.
- Availability is enhanced in both, with more online changes in both.

- Security is made stronger and more flexible with roles in DB2 9 and with more options in DB2 10.
- Productivity is helped in both releases, with more improvements in DB2 10. Upgrading to DB2 9 is easier than to DB2 10 from DB2 V8.
- Stability is better in more mature versions.
- Skills: What skill set is available within your organization? Do you have people with the right skills and time to plan and run a project? DB2 planning workshops can help with education. Transition classes provide more education for one or both versions. Services could be used if the skills are not presently available.
- Technology adoption model: Are you using the latest technology that is being shipped, or are the operating system and hardware back-level? Is the technology one level back, two or more? This question tells both how much work will be required and the comfort level of your organization for running the latest version.
- Platform management practices: What are your platform management practices? What type of change management practices are in place? How robust is your testing for new software? What is the inventory of software and tools? How many vendors are involved? Which ones? Almost every vendor has software ready for DB2 9, but DB2 10 may take some time after general availability.
- Numbers of servers: How many LPARs and subsystems does the organization have? An organization that has 100 subsystems has a different set of challenges than does one that has 5 subsystems.
- Organizational considerations: What additional organizational factors, such as politics and policies, must be considered?

What version are you running?

Here are the primary recommendations for customers who are running various DB2 versions.

- DB2 9 If you are on DB2 9 today, then you are a good candidate for an early upgrade to DB2 10, especially if your custom is to move in the first year after general availability. Listen to reports from early customers and upgrade for the value.
- V8 If you are on DB2 V8 today, then the next questions are on timing for you and readiness for the new version. How soon after general availability do you normally upgrade? Are you still in the process of moving to NFM or have you recently finished V8 upgrade? If you just finished, then you probably will wait a few years and use the skip. If you have resources for an upgrade, but DB2 10 is too new, then DB2 9 is probably your next move. If you have the resources and can work with a new version, then skipping to DB2 10 may work for you.
- V7 If you are currently on DB2 V7, then upgrade to DB2 V8. Then you can use the skip version upgrade to DB2 10 in a few years.

DB2 has several new versions and upgrade paths for you to consider. This story will be changing, but you can hear the latest at IDUG conferences and on the web. DB2 9 is ready for you now. DB2 10 is still in beta, but is delivering higher limits, lower costs, and more for less.

- <http://www.ibm.com/software/data/db2/zos/db2-10/>
- <http://www.ibm.com/data/db2/zos>

DB2 and Storage Management

Craig S. Mullins

As an IT professional who uses DB2, you know that all database management systems rely on some form persistent storage to maintain data. That means that the DBMS interoperates with operating system files, or data sets. As such, some form of storage management should be a key part of the database operations required of a DBA. Typically database storage means disk devices or subsystems, but it can also mean solid state disk, removable storage, or even trusty “old” tape devices.

DB2 Storage Basics

At the most basic level it is important to know that DB2 stores its data in VSAM Linear Data Sets (LDS). Each table space and index space you create requires at least one, possibly more, underlying VSAM data sets. DB2 uses VSAM Media Manager for its I/O operations. For every I/O, VSAM Media Manager builds a channel program and sends a request to the I/O supervisor.

But let’s back up a moment. The following items are the core of the storage-related objects and items you will need to know about for DB2 for z/OS:-

DB2 Storage Groups	List of disk volumes. DB2 can also work with SMS so you will need to differentiate between DB2 storage groups and SMS storage groups (see Figure 1).
Table Spaces	Stored on disk as at least one VSAM LDS data set
Indexes	Stored on disk (in an index space) as at least one VSAM LDS data set
System data sets	
Active Log	Stored on disk
Archive Logs	Stored on disk or tape
BSDS	Stored on disk
Image Copy Backups	Stored on disk or tape
Image Copy Backups	Stored on disk or tape
DB2 library data sets	
Temporary data sets	Used by utilities

Figure 1. DB2 Storage Groups vs. SMS Storage Groups

So you can see that there are multiple areas within DB2 that require storage and need to be managed. This article will touch upon most of these areas. But back to data set basics for the time being.

You may have noticed that I said that multiple data sets may be required so when does DB2 utilize multiple VSAM data sets for a table space or index? There are three situations where this will arise:-

1. When the object is partitioned, each partition will reside in a separate data set.
2. When a data set in a segmented or simple table space reaches its maximum size of 2 GB, DB2 can automatically create a new data set.

3. When the table space is cloned; each clone has its own underlying data set(s).

To understand how DB2 accommodates these situations we will need to take a look at how DB2 data sets are named. Figure 2 shows the naming convention for DB2 data sets. Although many of you reading this article may be familiar with this naming convention, a quick review is still a good idea. The database name and the page set name (table space or index space name) is part of the data set name. This is one of the reasons that you cannot have more than one table space or index space of the same name in the same database. The “interesting” part of the naming convention (if a naming convention can be interesting at all) comes at the end. We have an instance qualifier and a data set number.

Figure 2. DB2 Data Set Naming Convention

The instance qualifier is used when running online REORG and CHECK utilities. For an online utility DB2 uses a shadow data set and will switch from the current to the shadow after running the utility. So DB2 will switch the instance qualifier between I and J when you run online REORG and CHECK utilities. The numbers after the instance qualifier can change if you use clones. Although this is not the place for a comprehensive discussion of cloning let’s skim the surface to understand what happens to this portion of the data set name. Basically, cloning creates a table with the exact same attributes as a table that already exists, except that it has no data. The clone is created using the ALTER TABLE SQL statement with the ADD CLONE parameter and the clone table is created in the same table space as the existing table... except in a different VSAM data set. The base table starts with I0001 in the data set name; the clone will be I0002. This can change because the SQL EXCHANGE statement flips the VSAM data sets.

Next we have the data set number, which appropriately enough, is used when a page set requires multiple data sets. The z is usually an “A”, but it can be A, B, C, D, or E. For partitioned table spaces, the number is the partition number and A-E is used for partitions in excess of 999. So partition 1 would be A001 and partition 1,000 would be B001, and so on. For simple or segmented table spaces, the data set number starts at 001 and is incremented by one as the page set grows past the maximum size of 2GB.

These are the most basic storage “things” that you will need to know as you manage DB2 for z/OS storage.

Important DB2 for z/OS Storage Issues

Although storage management can be an afterthought for the DBA it really shouldn’t be. According to Gartner, Inc. the cost of managing storage is 4 to 10 times the initial cost of storage acquisition. And the growth rate for disk storage was 37% for the years 1996 through 2007. So storage issues are vitally important and unless it is managed appropriately it can be very costly. And unmanaged DB2 storage can result in system outages, which is the last thing any DBA wants to have happen, isn’t it?

Even so, it is common for storage-related issues to be relegated to the backburner by DBAs. Let’s face it, most robust mainframe organizations have an entire unit dedicated to storage management and administration. And the DBA has enough to contend with without adding storage tasks to the list. But DBAs and storage administrators are concerned about different things – and that is the way it should be.

Refer to Figure 3. The DBA has a database-centric focus, whereas the storage administrator will focus on storage issues for the entire shop, focusing on devices and data sets. But the storage folks are not DB2 experts, nor should they be. Likewise, most DBAs are not storage experts. Making matters worse is that these two groups rarely communicate well.

Figure 3. DBA versus Storage Administration

So there is a gap between Database Administration, Storage Administration and Capacity Planning. Information is available to DBAs from various sources including RUNSTATS, STOSPACE, real-time statistics (RTS), DB2 Catalog, and so on, but the details are scattered all over the place and it can be difficult to gain a complete, accurate, and up-to-date picture. And any historical view into DB2 storage usage has to be managed manually.

Think about your environment for a moment and then reflect on whether or not you can easily answer the following questions:

- Do all of my databases have sufficient allocation to satisfy business requirements?
- Why is DB2 storage growing when our business is not?
- Am I wasting any storage?
- When will more storage be required?
- How much additional storage is needed?
- What needs to be done to align the additional storage with the DBMS?

Without a tool to capture, integrate, and manage information about your DB2 storage infrastructure answering these questions can be quite difficult.

A Little Bit About Modern Disk Arrays

Mainframe disk, or DASD, is usually equated to a 3380 or 3390. In other words, people think of physical hardware devices with a one-to-one relationship between a disk drive and a volume. The logical view is broken down as:

- Track size, or the number of bytes per track.
- Capacity, or the size of the device, in terms of number of tracks or gigabytes.
- Device address, which is a thread onto which I/O operations are serialized by the operating system

But the physical world is not the same as the logical anymore. Today's modern storage architecture uses disk arrays, or RAID. RAID stands for Redundant Array of Independent Disk; an array is the combination of two or more physical disk storage devices in a single logical device or multiple logical devices. The array is perceived by the system to be a single disk device. RAID can offer big benefits in terms of availability because the disks are typically hot-swappable, meaning that a drive can be replaced while the array is up and running. There are many levels of RAID technology, each delivering different levels of fault-tolerance and performance. A nice educational overview of the various levels of RAID is offered by Advanced Computer & Network Corporation at http://www.acnc.com/04_00.html. But what about mainframe disk arrays?

The RAMAC Virtual Array (RVA) came first for the mainframe and it was based on virtual disks that emulated 3380s and 3390s. The RVA dynamically maps functional volumes to physical drives. This mapping structure is contained in a series of tables stored in the RVA control unit. RVA was OEM'ed from Storage Technology Corp. (STK) which is now part of Oracle.

The ESS (Enterprise Storage System), also known as Shark, followed when the STK OEM agreement expired. The ESS is scalable from 420GB to 55.9 TB. It offers improved performance over RAMAC, especially for prefetch and analytical queries.

And the latest and greatest IBM mainframe disk technology is the DS8000, which employs virtualized disk. It adds functionality for storage pool striping, thin provisioning, and quick initialization, among other innovations. Its capacity scales linearly from 1.1 TB up to 192 TB (up to 320 TB with turbo models).

Of course, IBM is not the only game in town for mainframe storage. EMC, Hewlett Packard, Hitachi, and Sun also offer modern disk arrays for the mainframe.

Cache Versus Buffer

Modern disk arrays cache data, but so does DB2. In the old days we used to disable disk cache for DB2, but no longer. There are two DSNZPARMs you should be aware of that can be setup to properly control disk caching for DB2 data sets.

First up is the SEQCACH parameter. The original meaning of this parameter, for 3390 DASD, was whether DB2 I/O should bypass the disk cache, but the meaning is different now because you do not want to bypass the cache on modern storage arrays. There are two options:

BY- The disk will perform Sequential Detection
PASS

SEQ Creates an explicit Prefetch request; the recommendation is to use this setting for improved performance.

The second DSNZPARM is SEQPRES, which is similar to SEQCACH, but for DB2 LOAD and REORG utilities. If set to YES the Cache is more likely to retain pages for subsequent update, particularly when processing NPIs. The general recommendation is to set to SEQCACH to YES.

Keep in mind, too, that your storage administrators should be aware that DB2 buffering may cause DB2 data to use the disk cache differently than other non-database data sets. But that doesn't mean that DB2 is not benefiting from disk caching.

A Little Bit About DFSMS

DFSMS is IBM's Data Facility Storage Management System. It offers data management, backup and HSM software from IBM mainframes, combining backup, copy, HSM and device driver routines into a single package. So DFSMS is actually multiple products; it is a suite of data and storage management offerings:-

DFSMSdfp Data Facility Product - provides the logical and physical input and output for z/OS storage, it keeps track of all data and programs managed within z/OS, and it provides data access both for native z/OS applications and other platforms.

DFSMSdss This is a priced optional feature. It is a DASD data and space management tool for moving and copying data.

DF- Hierarchical Storage Manager - a priced optional feature for managing low-activity and
SMSshm inactive data. It provides backup, recovery, migration, and space management functions.

DFSMSr- Removable Media Manager - a priced optional feature for managing removable media re-
mm sources (e.g. IBM's Virtual Tape Server).

DFSMSUvs Transactional VSAM Services – is another priced optional feature that enables batch jobs and CICS online transactions to update shared VSAM data sets concurrently.

But we are not going to delve into all of these various components. What we are most interested in is how DB2 can benefit from DFSMS. Using DFSMS, a DB2 DBA can simplify the interaction of DB2 database creation and storage specification. It can deliver:-

- Simplified data allocation
- Improved allocation control
- Improved performance management

- Automated disk space management
- Improved data availability management
- Simplified data movement

DFSMS has the necessary flexibility to support everything DB2 DBAs may want to accomplish in terms of data set placement and design. In this day and age there is no reason to not take advantage of DFSMS for DB2 data sets. However, to achieve a successful implementation, an agreement between the storage administrator and the DB2 administrator is required so that they can together establish an environment that satisfies both their objectives.

SMS Data Classes, Storage Classes, Management Classes, and Storage Groups can be used along with SMS ACS (Automatic Class Selection) routines to set up and automate DB2 data set allocation and placement. SMS Storage Groups contains volumes that satisfy the service requirements of the data sets allocated to them. They can handle more than one type of data. Separate Storage Groups should be defined for production table spaces, active logs, other production data, and non-production data. ACS routines assign data sets to SMS storage classes. For example, indexes can be assigned to one SMS storage class and table spaces to a different SMS storage class. For DB2 Storage Groups using SMS the volume list is set to '*'.

As of DB2 9, DATACLAS, MGMTCLAS, and STORCLAS can be specified in DB2 Storage Groups, and if so, then the VOLUMES clause can be omitted. If the VOLUMES clause is omitted, the volume selection is controlled by SMS. Keep in mind, though, that when processing the VOLUMES, DATACLAS, MGMTCLAS, or STORCLAS clauses, DB2 does not check the existence of the volumes or classes or determine the types of devices that are identified or if SMS is active. Later, when the storage group allocates data sets, the list of volumes is passed in the specified order to Data Facilities (DFSMSdfp).

Basically, using SMS with DB2 is the sane thing to do these days because the new disk architectures, with concepts like log structured files and with cache in the gigabyte sizes, render conventional database design rules based on data set placement less important. In most cases, placement is not an issue, and when it is, SMS classes and ACS routines can be setup to take care of things.

What About Extents?

Some folks think “With RAID/modern storage devices and new DB2 and z/OS features, extents are no longer anything to worry about.” But this is not exactly true. For one thing, the latest extent management features only work with SMS-managed data sets, so if you are still using user-managed data sets then all of the old rules apply!

For SMS-managed data set you can have up to 123 extents on each of 59 volumes. So as of z/OS 1.7, the limit is 7,257 extents for a data set instead of the 255 we’ve been used to for some time. Again though, to enable this requires DFSMS (modify the DFSMS Data Class to set the Extent Constraint Removal to YES).

Extent consolidation also requires SMS-managed STOGROUPs. If a new extent is adjacent to old, they will be merged together automatically. This can result in some extents being larger than the PRIQTY or SECQTY specification(s). Note that this feature was introduced in z/OS 1.5.

OK, so what if everything is SMS-controlled? Then extents don’t matter, right? Well, not really. Even then it is possible for extents to impact performance. Each extent on a disk file has different control blocks controlling access. This means that elapsed time can increase if there is heavy insert activity. For other types of processing (read and update) the number of extents really does not impact on performance.

At any rate, things are not like the olden days where you had to regularly monitor extents and clean them up all the time by reorganizing your table spaces and index spaces. Oh, we want to clean those extents

up periodically, but storage administrators have other methods of reducing extents that perhaps can be quicker and/or easier, for example.

- DFSMSHsm MIGRATE and RECALL functions
- DFSMSdss COPY or DUMP and RESTORE functions
- DEFRAG with the CONSOLIDATE keyword
- Other products: e.g. Real Time Defrag

As of V8, DB2 can allocate sliding scale secondary extents. This is enabled by setting MGEXTSZ DSNZ-PARM to YES. Note that the default is NO for V8, but changed to YES automatically when you upgrade to DB2 9. With sliding scale extents the extent sizes allocated gradually increase. DB2 uses a sliding scale for secondary extent allocations of table spaces and indexes when:

- You do not specify a value for the SECQTY option of a CREATE TABLESPACE or CREATE INDEX statement
- You specify a value of -1 for the SECQTY option of an ALTER TABLESPACE or ALTER INDEX statement.

Otherwise, DB2 uses the SECQTY value for secondary extent allocations, if one is explicitly specified (and the SECQTY value is larger than the value that is derived from the sliding scale algorithm). If the table space or index space has a SECQTY greater than 0, the primary space allocation of each subsequent data set is the larger of the SECQTY setting and the value that is derived from a sliding scale algorithm.

Without going into all of the gory details, sliding scale extent allocation can help to reduce the number of extents for your Db2 objects as they grow in size over time. And it can help when you do not have a firm understanding of how your data will grow over time.

Another Thing to Think About

Every now and then I hear from a DB2 DBA complaining about index growth. They'll say something like this: "My index keeps growing and growing and taking additional extents, even after deleting data from the base table. What is going on?"

Well, deleted index keys are not physically deleted, but marked as pseudo-deleted. This can cause an index to grow even as the table data remains at relatively the same level. And it can result in a high CPU cost for index scans because DB2 scans all of the entries, even the pseudo-deleted ones. For this reason, you should keep an eye on tables with a lot of delete activity and periodically reorganize their indexes. You can track pseudo-deleted index keys in SYSINDEXPART if using RUNSTATS or SYSINDEXSPACESTATS if using real time statistics. Consider reorganizing these indexes when pct of pseudo-deleted entries is:

- Greater than 10% for non Data Sharing
- Greater than 5% for Data Sharing

Best Practices

In general, you should adopt best practices for managing your DB2-related storage. Keep up-to-date on DB2/storage functionality and adopt new practices in the latest releases of DB2.

It is a good idea to perform regular and proactive monitoring. Examples of things you should be tracking include:

1. Space display and monitoring for your entire DB2 system.
2. Space display and monitoring of individual DB2 databases.
3. Space display and monitoring of all of your table spaces and indexes.
4. Display and monitoring of the Storage Groups and the associated volumes of a DB2 system. (Data, Workfile, Image Copies, Logs, Archives, Sort/Work etc.)
5. The ability to display all VSAM data sets for all table spaces and indexes (Used, Allocated, Primary and Secondary Quantity, Volumes) and monitoring of the extents for each.
6. Display of the Page Sets of table spaces and indexes that reach their maximum size and maximum number of data sets.
7. Tracking of image copy backup data sets, including the intelligent HSM migration of same. You should be able to reduce backups by track which are not used for local recovery (to CURRENT), as well as data sets older than the last full image copy (including dual and remote backups).
8. Managing the deletion of Image copy backup datasets that are no longer needed because of DROP, DROP/CREATE or MODIFY TABLESPACE ('orphaned', not listed in SYSIBM.SYSCOPY).

If possible, build alerts to inform you of problems, shortages, and potential errors. When possible, automate remediation tactics so that the alert tells you what happened, as well as what was done to correct the issue. Tools may be able to assist in automating reaction to shortages, potential errors, superfluous data sets, etc. For example, refer to Figures 4 and 5.

Figure 4. Automated Storage Alerts Figure 5. Tracking the Growth of DB2 Storage Space

Summary

Finally, there is a people issue that needs to be addressed. The DBA and the storage administrator will need to cooperate in order to facilitate proper database storage. Remember, other types of data that are not stored in the DBMS will need to be saved on disk, too. Databases use storage differently than non-database data. Indexing, partitioning, clustering, and separation of data will cause the database to require more storage, across more drives, and using cache differently than most storage administrators may anticipate.

DBAs need to work storage administrators to make sure that each understands the other domain. And to make sure that s/he has the storage information needed to properly administer DB2. The better the DBA communicates, and the better the relationship is between these two IT professionals, the better your database applications will perform. And that is what it is all really about, right?
