January 2009

Unloading fact data from SAP BI 7.0 using the Queryview Web Service and C#

Authors

Hermann Daeubler, Senior Program Manager, Microsoft Corp.,
hermannd@microsoft.com
Juergen Daiberl, Senior Technical Evangelist, D&PE, Microsoft Corp.,
jdaiberl@microsoft.com

Summary

This paper describes how to unload fact data from SAP NetWeaver BI using C# and the "queryview" Web service. The paper contains a walk-through of how to use the queryview Web service within a C# console program in order to unload fact data from SAP NetWeaver BI without the need of any additional driver or library installation.

Applies to

- Microsoft .NET
- Microsoft Business Intelligence (Microsoft BI)
- Microsoft SQL Server Integration Services
- Microsoft SQL Server Reporting Services
- SAP NetWeaver 04s
- SAP NetWeaver BI
- SAP NetWeaver BI hierarchies
- SAP NetWeaver BI queryview Web service
- C#

Keywords

SAP NetWeaver, SAP NetWeaver BI, XMLA, C#, Microsoft BI, Microsoft SQL Server, SSRS, SSIS, SAP NetWeaver BI Hierarchies, SAP NetWeaver BI queryview

Level of difficulty

IT Management, Technical Architects, Technical Consultants, Developers

Contact

This document is provided to you by Microsoft Corp. Please check the SAP interoperability area at www.microsoft.com/sap and the .NET interoperability area in the SAP Developer Network (http://sdn.sap.com) for any updates or further information.

Microsoft

Microsoft®

Contents

Introduction

In certain customer scenarios it is necessary to unload fact data from SAP BI in order to integrate with a Microsoft BI solution. The key question is how could or should it be implemented?

Here is a list which shows five examples:

- Direct SAP NetWeaver BI database table access (unsupported by SAP and not recommended)
- "Mis-"using one of the SAP NetWeaver BI reporting APIs (e.g. XMLA or Ole DB for OLAP)
- 3rd-party tool
- BizTalk Adapter Pack
- SQL Server 2008 Integration Services (upcoming SSIS 2008 certification for SAP BI 7.0)

This paper will focus on using the queryview Web service in SAP NetWeaver BI. It's an interface which allows you to trigger an existing SAP BI query and to retrieve the result set.

SAP BI queries are used for reporting and not for extracting mass data. The so-called 'Open Hub Service' is the official interface in SAP NetWeaver BI to unload data into external targets like Microsoft's SQL Server 2008 Analysis Services (SSAS). It's still an option though to use a reporting API for this purpose as long as the amount of data is limited or the extract won't happen too often. The reporting APIs were not built for mass data extraction and don't offer an option for incremental unloads, therefore depending on the amount of data the usage of the reporting APIs will very likely lead to performance bottlenecks. Therefore it might work pretty well for master data which is very static. Extracting fact data from an InfoCube this way is questionable. Nevertheless this paper shows the basics about how to accomplish it with a C# console program and the queryview Web service. With small result sets it works fine, with bigger result sets you might experience performance problems as mentioned before. The queryview Web service has to be configured and activated on the SAP side. Once this is done it's possible to include the Web Service in a Microsoft Visual Studio C# project via the 'Add Web Reference' wizard. As the communication works over HTTP/SOAP there is no need for any additional driver or library installation.

The SAP BI queryview Web service can be used without specifying a MDX query as it would be necessary using XMLA. That's definitely an advantage for people who are not familiar with MDX. You just have to use the name of an existing SAP BI query which was created before via the SAP BI Query Designer. Calling the Web service will trigger the SAP BI query and retrieve the data in certain structures. The goal for this paper is to show an example how you could traverse the result set.

Sample

The sample gives an overview about what's necessary to retrieve some fact data (for the test case the SAP demo cube 0D_DECU is used) from SAP NetWeaver BI via C# by using the queryview Web service. The focus is on extracting data. There are no special considerations of security aspects or programming style. The SAP login e.g. is just hard-

coded. The idea was to create an output format in a console program which is similar to the list format of the SAP BI Query Monitor (transaction RSRT / RSRT2). The sample should be considered as a quick start guide and not as a ready-to-use tool in a production environment. The sample was tested using a few really simple SAP BI queries. It will very likely fail when adding hierarchies, working with structures or using complex attributes. But this is related to the output format. The structures will still return the appropriate data. In this case one would have to change the code of the sample. On the other side it's the question if it's really necessary to define a complex query for the main goal to just extract fact data. It should be enough to add some dimensions and restrictions / filters on them to unload a subset of an InfoCube. Hierarchies and master data could be unloaded independently as described in a separate paper.

Here is a short list of the steps required:

- Activate the queryview Web service on the SAP side (transaction shortcut is /nSICF). You can find a description on help.sap.com which explains how to do this – see screenshot in the walk-through section.
- Test the service by a using a browser. The output should look like on the screenshot in the walk-through section further down.
- Now use the SAP Query Designer and define a query on the corresponding InfoCube which extracts master data and key figures.
- You can test the newly created query in the SAP NetWeaver BI Query Monitor. The C# console sample program should return the same result as it can be seen in the SAP GUI (transaction shortcut /nRSRT2).
- Next step is to open a C# console program in Microsoft Visual Studio and to add the queryview service as a web reference. Enter the URL which was used for the browser test. It should then tell that it found the "query_view_dataService" with the method "GetQueryViewData".
- Within the C# program you have to set the login properties before it's possible to execute a SAP BI query. The sample code is shown further down.
- The big challenge with the sample code was to try to generate a similar output format as the SAP BI Query Monitor. What made it much more complicated than expected was especially :
    - the default result set includes intermediate sums
    - the numbers of objects per axis don't include attributes but the attributes have to be handled like objects to get a nice tabular format
- Running the C# console test program should finally deliver the same output as seen before in SAP NetWeaver BI.

This approach doesn't require any additional driver, library or 3rd-party tool. It's using a SAP NetWeaver BI Web service; therefore it also doesn't matter on which platform (operating system and database) the SAP NetWeaver BI system is running.

Walk-Through

Look on help.sap.com for the description of the querview Web service.

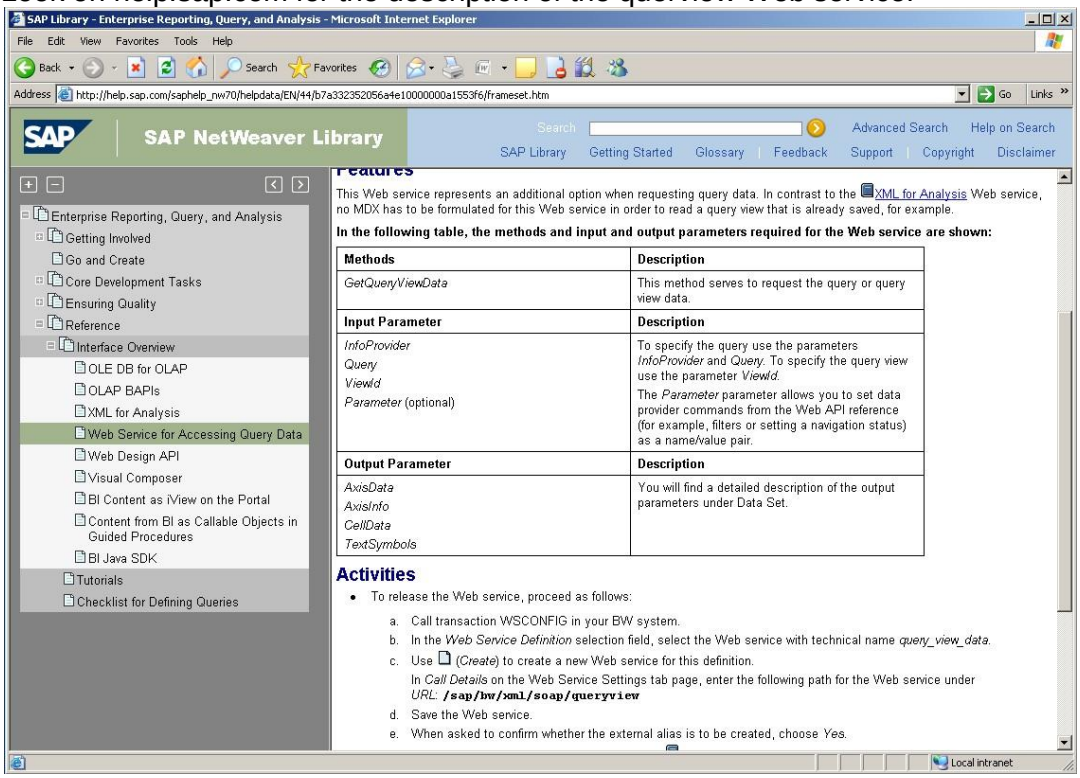

Figure 1 http://help.sap.com

Use transaction SICF to create an external alias for the QUERY_VIEW_DATA Web service.

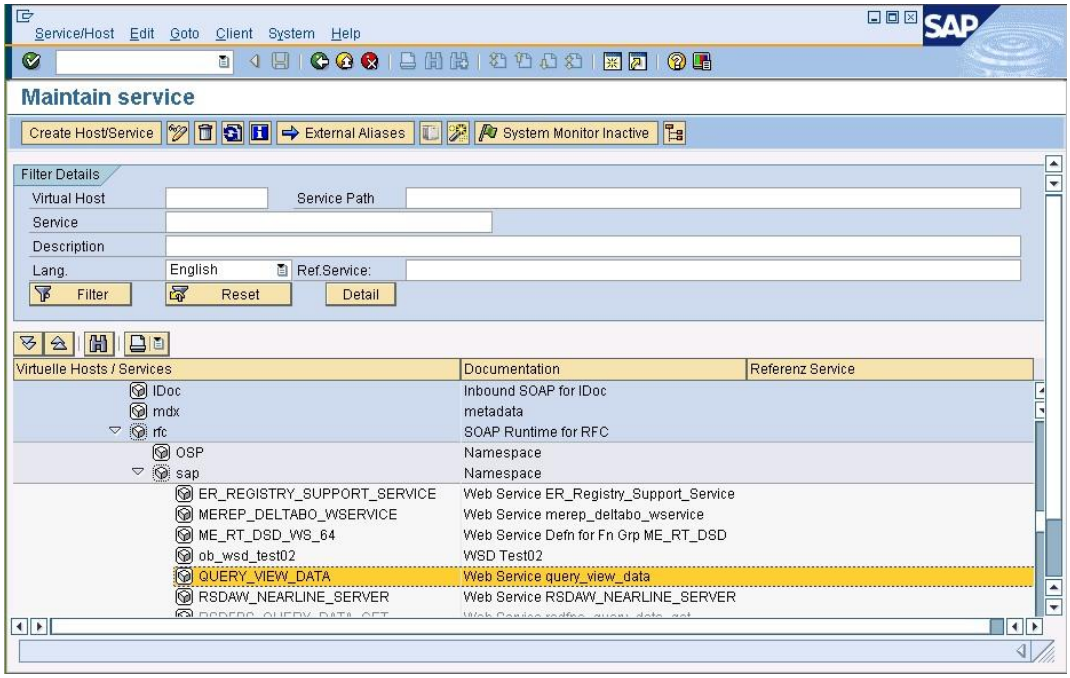

Figure 2 Transaction SICF in SAP GUI

There must be an external alias entry afterwards which can be used in an URL (/sap/bw/xml/soap/queryview).

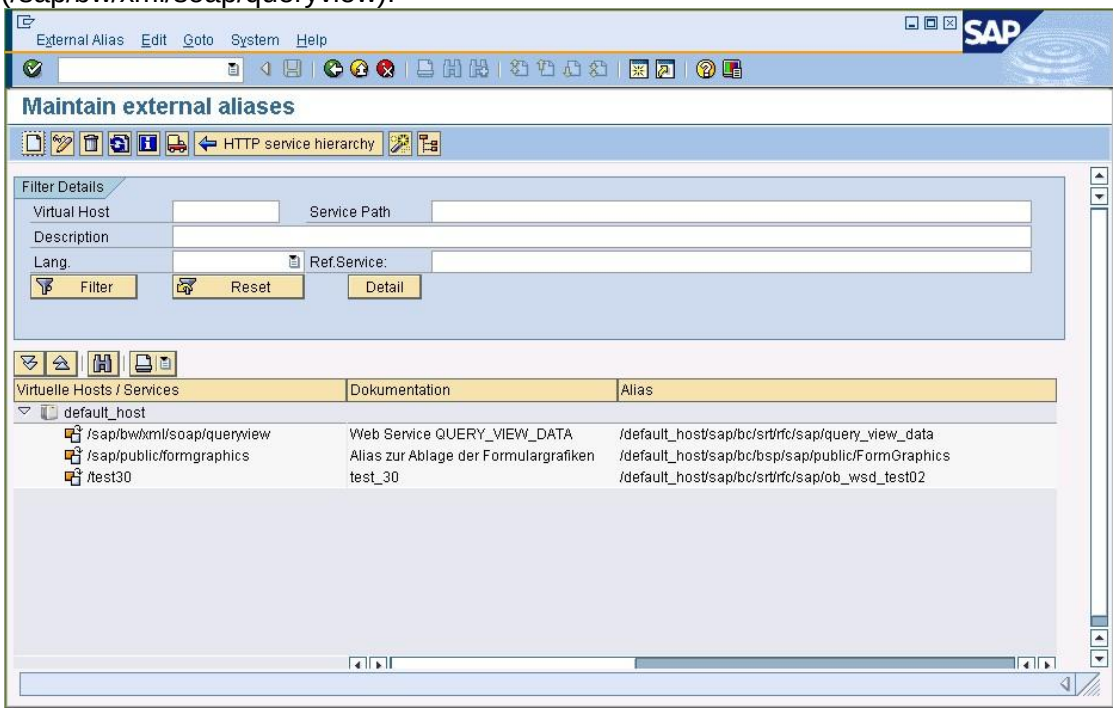

Figure 3 Maintaining external alias in SAP GUI

Test of the queryview Web service via a browser. The correct URL should return the same XML document (the WSDL) as seen on this screenshot.

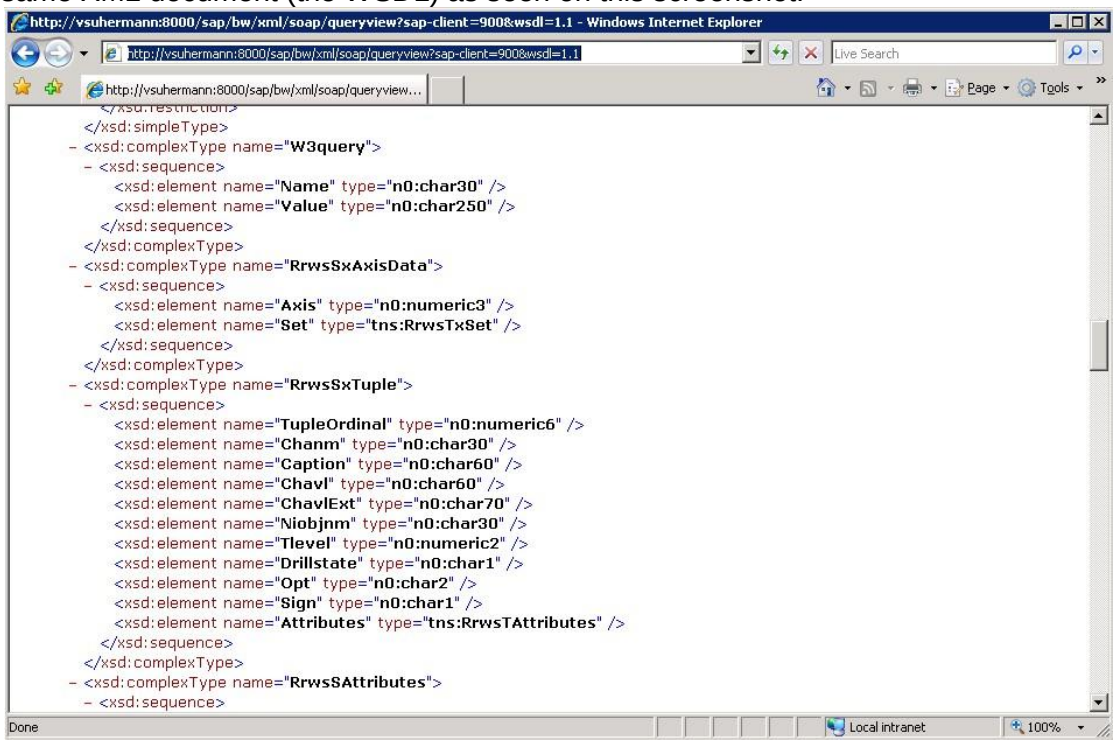

Figure 4 WSDL of queryview Web service in Browser

Use the URL from the browser test to add a web reference to your C# console program.

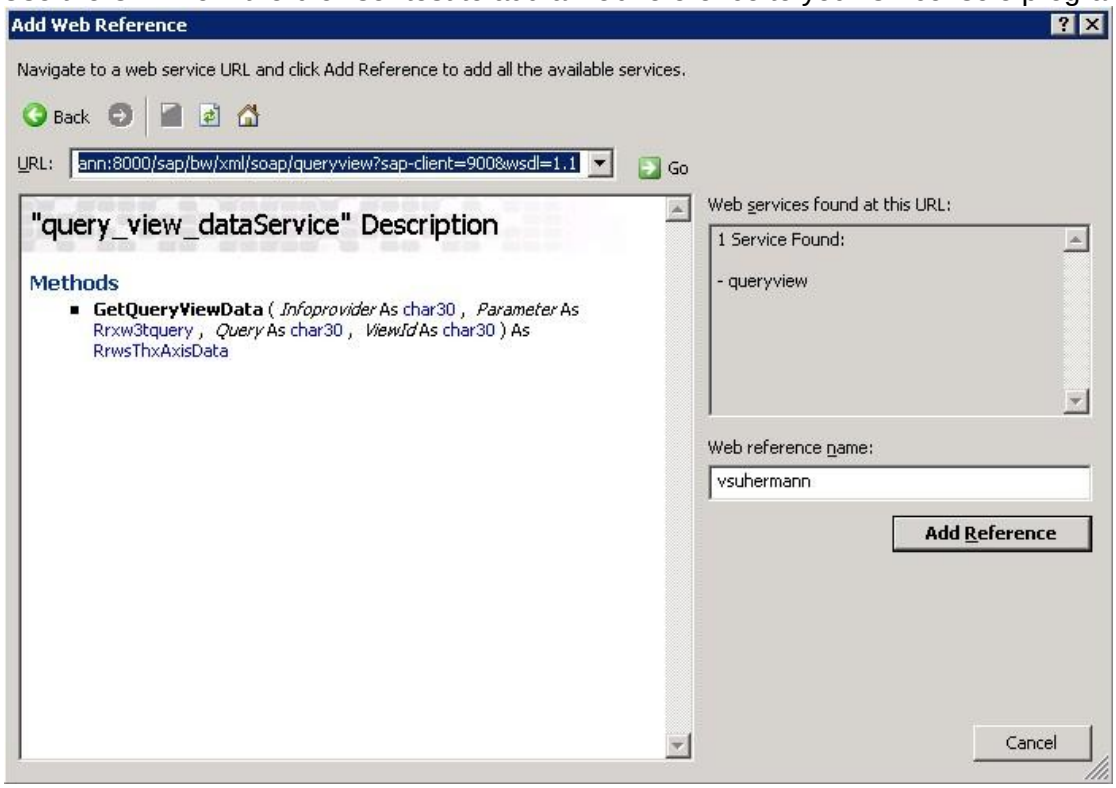

Figure 5 Add Web Reference Wizard in Microsoft Visual Studio

The SAP demo cube 0D_DECU was used for the sample.

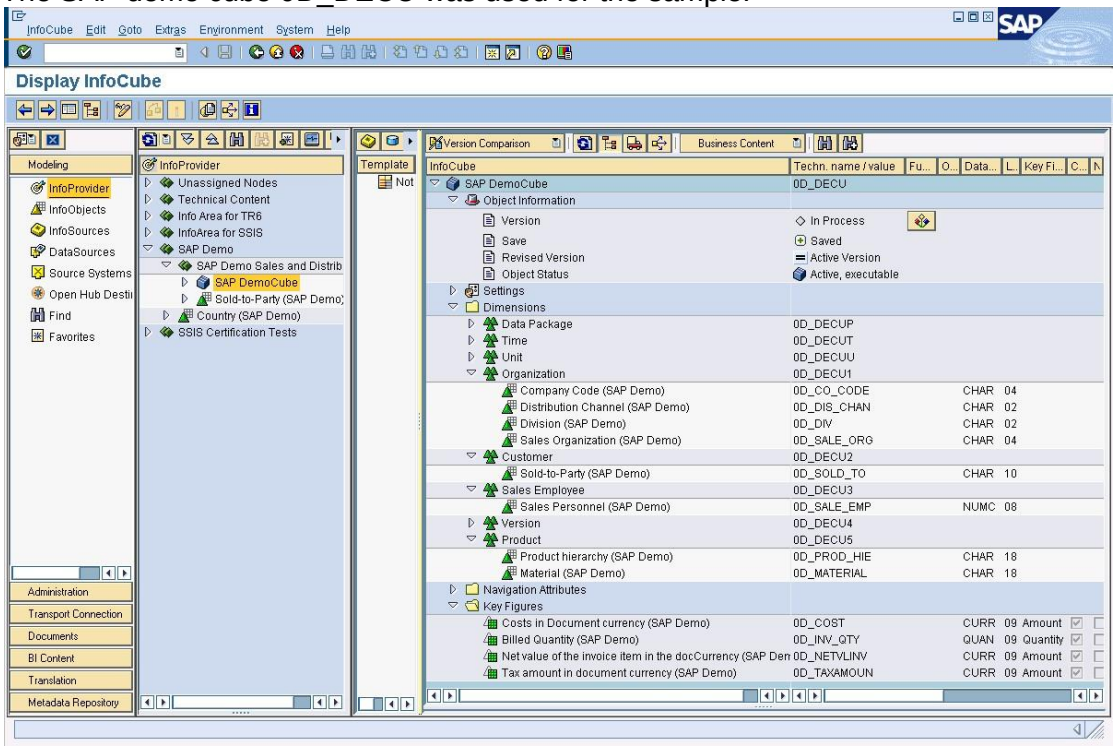

Figure 6 Demo Cube 0D_DECU in SAP GUI

Define a very simple first test query with master data on the rows and key figures on the columns. The second key figure is a calculated key figure (Net Value * 2).

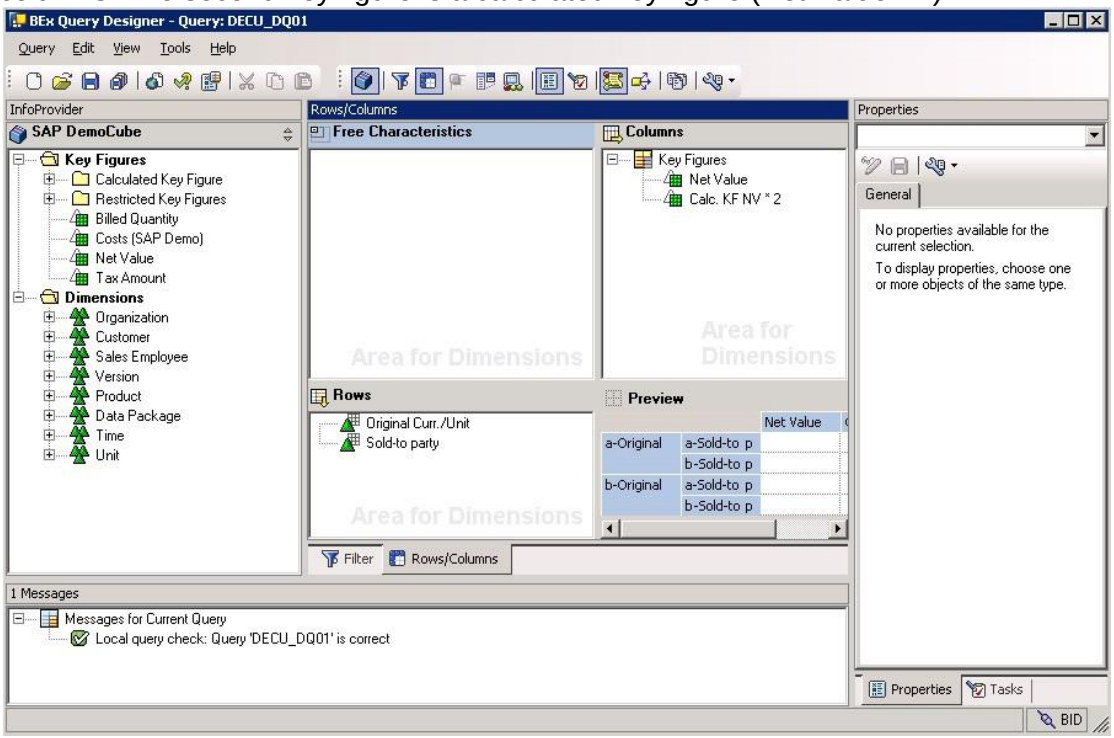

Figure 7 BEx Query Designer

Restrict the master data to just three different values to keep the console program output later on readable. Otherwise the result rows would be just too wide.

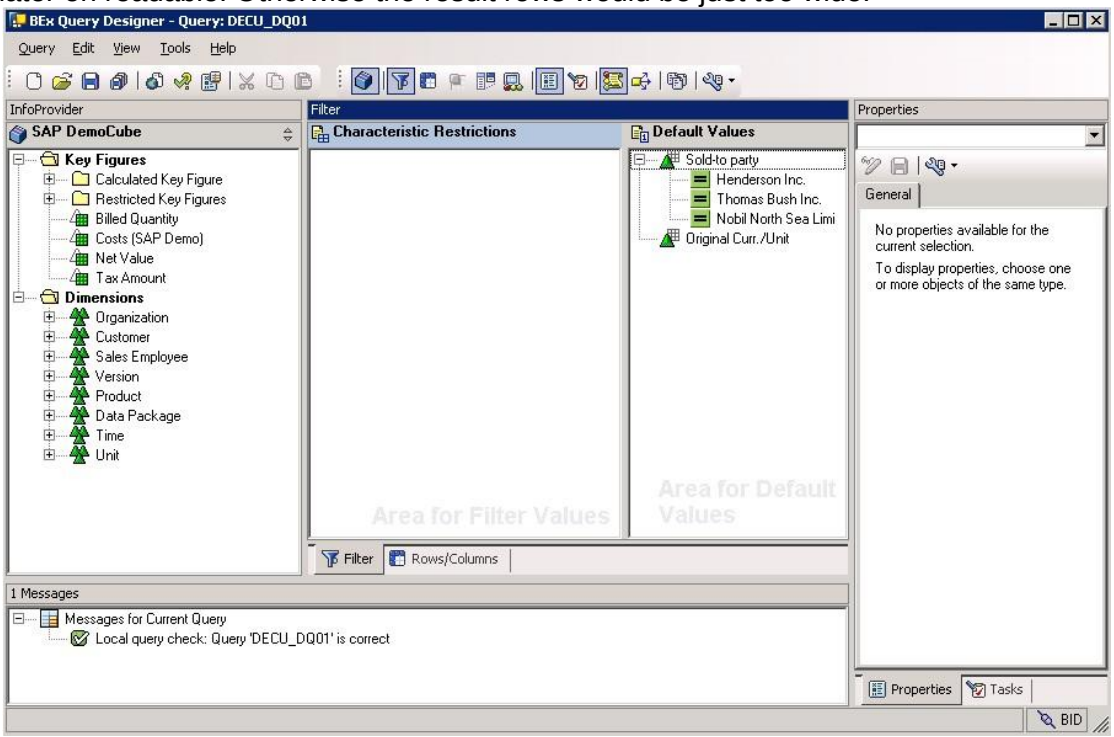

Figure 8 Restriction of Master Data to 3 values in BEx Designer

Run the first test query in the Query Monitor (shortcut /nRSRT2).

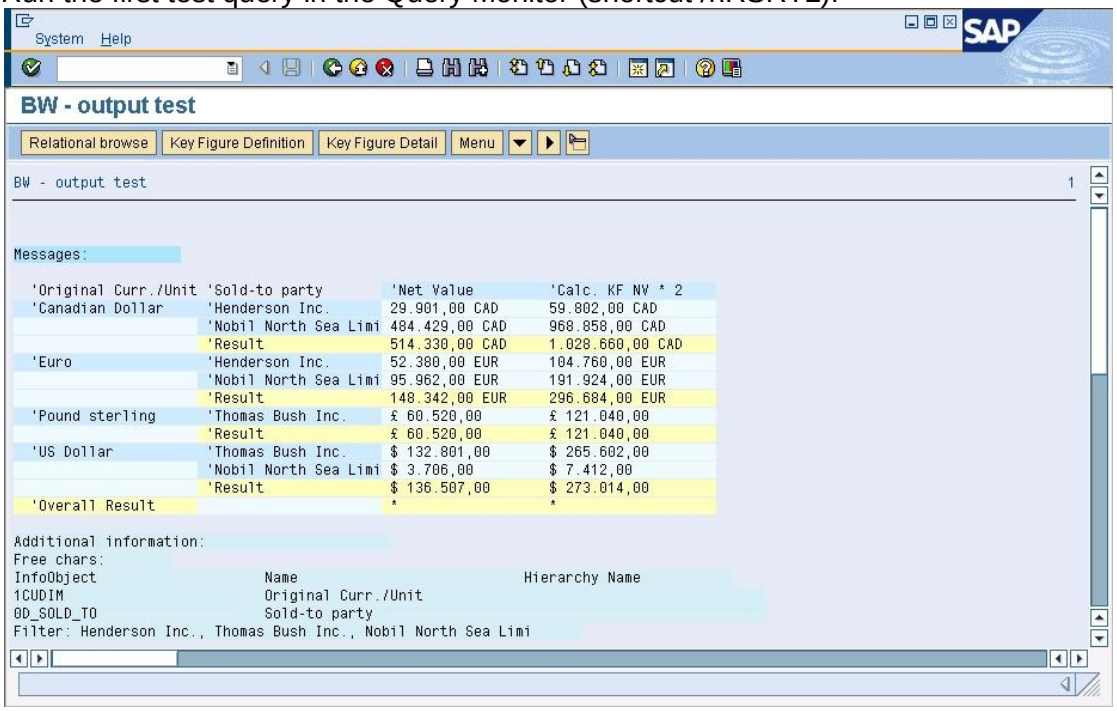

Figure 9 Transaction RSRT2 in SAP GUI

The output of the C# sample code should look very similar.

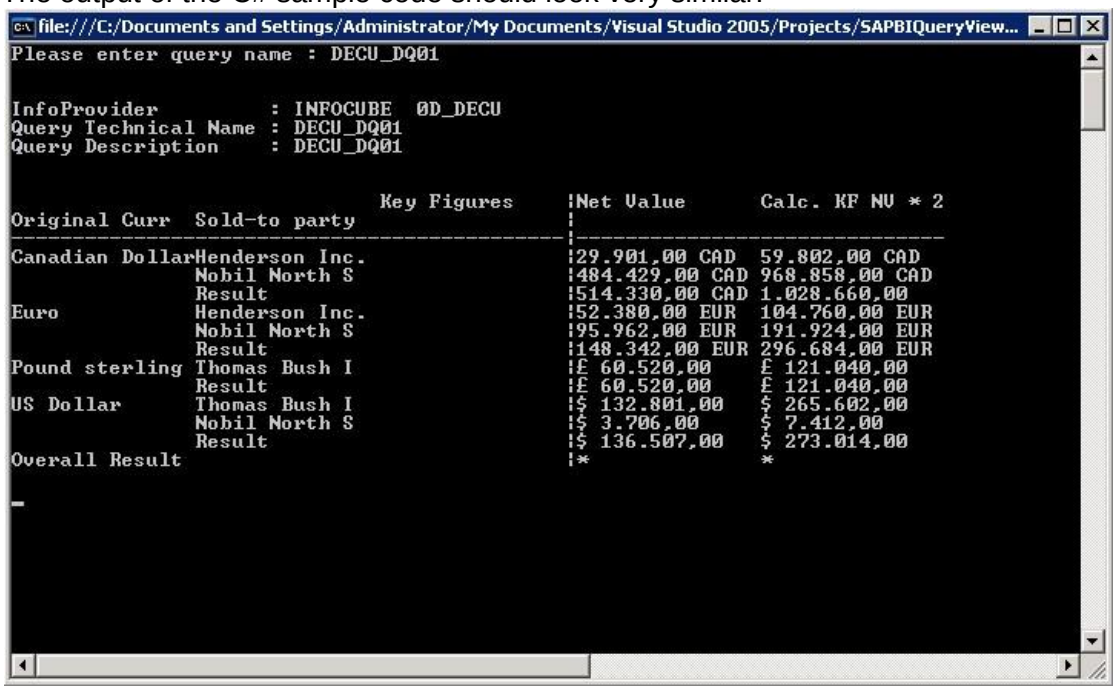

Figure 10 Output of Transaction in C# Program

Now create a second query and put the master data on the columns and the key figures on the rows.

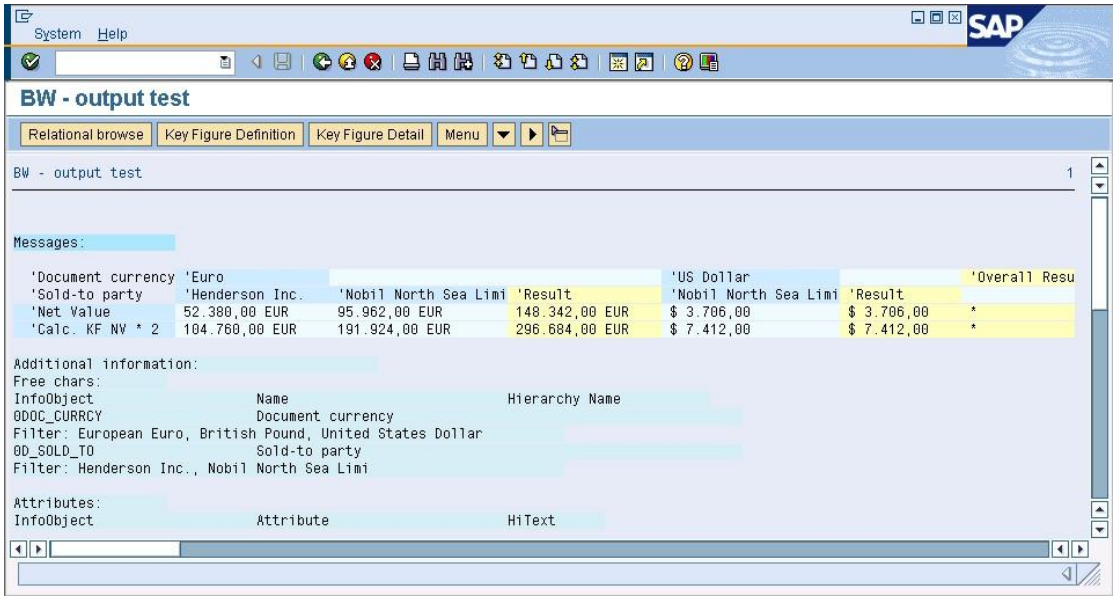

Figure 11 Second Query in SAP GUI

The C# sample code should still work as expected and should produce an output like shown in the next figure.

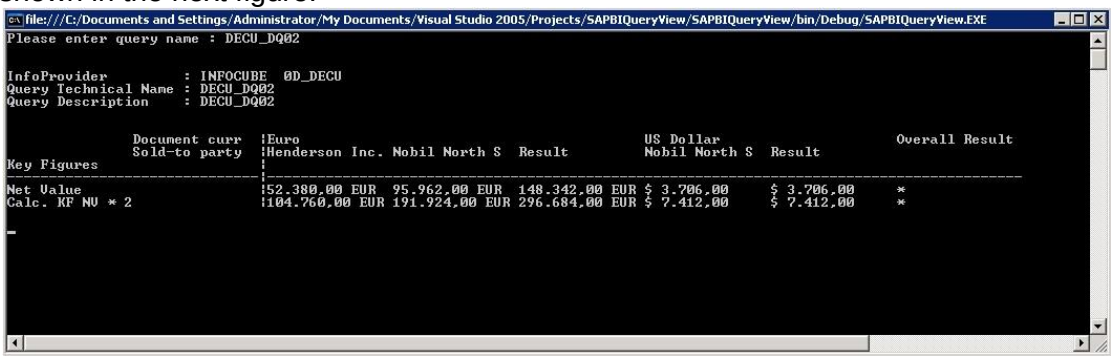

Figure 12 Output of Second Query with C# program

Now define a more complex query adding attributes to the master data and mix master data with key figures on the columns.

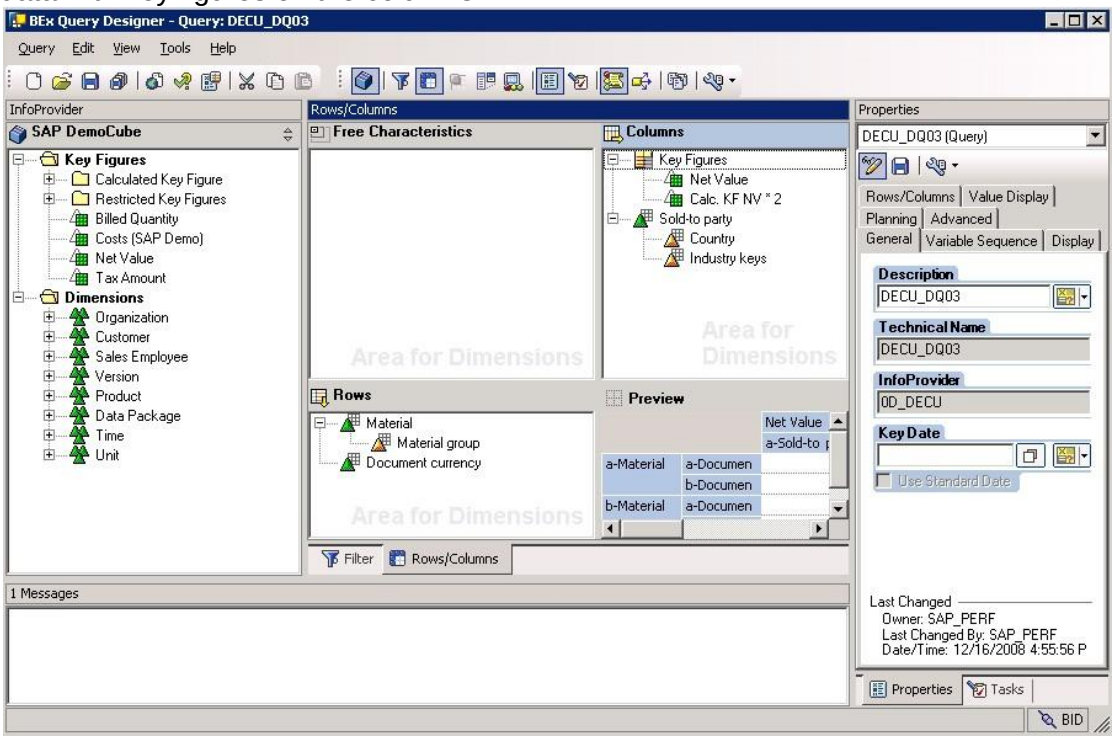

Figure 13 BEx Query Designer

The list format output of the SAP BI Query Monitor becomes more challenging.

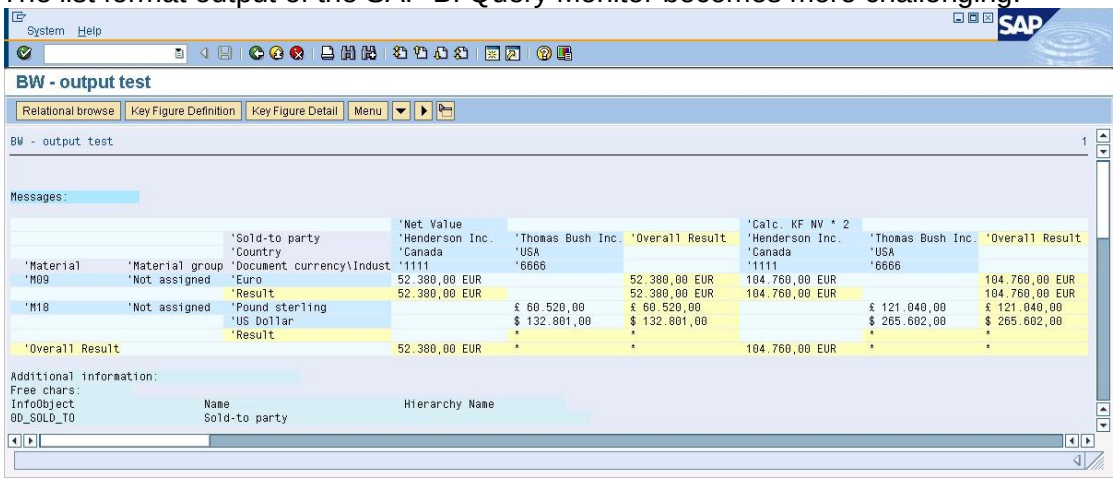

Figure 14 Query output in SAP GUI

Again the C# sample code is able to produce a similar output.

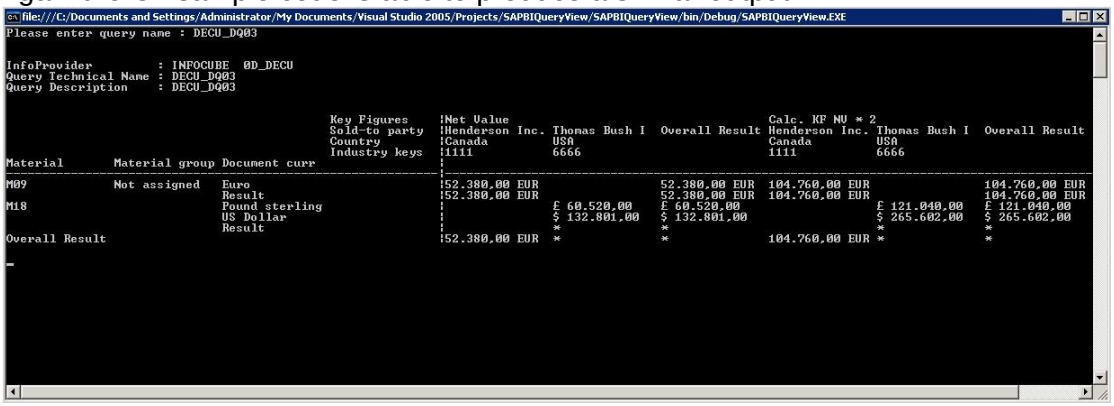

Figure 15 Result of Query with C# Program

Be very careful when retrieving the “unformatted” values of the key figures. As you can see it just added different currencies in the intermediate results. This is not a fault of the C# sample. The data was returned like this from SAP BI. So you have to ignore / disregard the sums in this case.

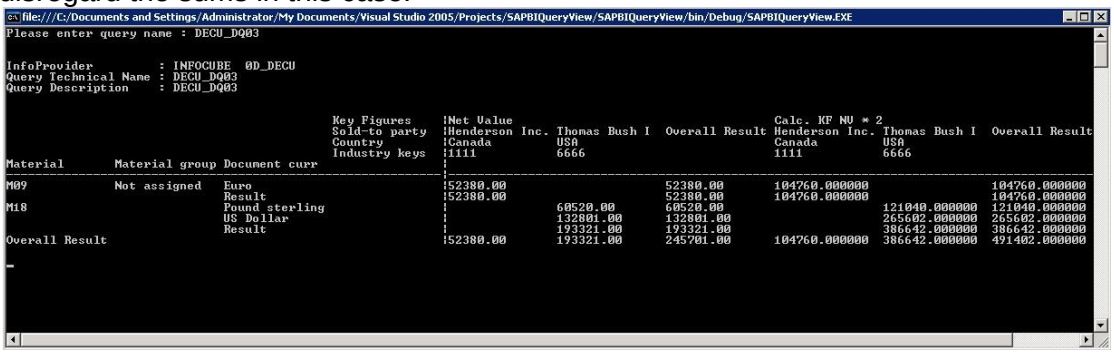

Figure 16 Unformatted Values in Result

Sample code

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using SAPBIQueryView.vsuhermann;

namespace SAPBIQueryView
{
    class Program
    {
        static void Main(string[] args)
        {
            RrwsSxAxisData[] BWQuery_AxisData;
            RrwsSxAxisInfo[] BWQuery_AxisInfo;
            RrwsSCell[] BWQuery_Cell;
            RrwsSTextSymbols[] BWQuery_TextSymbols;

            int col_width = 15;
            int i;
            int ii;
            int iii;
            int iiii;
```

```
int nr_obj_cols = 0;
int nr_obj_rows = 0;
int nr_cell_cols;
int nr_cell_rows;
int nr_attr_max;
int nr_attr_max2;
int cell_offset;
int set_offset;
int set_offset2;
RrwsSAttributes[] iobj_attr;
RrwsSAttributes[] iobj_attr2;
RrwsSxTuple[] ax0set;
RrwsSxTuple[] ax1set;
RrwsSxAxisChars[] ax0chars;
RrwsSxAxisChars[] ax1chars;
NetworkCredential login;
string infocube;
string queryname;

query_view_dataService BWQuery = new query_view_dataService();
login = new NetworkCredential();
login.UserName = "XYZ";
login.Password = "ABC";

BWQuery.Credentials = login;
BWQuery.PreAuthenticate = true;

infocube = "0D_DECU";
Console.Write("Please enter query name : ");
queryname = Console.ReadLine();
BWprint_new_line();
BWprint_new_line();

BWQuery_AxisData = BWQuery.GetQueryViewData(
 infocube,
 null,
 queryname,
 null,
 out BWQuery_AxisInfo,
 out BWQuery_Cell,
 out BWQuery_TextSymbols);

//print name of SAP BI query as well as the SAP BI InfoProvider from
//which it will retrieve the data

for (i = 0; i < BWQuery_TextSymbols.Length; i++)
{
    if (BWQuery_TextSymbols[i].SymCaption.ToString().Equals
        ("Query Technical Name"))

        Console.WriteLine("Query Technical Name : " +
          BWQuery_TextSymbols[i].SymValue.ToString());
    else if (BWQuery_TextSymbols[i].SymCaption.ToString().Equals
        ("Query Description"))
        Console.WriteLine("Query Description    : " +
          BWQuery_TextSymbols[i].SymValue.ToString());
    else if (BWQuery_TextSymbols[i].SymCaption.ToString().Equals
        ("InfoProvider"))
        Console.WriteLine("InfoProvider         : " +
          BWQuery_TextSymbols[i].SymName.ToString() + "  " +
          BWQuery_TextSymbols[i].SymValue.ToString());
}
```

```
BWprint_new_line();
BWprint_new_line();

//keep some basic numbers about objects per axis and number of rows
//in variables to make the code below more readable
nr_cell_cols = BWQuery_AxisInfo[0].Ncoords;
nr_obj_cols = BWQuery_AxisInfo[0].Nchars;

nr_cell_rows = BWQuery_AxisInfo[1].Ncoords;
nr_obj_rows = BWQuery_AxisInfo[1].Nchars;

ax0set = BWQuery_AxisData[0].Set;
ax1set = BWQuery_AxisData[1].Set;

ax0chars = BWQuery_AxisInfo[0].Chars;
ax1chars = BWQuery_AxisInfo[1].Chars;

//every characteristic might have attributes which are not counted
//by the numbers above. To make it easier later on to
//print them in the correct format it helps to first figure out
//which object on axis 0 has the highest number of attributes
nr_attr_max = 0;

for (i = 0; i < ax0set.Length; i++)
{
    if (ax0set[i].Attributes.Length > nr_attr_max)
        nr_attr_max = ax0set[i].Attributes.Length;
}

//we start to print the horizontal caption at the top first. This
//means that we need "blanks" on the left according to the
//number of objects in the caption for axis 1 which will be printed
//later for the vertical rows.
//Example :
//- let's assume we get two SAP BI key figures in the columns and a
//  customer object with one attribute ( city ) per row
//- what we want to achieve is to print the term "key figures" and
//  then the name of the key figure ( e.g. "revenue" ) on
//  top of the column
//- but later on we will print the customer name and city for every
//  row. So for the top caption we need blanks on the left
//  with the same width as customer name and city to get a nice
//  tabular output. To make this easier we use a global variable
//  which defines the column width for all fields ( no difference
//  between text or numbers )

for (i = 0; i < nr_obj_cols; i++)
{
    for (ii = 0; ii < nr_obj_rows; ii++)
    {
        //print a "blank" field for every object in the vertical
        //caption and then check how many attributes it has. For
        //every attribute print a "blank" field too
        BWprint_empty_col(col_width);

        if (ax1chars[ii].Attrinm.Length > 0)
            for (iii = 0; iii < ax1chars[ii].Attrinm.Length; iii++)
                BWprint_empty_col(col_width);
    }

    //now print the name of the horizontal caption. We reserve only
    //one column as potential attributes will be printed
    //vertically beneath. Afterwards print a separator to make the
```

```
//output more readable. The name we print could be e.g.
//the term "key figure".

BWprint_string(ax0chars[i].Caption, col_width);
BWprint_vert_sep();

//now in the same line we continue to print the horizontal
//caption for every single column which will show the actual
//key figure numbers further down. The term we printed first on
//the left could have been "key figures". In this case we would
//print now the real name of every single key figure object like
//"revenue" or "gross margin" or whatever.

for (ii = 0; ii < nr_cell_cols; ii++)
{
    set_offset = i + nr_obj_cols * ii;

    if (ii > 0)
    {
        //there is a special feature in the output of the SAP BI
        //query monitor. If a certain text will be repeated a a
        //few times per row or per column it will be printed
        //only once. To simulate this we always print the first
        //one but then for the following ones check if the
        //predecessor has the same value - if yes then just
        //print a blank field.
        set_offset2 = i + nr_obj_cols * (ii - 1);
        BWprint_string_check_dupl(ax0set[set_offset].Caption,
                  ax0set[set_offset2].Caption, col_width);
    }
    else
        BWprint_string(ax0set[set_offset].Caption, col_width);
}

BWprint_new_line();

//we are in the loop to print all objects of the top caption
//below each other. It is possible in the SAP BI query designer
//to have e.g. the key figures per some characteristic on the
//horizontal columns. But the caption would be in fact
//vertically.
//Example :
//- you have two key figures per three different customers on
//  the horizontal columns
//- this means that the caption would tell : "key figures" and
//  beneath of it "customer"
//- then you would see six columns : three customer names per
//  key figure
//- in the top line you print the term "key figure" and then
//  e.g. "revenue" and then e.g. "gross margin"
//- in the line below you print the three customer names two
//  times
//- but to get a tabular correct output you need two "blank
//  fields" after "revenue" before you can print "gross margin"
//- what makes it tricky is the fact that an object like
//  customer might have attributes which would be printed in a
//  new line like a separate object in case there are attributes
//  at all
if (nr_attr_max > 0)
{
    for (ii = 1; ii <= nr_attr_max; ii++)
    {
        //for the current object check if all attributes are
```

```
//already printed
if (ax0chars[i].Attrinm.Length >= ii)
{
    //print the appropriate number of "blank fields" on
    //the left
    for (iii = 0; iii < nr_obj_rows; iii++)
    {
        BWprint_empty_col(col_width);

        if (ax1chars[iii].Attrinm.Length > 0)
           for(iiii = 0;
                iiii < ax1chars[iii].Attrinm.Length;
                iiii++)
                BWprint_empty_col(col_width);
    }

    //print the attribute caption (e.g. "City") and
    //then the separator
    BWprint_string(
         ax0chars[i].Attrinm[ii - 1].Caption,
         col_width);
    BWprint_vert_sep();

    //then print the attribute value (e.g. "New York",
    //"London") for every column which will show key
    //figures
    for (iii = 0; iii < nr_cell_cols; iii++)
    {
        set_offset = i + nr_obj_cols * iii;
        iobj_attr = ax0set[set_offset].Attributes;

        if (iii > 0)
        {
            //like before - to simulate the output of
            //the SAP BI query monitor we always print
            //the first value and then check if it will
            //be repeated. In the latter case we will
            //just print a "blank field".
            set_offset2 = i + nr_obj_cols * (iii - 1);
            iobj_attr2 = ax0set[set_offset2].Attributes;

            if (iobj_attr.Length >= ii)
            {
                if (iobj_attr2.Length >= ii)
                    BWprint_string_check_dupl(
                      iobj_attr[ii - 1].Caption,
                      iobj_attr2[ii - 1].Caption,
                      col_width);
                else
                    BWprint_string(
                      iobj_attr[ii - 1].Caption,
                      col_width);
            }
            else
                BWprint_empty_col(col_width);
        }
        else if (iobj_attr.Length >= ii)
            BWprint_string(
               iobj_attr[ii - 1].Caption,
               col_width);
        else
            BWprint_empty_col(col_width);
    }
```

```
                BWprint_new_line();
            }
        }
    }
}

//now we are done with the top caption and start to print the
//caption on the left for the rows below
for (i = 0; i < nr_obj_rows; i++)
{
    //print the caption ( e.g. "customer" ) and then check for
    //attributes. Now as we are working on the left caption
    //we have to print attributes in the same line
    BWprint_string(ax1chars[i].Caption, col_width);

    if (ax1chars[i].Attrinm.Length > 0)
        for (ii = 0; ii < ax1chars[i].Attrinm.Length; ii++)
            BWprint_string(
              ax1chars[i].Attrinm[ii].Caption,
              col_width);
}

//to make it easier we just reserve one column for the top caption.
//The SAP BI query monitor optimizes this to save space.
//But the easy solution makes the code more readable. Therefore we
//have to print one "blank field" for the left caption
//before the separator and the cells with the key figures.

BWprint_empty_col(col_width);
BWprint_vert_sep();
BWprint_new_line();

//for the left caption there is only one line. Beneath it we print a
//separator line to make the output more readable
for (i = 0; i < nr_obj_rows; i++)
{
    BWprint_hori_sep(col_width);

    if (ax1chars[i].Attrinm.Length > 0)
        for (ii = 0; ii < ax1chars[i].Attrinm.Length; ii++)
            BWprint_hori_sep(col_width);
}

BWprint_hori_sep(col_width);
BWprint_vert_sep();

for (i = 0; i < nr_cell_cols; i++)
    BWprint_hori_sep(col_width);

BWprint_new_line();


//we printed all the captions. Now it's time to go through every row
//of the result set and print the object names / values on
//the left and then the real key figures. An object caption could be
//"Customer". The real name/value per row would then be e.g. "Smith"
for (i = 0; i < nr_cell_rows; i++)
{
    for (ii = 0; ii < nr_obj_rows; ii++)
    {
        //like before it helps to first figure out which object in a
        //row has the most attributes. We need to consider this to
        //get a correct tabular output.
```

```
nr_attr_max2 = 0;

for (iii = 0; iii < nr_cell_rows; iii++)
{
    set_offset = (iii * nr_obj_rows) + ii;

    if (ax1set[set_offset].Attributes.Length > nr_attr_max2)
        nr_attr_max2 = ax1set[set_offset].Attributes.Length;

}

set_offset = i * nr_obj_rows + ii;
iobj_attr = ax1set[set_offset].Attributes;

//now print the name/value of every object for each row
//(e.g. "Smith" for "Customer")
if (i > 0)
{
    //like before we simulate the SAP BI query monitor and
    //check for repeating values which won't be printed
    set_offset2 = ((i - 1) * nr_obj_rows) + ii;
    BWprint_string_check_dupl(ax1set[set_offset].Caption,
       ax1set[set_offset2].Caption, col_width);
}
else
    BWprint_string(ax1set[set_offset].Caption, col_width);

//now check for attributes (e.g. "City" which belongs to
//"Customer" )
if (iobj_attr.Length > 0)
    for (iii = 0; iii < iobj_attr.Length; iii++)
    {
        if (i > 0)
        {
            //again we check if a certain value will be
            //repeated. As before we print only unique
            //values and a "blank field" otherwise
            set_offset2 = ((i - 1) * nr_obj_rows) + ii;
            iobj_attr2 = ax1set[set_offset2].Attributes;

            if (iobj_attr2.Length >= iii)
                BWprint_string_check_dupl(
                    iobj_attr[iii].Caption,
                    iobj_attr2[iii].Caption,
                    col_width);
            else
                BWprint_string(
                    iobj_attr[iii].Caption,
                    col_width);
        }
        else
            BWprint_string(
                    iobj_attr[iii].Caption,
                    col_width);
    }

//another tricky thing to consider : the rows might include
//"Result" lines. They don't have attributes. So imagine an
//object in a column on the left is "customer" with
//attribute "city". Below this you get a "Result" row. But
//the "Result" row has no attribute. So in this case we
//print just a "blank field". And in case the regular object
```

```
            //of the column has <n> attributes we have to print <n>
            //"blank fields".
            //Otherwise all objects/key figures on the right wouldn't be
            //in a nice tabular format.

            if (iobj_attr.Length < nr_attr_max2)
                for (    iii = 0;
                         iii < (nr_attr_max2 - iobj_attr.Length);
                         iii++)
                    BWprint_empty_col(col_width);
        }

        //again print one "blank field" for the top caption
        BWprint_empty_col(col_width);
        BWprint_vert_sep();

        //finally print the key figures (usually numbers). As you can
        //see the structures which are returned by the Web service
        //might include both - rows and key figures as epxected but also
        //the "Result lines" with intermediate aggregations
        //( typically sum ).
        for (ii = 0; ii < nr_cell_cols; ii++)
        {
            cell_offset = i * nr_cell_cols + ii;
            BWprint_string(
                  BWQuery_Cell[cell_offset].FormattedValue,
                  col_width);
            //BWprint_string(
                  BWQuery_Cell[cell_offset].Value,
                  col_width);
        }

        BWprint_new_line();
    }

    BWprint_new_line();
    Console.ReadLine();
}

//a few print routines to make the code above more readable

static void BWprint_empty_col(int col_width)
{
    Console.Write("".PadRight(col_width));
}

static void BWprint_new_line()
{
    Console.WriteLine("");
}

static void BWprint_vert_sep()
{
    Console.Write("|");
}

static void BWprint_hori_sep(int col_width)
{
    Console.Write("".PadRight(col_width, '-'));
}

static void BWprint_string(string BWfield, int col_width)
{
```

```
            if (BWfield.Length > col_width)
                Console.Write(BWfield.Substring(0, col_width - 2) + "  ");
            else
                Console.Write(BWfield.PadRight(col_width));
        }

        static void BWprint_string_check_dupl(string BWfield1,
                  string BWfield2, int col_width)
        {
            if (BWfield1.Equals(BWfield2))
                BWprint_empty_col(col_width);
            else
                BWprint_string(BWfield1, col_width);
        }
    }
}
```

References

- Microsoft SAP Customer Information Center
  http://www.microsoft.com/sap
- Microsoft SQL Server BI
  http://www.microsoft.com/sql/solutions/bi/default.mspx
- SAP Help Portal
  http://help.sap.com