E-Rechnung – RPG, SQL und XML, 2. Teil

Eine E-Rechnung stellt Rechnungsdaten in einem strukturierten XML-Format bereit.

Dieser Artikel und die folgenden zu dieser Serie sollen Ihnen unverbindliche Möglichkeiten zeigen, wie Sie den Einstieg in die Erstellung von E-Rechnungen mit RPG und SQL selbstständig bewerkstelligen können. In den weiteren Folgeartikeln werden dann die speziellen Anforderungen an die E-Rechnungen und mögliche Lösungsansätze dargelegt.

In diesem Artikel wird für die Erstellung des XML-Codes die SQL-Funktion XMLGROUP verwendet.

Die skalare SQL-Funktion XMLGROUP

Die XMLGROUP-Funktion gibt einen XML-Wert mit einem einzelnen XQuery-Dokumentknoten zurück, der einen Elementknoten der obersten Ebene enthält. Hierbei handelt es sich um einen Aggregatausdruck, der ein XML-Dokument mit einfachem Stamm aus einer Gruppe von Zeilen zurückgibt, wobei jede Zeile einem Zeilenunterelement zugeordnet ist.

Für das folgende Beispiel verwende ich die Datei ORDERS aus der Beispielbibliothek von Db2 Web Query QWQCENT. Diese enthält Aufträge mit Auftragsnummer, Auftragsdatum, Artikelnummer und Positionswert. Es sollen nur die Aufträge verwendet werden, deren Auftragsdatum vor dem 02.01.2020 liegt.

```
-- Tabelle ORDERS in QWQCENT, nur bestimmte  Spalten
Select Ordernumber, Orderdate, Productnumber, Linetotal
   From Qwqcent.Orders
   Where Orderdate < '2020-01-02'
   Order By Ordernumber;
```

SQL_01: Skript für Tabelle ORDERS in QWQCENT

Diese Tabelle mit den angegebenen Spalten verwende ich nun für die skalare Funktion XMLGROUP:

```
- XMLGROUP und ORDERS
SELECT XMLGROUP(RTRIM(ORDERNumber) AS "Order_Number",
                        ORDERDATE AS "Order_Date",
                        LINETOTAL AS "Order_Amount",
                        RTRIM(ProductNumber) AS "Part_Number"
                order by Ordernumber   -- muss hier stehen
                OPTION ROW "Order" ROOT "New_Orders"
                as attributes)
    --      INTO :Outfile
                FROM qwqcent.ORDers
                 where Orderdate < '2020-01-02'
            --      order by Ordernumber  -- das geht nicht
                ;
```

SQL_02: Verwendung der XML-Funktion XMLGROUP

Damit erhält man das folgende Ergebnis:

-Formatted XML-

```xml
<?xml version="1.0" encoding="UTF-8"?>
<New_Orders>
   <Order Order_Number="48724" Order_Date="2020-01-01" Order_Amount="399.00" Part_Number="1003" />
   <Order Order_Number="48724" Order_Date="2020-01-01" Order_Amount="129.00" Part_Number="1051" />
   <Order Order_Number="48724" Order_Date="2020-01-01" Order_Amount="1499.00" Part_Number="2004" />
   <Order Order_Number="48724" Order_Date="2020-01-01" Order_Amount="529.00" Part_Number="4001" />
   <Order Order_Number="48724" Order_Date="2020-01-01" Order_Amount="1999.00" Part_Number="4014" />
   <Order Order_Number="68651" Order_Date="2020-01-01" Order_Amount="17556.00" Part_Number="1011" />
   <Order Order_Number="68651" Order_Date="2020-01-01" Order_Amount="5676.00" Part_Number="1051" />
   <Order Order_Number="68651" Order_Date="2020-01-01" Order_Amount="65956.00" Part_Number="2004" />
   <Order Order_Number="68651" Order_Date="2020-01-01" Order_Amount="239925.00" Part_Number="4015" />
   <Order Order_Number="68651" Order_Date="2020-01-01" Order_Amount="15525.00" Part_Number="5004" />
   <Order Order_Number="74310" Order_Date="2020-01-01" Order_Amount="187906.00" Part_Number="1014" />
   <Order Order_Number="74310" Order_Date="2020-01-01" Order_Amount="31255.00" Part_Number="1034" />
   <Order Order_Number="74310" Order_Date="2020-01-01" Order_Amount="481256.00" Part_Number="2003" />
   <Order Order_Number="74310" Order_Date="2020-01-01" Order_Amount="217355.00" Part_Number="2004" />
   <Order Order_Number="74310" Order_Date="2020-01-01" Order_Amount="21755.00" Part_Number="3001" />
</New_Orders>
```

Abbildung_01: Aufbereitete Ergebnismenge von Funktion XMLGROUP

Hinweis:
Die Ergebnismenge des Skriptes von SQL_01 habe ich mit dem folgenden kostenlosen Online-Tool XML-gerecht aufbereitet:

https://www.freeformatter.com/xml-formatter.html

Weitere Informationen zur skalaren Funktion XMLGROUP finden Sie unter:

https://www.ibm.com/docs/en/ias?topic=functions-xmlgroup

Wie kann diese SQL-Funktion nun in RPG verwendet werden?

Das RPG-Programm XML20 mit XMLGROUP

Das folgende Beispielprogramm liest Daten aus der Datei ORDERS in der Bibliothek QWQCENT und konvertiert diese mit Hilfe der skalaren SQL-Funktion XMLGROUP im XML-Format ins IFS. Die Verbindung zwischen den unterschiedlichen Welten wird mit Hilfe des RPG-Datentyps SQLTYPE hergestellt.

SQLTYPE-Variablen sind etwas ungewöhnlich. Der SQL Pre-Compiler konvertiert diese Datentypen in entsprechende RPG-Formate. Wenn Sie eine SQLTYPE-Variable definieren, erstellt das Programm eine spezielle Datenstruktur. Grundsätzlich wird der von Ihnen definierte Name an ein Suffix angehängt, um die Unterfelder zu identifizieren.

XML Host-Variable und XML Datei-Referenzen

Für die Definition mit SQLTYPE gibt es nur bestimmte SQL-Datentypen, die verwendet werden können. Die folgenden beiden Abbildungen zeigen eine Übersicht über die möglichen Datentypen.

Description	SQL type	Minimum length	Maximum length
XML BLOB	XML_BLOB	1	16,773,100
XML CLOB	XML_CLOB	1	16,773,100
XML Double Byte CLOB	XML_DBCLOB	1	8,386,550
XML locator	XML_LOCATOR	> 16M	

Abbildung_02: Definition von XML Host-Variablen

Description	SQL type	Minimum length	Maximum length
XML Binary LOB file	XML_BLOB_FILE	N/A	
XML Character LOB file	XML_CLOB_FILE	N/A	
XML Double Byte Character LOB file	XML_DBCLOB_FILE	N/A	

Abbildung_03: Definition von XML Datei-Referenzen

Der wesentliche Unterschied zwischen den beiden Arten von SQL-Datentypen ist, dass bei der Definition als XML Datei-Referenz keine Dateilänge angegeben werden muss.

Weiter interessante Infos zu SQLTYPE finden Sie unter
https://www.rpgpgm.com/2018/09/sql-type-variables-in-rpg.html

Im folgenden Programm XML20 wird SQLTYPE mit der Definition als Datei-Referenz verwendet.

```
**free

dcl-s Outfile sqltype(xml_clob_File) ;
dcl-s Path varchar(100) inz('/home/TB/xmlfile.xml') ;

dcl-s StartDate date(*iso) inz(d'2020-01-02') ;

exec sql SET OPTION COMMIT=*NONE,CLOSQLCSR=*ENDMOD,DATFMT=*ISO ;

//-------------------------------------------------------------------------------
----//
//
  clear Outfile ;
  Outfile_Name = %trimr(Path) ;               // IFS-Pfad und Name der XML-File
  Outfile_NL = %len(%trimr(Outfile_Name)) ;   // Länge des IFS-Pfads+Name der IFS-
File
//Outfile_FO = SQFRD  ;                       // Dateioperation - Read
//Outfile_FO = SQFCRT ;                       // Dateioperation - Create
  Outfile_FO = SQFOVR ;                       // Dateioperation - Override
//Outfile_FO = SQFAPP ;                       // Dateioperation - Append

//-------------------------------------------------------------------------------
----//

exec sql SELECT XMLGROUP(RTRIM(ORDERNumber) AS "Order_Number",
                                 ORDERDATE AS "Order_Date",
                                 LINETOTAL AS "Order_Amount",
                                 RTRIM(ProductNumber) AS "Part_Number"
                         order by Ordernumber
                         OPTION ROW "Order" ROOT "New_Orders"
                         as attributes)
            INTO :Outfile
                         FROM qwqcent.ORDers
                          where Orderdate < :StartDate
                          ;

  *inlr = *on ;
  Return;
```

RPG_01: Beispielprogramm XML20

Weitere Infos zu SQLTYPE

Durch die Definition der Variablen *Outfile* mit

dcl-s Outfile sqltype(XML_BLOB_FILE) ;

werden durch den Pre-Compiler die folgenden Definitionen generiert:

```
DCL-DS OUTFILE;
  OUTFILE_NL UNS(10);
  OUTFILE_DL UNS(10);
  OUTFILE_FO UNS(10);
  OUTFILE_NAME CHAR(255) CCSID(*JOBRUNMIX);
END-DS OUTFILE;
```

RPG_02: Generierte Datenstruktur

```
DCL-C SQFRD CONST(2);
DCL-C SQFCRT CONST(8);
DCL-C SQFOVR CONST(16);
DCL-C SQFAPP CONST(32);
```

RPG_03: Generierte Definition von Konstanten

Was bedeuten diese Definitionen?

- OUTFILE_NAME: Name der IFS-Datei einschließlich des vollständigen Pfads
- OUTFILE_NL: Länge des Namens der IFS-Datei
- OUTFILE_FO: Dateioperation
- OUTFILE_DL: wird nicht verwendet

Für die Variable OUTFILE_FO – Dateioperation können die folgenden vor-definierten Werte verwendet werden:

- SQFRD: Read only – nur Lesen
- SQCRT: Create – Erstellen
- SQFOVR: Override - Überschreiben
- SQFAPP: Append - Hinzufügen

Als Ergebnis von XML20 wird eine XML-Datei im angegebenen IFS-Ordner erstellt:

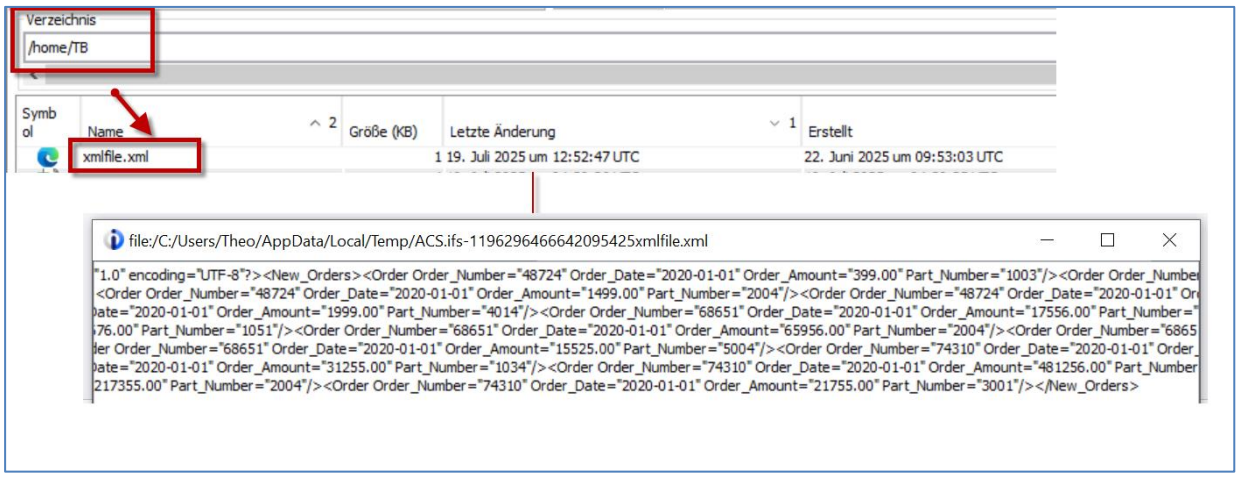

Abbildung_04: XML-File in IFS

Nach der Aufbereitung mit dem Online-Tool ergibt sich dieselbe Darstellung wie in Abbildung_01.

Vorschau:
Im nächsten Artikel werden weitere Möglichkeiten gezeigt, wie XML-Dateien mit RPG und SQL erstellt werden können.

Bis dahin wünsche ich Ihnen weiterhin viel Spaß beim Vermehren Ihrer Fertigkeiten.

Den Autor Theo Bär erreichen Sie unter EDV-Beratung Theo Bär - Ringmauerweg 1 - 69250 Schönau - Tel. (+49) 6228 912 630 - e-Mail info@edv-baer.com