

SQL your XML

David Andruchuk
Sr. Architect
Computer Systems Design Associates, Inc.
February 12, 2009

What can *i* do.....*i* can do XML

SQL your XML

FUD = Fear, Uncertainty and Doubt

- FUD is generally a strategic attempt to influence public perception by disseminating negative information designed to undermine the credibility of their beliefs
- The term originated to describe disinformation tactics in the computer hardware industry and has since been used more broadly
- FUD is a manifestation of the appeal to fear.

SQL your XML

FUD = Fear, Uncertainty and Doubt

- Productivity is key in this economic climate, and showing your ability to do more with less, or in our case with nothing, will certainly be noticed and appreciated
- Technology is integral to business operations, and the efficiencies that business leaders seek to wring from their operations are often available only via technology

SQL your XML

- **So what does FUD have to do with SQL and XML on the *i*?**
- **What is pureXML?**
- **Why does my *i* not support pureXML?**

SQL your XML

So what does FUD have to do with SQL and XML on the *i*?

Recall that FUD is the strategic attempt to influence public perception.

Since pureXML is not supported in the current version of DB2 for the *i*, and since it cannot process XML similar to other platform versions of DB2, it is the public perception that you cannot then use SQL to process your XML

SQL your XML

What is pureXML?

pureXML is the native XML storage feature in the IBM DB2 data server.

pureXML provides query languages, storage technologies, indexing technologies, and other features to support XML data.

The word pure in pureXML was chosen to indicate that DB2 natively stores and natively processes XML data in its inherent hierarchical structure, as opposed to treating XML data as plain text or converting it into a relational format

SQL your XML

Why does my *i* not support pureXML?

pureXML was first included in the DB2 9 for Linux, Unix, and Windows release, which was codenamed Viper, in June 2006

DB2 9 is a hybrid data server—it offers data management for traditional relational data, as well as providing native XML data management

DB2 for the *i* does not yet support native XML data management due to the tight integration with i5/OS.....yet

SQL your XML

FUD is CRUD!

SQL your XML

SQL & RPG to the rescue

SQL your XML

SQL's User Defined Table Functions (UDTF)

V5R2 DB2 for i5/OS was enhanced to support user-defined table functions (UDTF).

A UDTF can be used from SQL interfaces to access non-relational data sources such as S/36 files and stream files residing in the IFS.

The UDTF enhancement complemented the user-defined scalar function (UDF) capability added way back in V4R4 of OS/400.

A user-defined function is classified as scalar meaning that the function can only return a single value (or output parameter). That's where UDTFs come into play with their ability to return multiple values in the form of a table (or result set).

SQL your XML

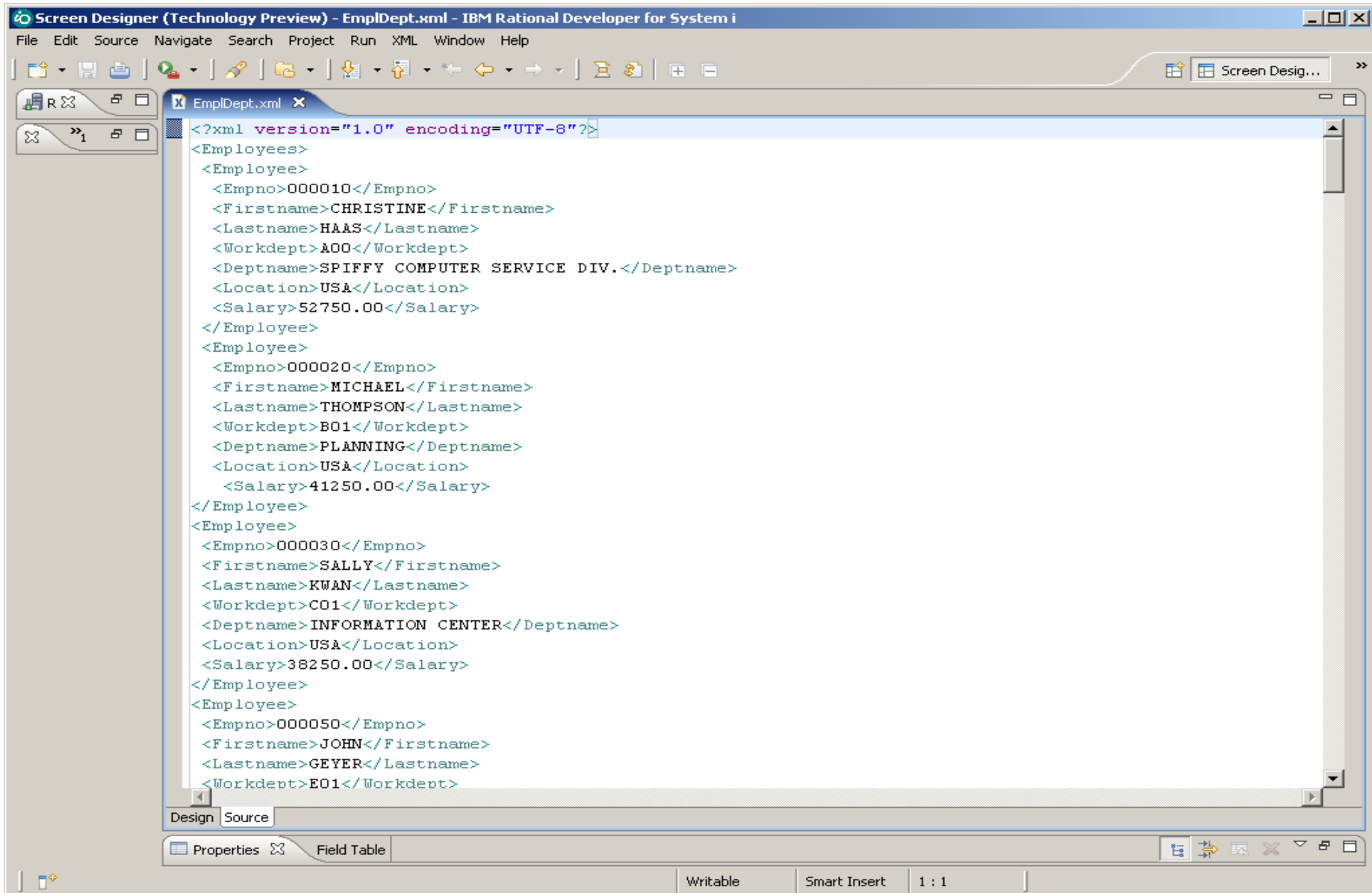
RPG's XML-INTO Built in Function (BIF)

V5R4 ILE RPG for i5/OS was enhanced to facilitate decomposition of an XML document into data structures defined in the program using a non-validating parser.

The new operation codes and built-in functions makes the decomposition of the XML data and population of data structure fields automatic and efficient without the developer having to manipulate storage to get the XML data in to workable form.

The complexity has been removed from the XML decomposition process and the developer now can just concentrate on business logic required to implement the XML data.

SQL your XML



The screenshot displays the IBM Rational Developer for System i interface. The main window, titled "EmplDept.xml", shows an XML document with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<Employees>
  <Employee>
    <Empno>000010</Empno>
    <Firstname>CHRISTINE</Firstname>
    <Lastname>HAAS</Lastname>
    <Workdept>A00</Workdept>
    <Deptname>SPIFFY COMPUTER SERVICE DIV.</Deptname>
    <Location>USA</Location>
    <Salary>52750.00</Salary>
  </Employee>
  <Employee>
    <Empno>000020</Empno>
    <Firstname>MICHAEL</Firstname>
    <Lastname>THOMPSON</Lastname>
    <Workdept>B01</Workdept>
    <Deptname>PLANNING</Deptname>
    <Location>USA</Location>
    <Salary>41250.00</Salary>
  </Employee>
  <Employee>
    <Empno>000030</Empno>
    <Firstname>SALLY</Firstname>
    <Lastname>KWAN</Lastname>
    <Workdept>C01</Workdept>
    <Deptname>INFORMATION CENTER</Deptname>
    <Location>USA</Location>
    <Salary>38250.00</Salary>
  </Employee>
  <Employee>
    <Empno>000050</Empno>
    <Firstname>JOHN</Firstname>
    <Lastname>GEYER</Lastname>
    <Workdept>E01</Workdept>
  </Employee>
</Employees>
```

The interface includes a menu bar (File, Edit, Source, Navigate, Search, Project, Run, XML, Window, Help), a toolbar, and a status bar at the bottom with options like "Writable", "Smart Insert", and "1 : 1".

SQL your XML

Coding the SQL UDTF

```
-- *-----*
-- * Create Instructions: *
-- * * *
-- * RUNSQLSTM SRCFILE(QSQLSRC) SRCMBR(SQLXMLUDTF) *
-- * COMMIT(*NONE) *
-- *-----*
```

Create Function SQLXMLUDTF

```
( i_empno VARCHAR(6)
)
```

Returns Table

```
( empno varchar(6),
  firstname varchar(14),
  lastname varchar(17),
  workdept varchar(3),
  departname varchar(38),
  location varchar(16),
  salary dec(9,2)
)
```

External Name SQLXMLUDTF

Language RPGLE

No SQL

Parameter Style DB2SQL

No Scratchpad

No Final Call

Deterministic

Disallow Parallel

No External Action

Not Fenced

Cardinality 10;

SQL your XML

Coding the RPGLE program

```
// *-----*
// * Create Instructions: *
// * * *
// * CRTBNDRPG PGM(SQLXMLUDTF) *
// *-----*
h DftActGrp(*No) actgrp(*CALLER) alwnull(*usrctl)
h option(*srcstmt: *nodebugio)

// SQL UDTF prototype
d SQLXMLUDTF pr extpgm('SQLXMLUDTF')

// SQL UDTF passed argument(s)
d arg_empno 6a varying

// SQL UDTF returned result(s)
d res_empno 6a varying
d res_firstname 14a varying
d res_lastname 17a varying
d res_workdept 3a varying
d res_departmentname 38a varying
d res_location 16a varying
d res_salary 9p 2

// SQL Argument Null Indicator(s)
d arg_empno_null... 5i 0
// SQL Result Null Indicator(s)
d res_empno_null... 5i 0
d res_firstname_null... 5i 0
d res_lastname_null... 5i 0
d res_workdept_null... 5i 0
d res_departmentname_null... 5i 0
d res_location_null... 5i 0
d res_salary_null... 5i 0
```

SQL your XML

Coding the RPGLE program (continued)

```
// SQL Function Parameters
d SQL_State          5
d Function_Name     517
d Specific_Name     128
d Msg_Text          70 varying
d CallType          5i 0

// SQL UDTF Procedure interface
d SQLXMLUDTF        pi

// SQL UDTF passed argument(s)
d arg_empno         6a varying

// SQL UDTF returned result(s)
d res_empno         6a varying
d res_firstname     14a varying
d res_lastname      17a varying
d res_workdept      3a varying
d res_department    38a varying
d res_location      16a varying
d res_salary        9p 2

// SQL Argument Null Indicator(s)
d arg_empno_null... 5i 0
// SQL Result Null Indicator(s)
d res_empno_null... 5i 0
d res_firstname_null... 5i 0
d res_lastname_null... 5i 0
d res_workdept_null... 5i 0
d res_department_null... 5i 0
d res_location_null... 5i 0
d res_salary_null... 5i 0
```

SQL your XML

Coding the RPGLE program (continued)

```
// SQL Function Parameters
```

```
d SQL_State          5  
d Function_Name     517  
d Specific_Name     128  
d Msg_Text          70  varying  
d CallType          5i 0
```

```
// SQL Function Prototypes
```

```
d $close            pr  
d $fetch            pr  
d $open             pr  
d setEndOfTable    pr  
d setMore           pr
```

```
// Get EmplDept parsed XML data from memory prototype
```

```
d getEmplDept      PR
```

```
// Parse XML document prototype
```

```
d ParseXML         PR
```

SQL your XML

Coding the RPGLE program (continued)

```
// Data structure(s)/column(s) for XML document parsing
d Employees      DS          QUALIFIED          Employees XML Elem
d Employee      DS          likeds(Employee)
d               DS          dim(100)

d Employee      DS          QUALIFIED          Employee XML Elem
d Empno         DS          6a
d Firstname     DS          12a
d Lastname      DS          15a
d Workdept      DS          3a
d Deptname      DS          36a
d Location      DS          16a
d Salary        DS          9p 2

// Work field(s)
D FirstFetch    S           N
d i1            S           5 0 inz(0)          index
d IFSXMLDoc     C           const('/tempxml/doc/EmplDept.xml') IFS XML Document
d Null          C           const(x'00')        null constant
D NullData     S           N
d XMLDoc       S           256                  XML document name
```

SQL your XML

Coding the RPGLE program (continued)

```
// *-----*
// * Mainline calculations *
// *-----*
/free

SQL_State = '00000';           // reset SQL state

Monitor;                       // b-monitor 4 errs

// Select processing by SQL Call Type
select;
  when CallType = -1;          // open cursor
    callp $open();
  when CallType = *zero;      // fetch from cursor
    callp $fetch();
  when CallType = 1;          // close cursor
    callp $close();
endsl;

On-Error;                       // x-error occurred
*InLR=*On;
Endmon;                          // e-monitor 4 errs

// Return to caller
return;

/end-free
```

SQL your XML

Coding the RPGLE program (continued)

```
// *-----*  
// * CLOSE cursor procedure *  
// *-----*  
  
p $close      b  
  
d $close      pi  
  
/free  
  
  *InLR=*On;  
  FirstFetch = *on;  
  
// Clean up activation group  
  callp ceetrec( );  
  
// Exit procedure  
  return;  
  
/end-free  
  
p $close      e
```

SQL your XML

Coding the RPGLE program (continued)

```
// *-----*
// * FETCH cursor procedure *
// *-----*

p $fetch      b

d $fetch      pi

/free

if FirstFetch;

if not NullData;

// Loop through multi occurrence DS
i1 = i1 + 1;

// Execute get EmplDept record(s) procedure
callp getEmplDept();

else;
res_empno_null    = -1;
res_firstname_null = -1;
res_lastname_null = -1;
res_workdept_null = -1;
res_departname_null = -1;
res_location_null = -1;
res_salary_null   = -1;
endif;

else;
SQL_State = '02000';
endif;

// Exit procedure
return;

/end-free

p $fetch      e
```

SQL your XML

Coding the RPGLE program (continued)

```
// *-----*  
// * OPEN cursor procedure *  
// *-----*  
  
p $open      b  
  
d $open      pi  
  
/free  
  
    eval i1 = *zeros;  
    NullData = %Error;  
    FirstFetch = *on;  
  
// Parse XML document into memory  
    callp ParseXML( );  
  
// Exit procedure  
    return;  
  
/end-free  
  
p $open      e
```

SQL your XML

Coding the RPGLE program (continued)

```
// *-----*
// * getEmplDept procedure *
// *-----*

p getEmplDept b

d getEmplDept pi

/free

// Select processing by one or all employees
select; // b-select by emp#

    when arg_empno <> *blanks; // single empno #

dow i1 < 101;

// Check if EmpNo array element populated and same as passed argument
// -If populated, check if specific employee # passed by user

if Employees.Employee(i1).empno <> *blanks and // b-EmpNo check
    Employees.Employee(i1).empno = arg_empno;

    eval res_EmpNo = Employees.Employee(i1).empno;
    eval res_FirstName = Employees.Employee(i1).Firstname;
    eval res_LastName = Employees.Employee(i1).Lastname;
    eval res_WorkDept = Employees.Employee(i1).Workdept;
    eval res_DeptName = Employees.Employee(i1).Deptname;
    eval res_Location = Employees.Employee(i1).Location;
    eval res_Salary = Employees.Employee(i1).Salary;

    FirstFetch = *off;
    i1 = 101;
endif; // b-EmpNo check

i1 = i1 + 1;
enddo;
```

SQL your XML

Coding the RPGLE program (continued)

```
        when arg_empno = *blanks;                // all employees

// Check if EmpNo array element populated
// -If populated, check if specific employee # passed by user

        if Employees.Employee(i1).empno <> *blanks;        // b-EmpNo <> *blank

            eval res_EmpNo = Employees.Employee(i1).empno;
            eval res_FirstName = Employees.Employee(i1).Firstname;
            eval res_LastName = Employees.Employee(i1).Lastname;
            eval res_WorkDept = Employees.Employee(i1).Workdept;
            eval res_DepartName = Employees.Employee(i1).Deptname;
            eval res_Location = Employees.Employee(i1).Location;
            eval res_Salary = Employees.Employee(i1).Salary;

        else;                                        // x-Empno <> *blank
            FirstFetch = *off;
            callp setEndOfTable();
            return;

        endif;                                        // e-Empno <> *blank
    endsl;                                        // e-select by emp#

// Exit procedure
return;

/end-free

p getEmplDept   e
```

SQL your XML

Coding the RPGLE program (continued)

```
// *-----*
// * ParseXML procedure *
// *-----*

p ParseXML    b

d ParseXML    pi

/free

// Set work variables
eval XMLDocument = %trim(IFSXMLDocument) + Null;

// Decompose XML into column(s)/data structure(s)
xml-into Employees %XML(XMLDocument :
    'doc=file +
    allowextra=yes +
    allowmissing=yes +
    case=any');

/end-free

p ParseXML    e
```

SQL your XML

Coding the RPGLE program (continued)

```
// *-----*
// * SetEndOfTable procedure *
// *-----*

p setEndOfTable b

d setEndOfTable pi

/free

// Set return SQL state
SQL_State = '02000';

// Clear result set field(s)
clear res_empno ;
clear res_firstname ;
clear res_lastname ;
clear res_workdept ;
clear res_departmentname;
clear res_location ;
clear res_salary ;

// Set result set field(s) null indicators
res_empno_null = -1;
res_firstname_null = -1;
res_lastname_null = -1;
res_workdept_null = -1;
res_departmentname_null = -1;
res_location_null = -1;
res_salary_null = -1;

// Set return Message Text
Msg_Text = 'End of Table';

// Exit procedure
return;

/end-free

p setEndOfTable e
```

SQL your XML

Coding the RPGLE program (continued)

```
// *-----*
// * SetMore procedure *
// *-----*

p setMore    b

d setMore    pi

/free

// Set return SQL state
  SQL_State = '00000';

// Set result set field(s) null indicators
  res_empno_null    = *zeros;
  res_firstname_null = *zeros;
  res_lastname_null = *zeros;
  res_workdept_null = *zeros;
  res_departmentname_null = *zeros;
  res_location_null = *zeros;
  res_salary_null   = *zeros;

// Set return Message Text
  Msg_Text = *blanks;

// Exit procedure
  return;

/end-free

p setMore    e
```

SQL your XML

Notes about the RPGLE code

This example is a simple demonstration and does not take in to account any error checking or large XML documents. The simple XML document and DS built to parse it can only process up to 100 records.

Use the XML-HANDLER OpCode instead of the XML-INTO to be able to handle unknown sizes of XML documents.

While this example shows XML parsing, a UDTF program can also do any DB processing, calculations, specific business logic, etc. which makes the UDTF a very versatile addition to the SQL toolbox when a result set is desired.

SQL your XML

Executing the code

The final piece of the example is demonstrating how to call the SQLXMLUDTF table function.

As you may recall, we allowed for the Employee # to be passed from the SQL interface to the UDTF RPGLE program.

The following two examples will show the SQL statements required to execute the select by a single Employee # or for all Employee data.

You can execute the SQL Select statement in any SQL interface such as STRSQL, SQL Procedure Language, JAVA, RPG, etc.

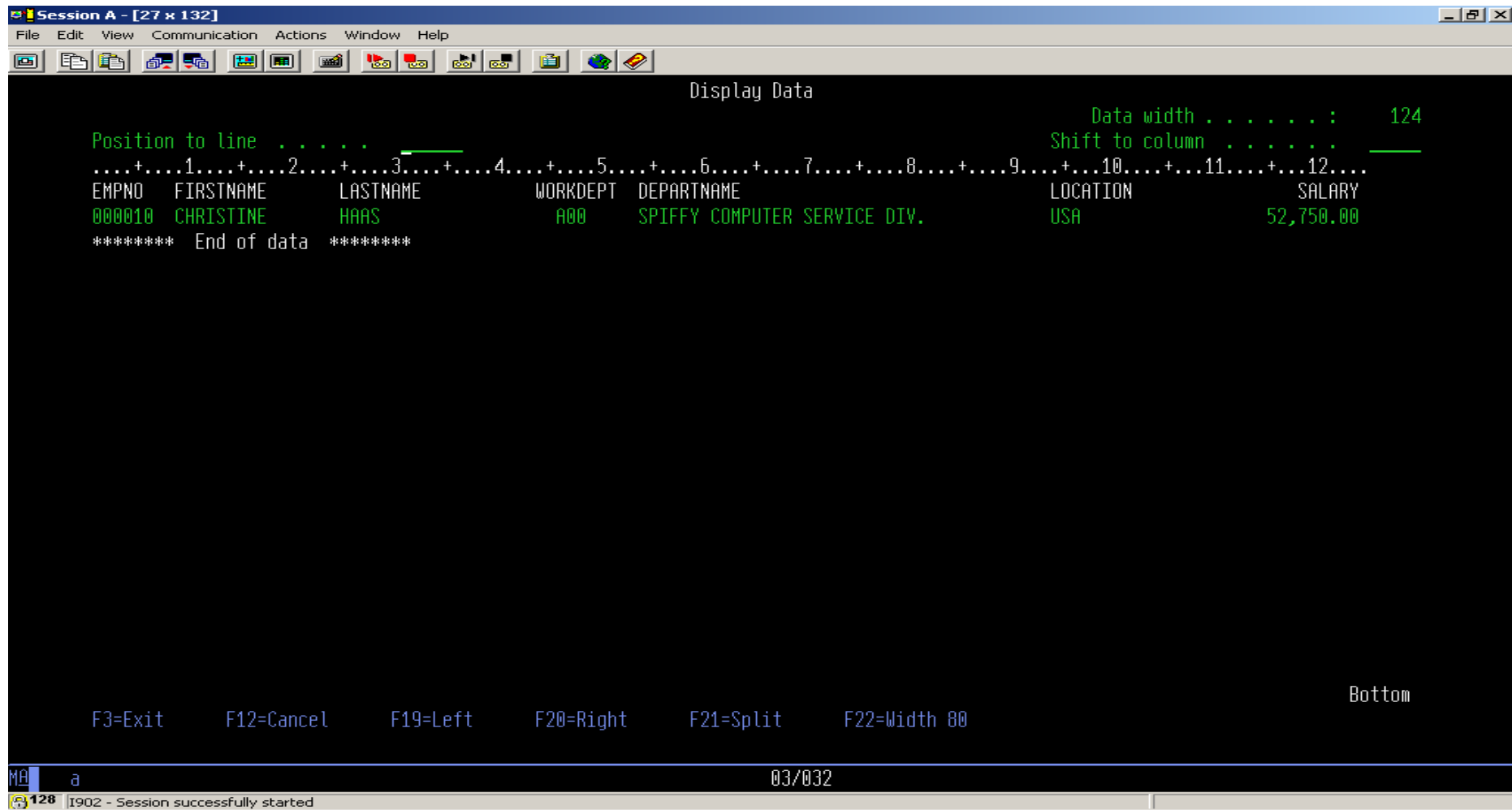
In addition, once the table is returned you can perform any SQL processing as if it was a DB2 table that existed on disk. You can order by, sub select using the where clause, insert data into another table.....

SQL your XML

Executing the code (continued)

Single Employee # retrieve:

```
select * from table(sqlxmludtf('000010')) x
```



The screenshot shows a SQL*Plus session window titled "Session A - [27 x 132]". The window contains the following text:

```
Display Data
```

Position to line	Data width	124				
Shift to column						
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...10...+...11...+...12...						
EMPNO	FIRSTNAME	LASTNAME	WORKDEPT	DEPARTNAME	LOCATION	SALARY
000010	CHRISTINE	HAAS	A00	SPIFFY COMPUTER SERVICE DIV.	USA	52,750.00
*****	End of data	*****				

At the bottom of the window, there are several function key shortcuts: F3=Exit, F12=Cancel, F19=Left, F20=Right, F21=Split, F22=Width 80, and Bottom. The status bar at the bottom shows "MA a" and "03/032".

SQL your XML

Executing the code (continued)

Multiple Employee # retrieve:

```
select * from table(sqlxmludtf(' ')) x
```

Display Data

Position to line
Data width : 124
Shift to column

EMPNO	FIRSTNAME	LASTNAME	WORKDEPT	DEPARTMENTNAME	LOCATION	SALARY
000010	CHRISTINE	HAAS	A00	SPIFFY COMPUTER SERVICE DIV.	USA	52,750.00
000020	MICHAEL	THOMPSON	B01	PLANNING	USA	41,250.00
000030	SALLY	KWAN	C01	INFORMATION CENTER	USA	38,250.00
000050	JOHN	GEYER	E01	SUPPORT SERVICE'S	USA	40,175.00
000060	IRVING	STERN	D11	MANUFACTURING SYSTEMS	USA	32,250.00
000070	EVA	PULASKI	D21	ADMINISTRATION SYSTEMS	USA	36,170.00
000090	EILEEN	HENDERSON	E11	OPERATIONS & SUPPORT	USA	29,750.00
000100	THEODORE	SPENSER	E21	SOFTWARE SUPPORT	USA	26,150.00
000110	VICENZO	LUCCHESSE	A00	SPIFFY COMPUTER SERVICE DIV.	USA	46,500.00
000120	SEAN	O'CONNELL	A00	SPIFFY COMPUTER SERVICE DIV.	USA	29,250.00
000130	DELORES	QUINTANA	C01	INFORMATION CENTER	USA	23,800.00
000140	HEATHER	NICHOLLS	C01	INFORMATION CENTER	USA	28,420.00
000150	BRUCE	ADAMSON	D11	MANUFACTURING SYSTEMS	USA	25,280.00
000160	ELIZABETH	PIANKA	D11	MANUFACTURING SYSTEMS	USA	22,250.00
000170	MASATOSHI	YOSHIMURA	D11	MANUFACTURING SYSTEMS	USA	24,680.00
000180	MARILYN	SCOUTTEN	D11	MANUFACTURING SYSTEMS	USA	21,340.00
000190	JAMES	WALKER	D11	MANUFACTURING SYSTEMS	USA	20,450.00
000200	DAVID	BROWN	D11	MANUFACTURING SYSTEMS	USA	27,740.00
000210	WILLIAM	JONES	D11	MANUFACTURING SYSTEMS	USA	18,270.00

F3=Exit F12=Cancel F19=Left F20=Right F21=Split F22=Width 80 More...

MA a 03/032

128 | 1902 - Session successfully started

SQL your XML

Questions?