How to Implement a SAP HANA Database Procedure and consume it from an ABAP Program

Step-by-Step Tutorial

Table of Contents

This tutorial demonstrates how easy it is to use the new AS ABAP built-in possibilities provided with SAP NetWeaver 7.4 to leverage the power of SAP HANA in your ABAP applications.

You will get to know step-by-step, how to create and implement a SAP HANA Procedure, expose it in the application server ABAP using the new dictionary features and then easily consume it in an ABAP report.

For more information and guides about development based on SAP NetWeaver AS ABAP and SAP HANA visit our SCN Page http://scn.sap.com/docs/DOC-35518.

Prerequisites

SAP NetWeaver AS ABAP 7.4 SP2, SAP HANA Appliance Software SPS 04 (or higher), SAP HANA DB SQLScript V2.0 (or higher), SAP HANA Studio, ABAP Development Tools for SAP NetWeaver

Benefits of using SAP HANA Procedures

SAP HANA procedures enable you to embed data-intensive application logic into the database, where it can be optimized for performance.

Some of the advantages of using SAP HANA procedures instead of standard SQL are:

- Large data are not transferred to the application server whenever possible resulting in less network and processor load (e.g. in case of data-intensive aggregations).

- Less roundtrips by returning multiple result sets while only one result set can be retuned with standard SQL queries.

- Local variables are possible, eliminating the need to explicitly create temporary tables for intermediate results.

- Easy and advanced parameterization of queries using input parameters (e.g. versus standard SQL SELECT)

- Advanced features like parallel execution or complex calculation – such as for financials - are supported on the database

More information about Procedures can be found in the SAP HANA Developer Guide.

Objectives

After completing this tutorial, you will be able to:

- Create and implement a SAP HANA Database Procedure
- Expose a SAP HANA Procedure in the ABAP dictionary as Database Procedure Proxy
- Consume a Database Procedure Proxy in an ABAP program

Use Case Description

The Account Receivables accountant of your company want to be able to display the so-called top and flop customers in regards to their payment ability based on the open invoices.

The company accountant should be able to select how many customers have to be displayed per category and due to the regular update of the business data, the categorization have to be executed on-the-fly.

More information about the *Open Items Analytics* reference scenario underlying this use case is available here.

Procedure Overview

Step-by-Step Procedure

Step 1: Create and implement a SAP HANA Procedure with Input and Output Parameters

Note: You need a database user in order to create and manipulate objects in the SAP HANA DB.
It is recommended to create your test objects in package *system-local.private* located under the *Content* folder of the SAP HANA Modeler.

a. Start the SAP HANA Studio and go the *Modeler* perspective
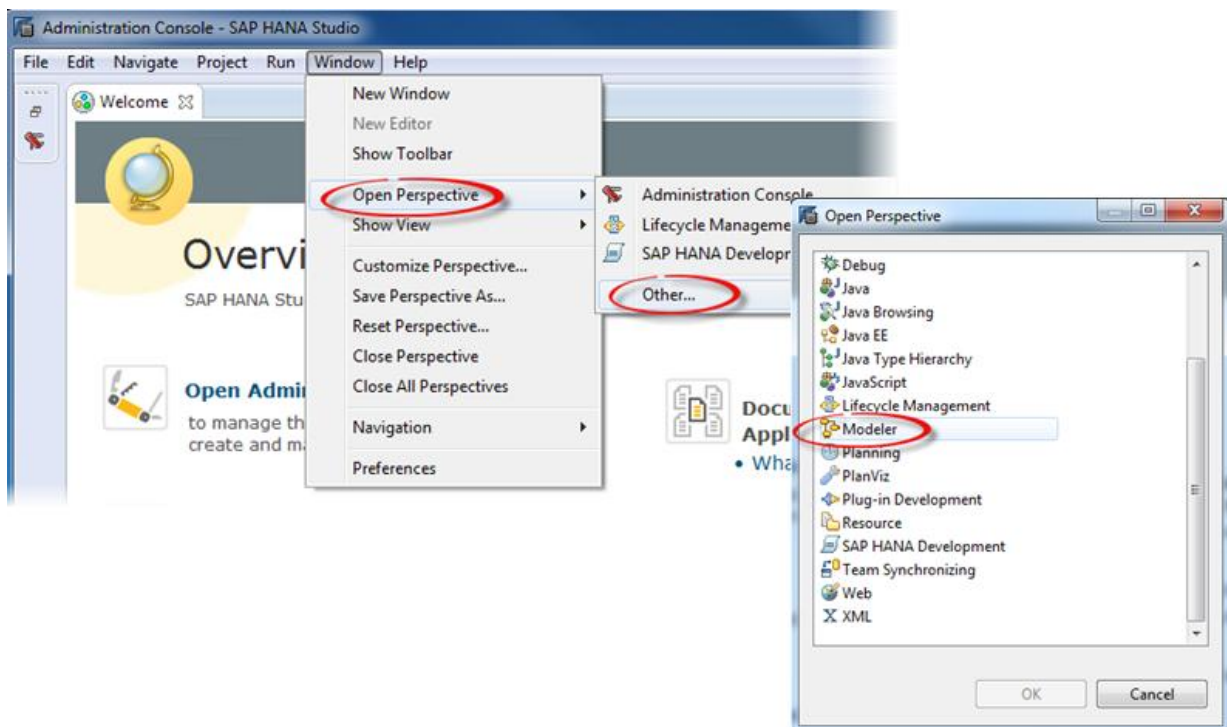
b. *Optional*: First create a new package (e.g. *demo*) for your demo objects under package *system-local.private which is* located under the *Content* folder

    Now create a new database procedure by right-clicking on package *system-local.private.demo* and selecting menu path *New -> Procedure …*

    Enter a Name (e.g. Z*DP_OIA_TOPANDFLOP*) and a description ("*OIA demo: Top and flop Customers*").

    Select the required Schema from the *Default Schema* dropdown list, for unqualified access in SQL.

Note: An application server ABAP is installed in the database in a schema, the so-called *SAP<SID>*, so you can work in the database with multiple schemas.
For example, the schema SAPABC belongs to the system whose SID is 'ABC'.

In order to not specify the *SAP<SID>* directly in the procedure's script, a default schema has to be specified.
For example:
- Procedure with *SAPABC* specified as *Default Schema*: SELECT company_name FROM snwd_bpa
- Procedure with no *Default Schema* specified: SELECT company_name FROM *SAPABC*.snwd_bpa

Select '*Invoker's Rights*' from the *Run With* dropdown list.

Note: The option selected from the *Run With* dropdown list identifies whose rights are to be considered while executing the scenario.

- Invoker's right: If you want the standard database user SAP<SID> to use the rights of the current user while executing the view or procedure.
- Definer's right: If you want SAP<SID> to use the rights of the definer while executing the view or procedure for any user.

Press on *Finish.*
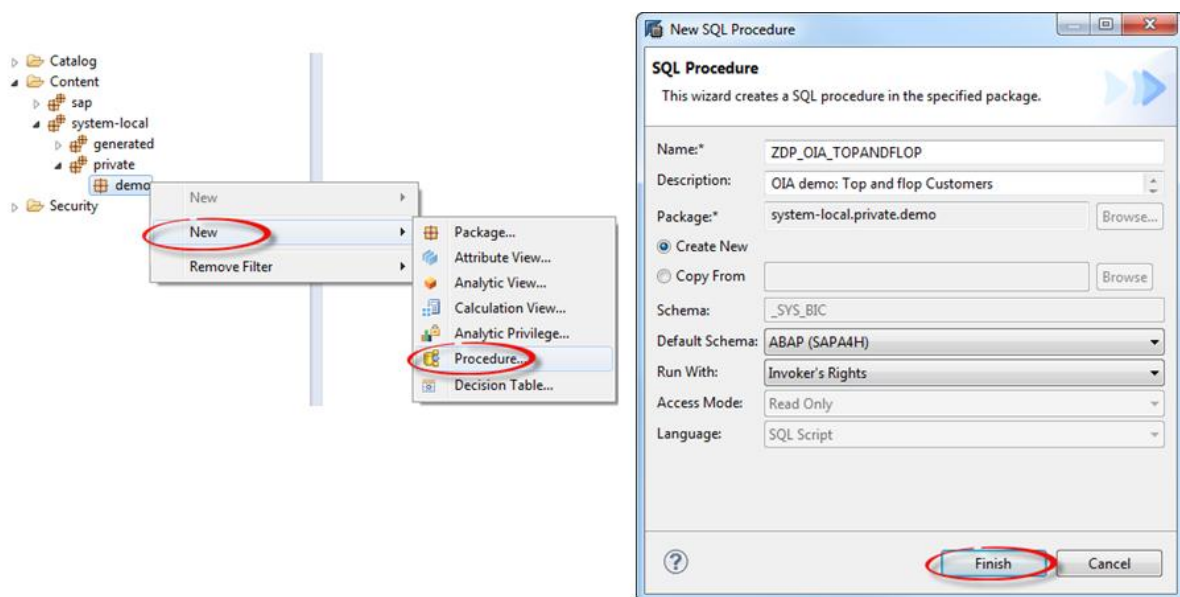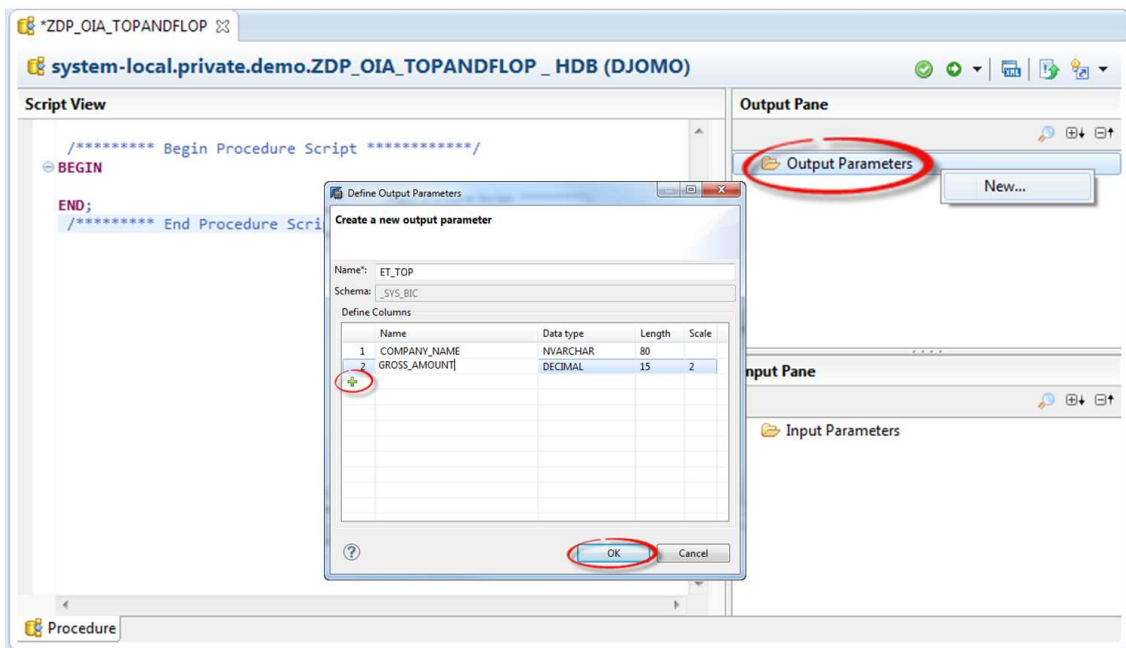

c. Create the output parameter ET_TOP
Go to the *Output Pane*, right-click on the root folder *Output Parameters* and select *New…* from the context menu. Now enter the name "*ET_TOP*" and add two columns (by clicking on the
✛ icon) with following values and confirm:
- Column 1
  - Name   : COMPANY_NAME
  - Data Type : NVARCHAR
  - Length   : 80
- Column 2
  - Name   : GROSS_AMOUNT
  - Data Type : DECIMAL
  - Length   : 15
  - Scale   : 2

d. Create the output parameter ET_FLOP
   Just repeat the steps performed previously for ET_TOP.

e. Create the input parameter IV_NUMBER
   Go to the *Input Pane*, right-click on the root folder *Input Parameters* and select *New Scalar Parameter…*
   from the context menu. Enter following values and confirm
   - Name   : IV_NUMBER
   - Data Type : INTEGER
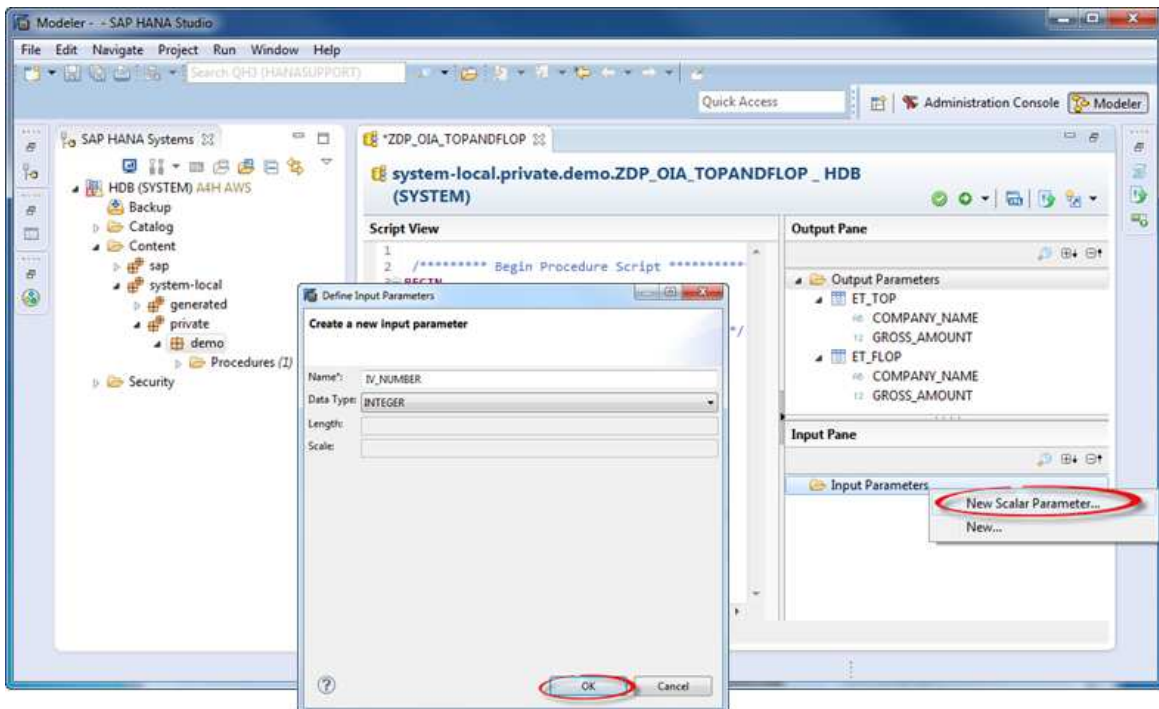
f. Implement the procedure.
   Copy & paste the SQLScript below into the *Procedure Script View Editor*

<table>
<tr><td align="center">**Procedure DP_OIA_TOPANDFLOP**</td></tr>
</table>

```
/********* Begin Procedure Script ************/
BEGIN
--retrieve the best customers
        et_top = select top :iv_number b.company_name as company_name, sum(i.gross_amount)
as gross_amount from snwd_so_i as i
                    inner join snwd_so as h on h.node_key = i.parent_key
                    inner join snwd_bpa as b on b.node_key = h.buyer_guid
                    group by company_name
                    order by gross_amount desc;

--retrieve the worst customers
      et_flop = select top :iv_number b.company_name as company_name, sum(i.gross_amount)
as gross_amount from snwd_so_i as i
                    inner join snwd_so as h on h.node_key = i.parent_key
                    inner join snwd_bpa as b on b.node_key = h.buyer_guid
                    group by company_name
                    order by gross_amount ASC;
END;
 /********* End Procedure Script ************/
```

Note: Showing how complex the logic of a procedure can be is not the main purpose of the present exercise. The above procedure implements a relatively simple data-intensive function, but of course very complex logic can be implemented instead by making use of the advanced native SAP HANA features and functions.

g. Save and activate the procedure.

Step 2: Create a Database Procedure Proxy in the ABAP Dictionary to expose the HANA Procedure

We will now expose the created HANA procedure *DP_OIA_TOPANDFLOP* as Database Procedure Proxy in the application server using the new ABAP dictionary features.

Note: The new ABAP dictionary features are only available in the ABAP Development Tools for SAP NetWeaver (aka ABAP in Eclipse) and not in the ABAP Workbench (transaction SE80).

a. First switch to the ABAP perspective
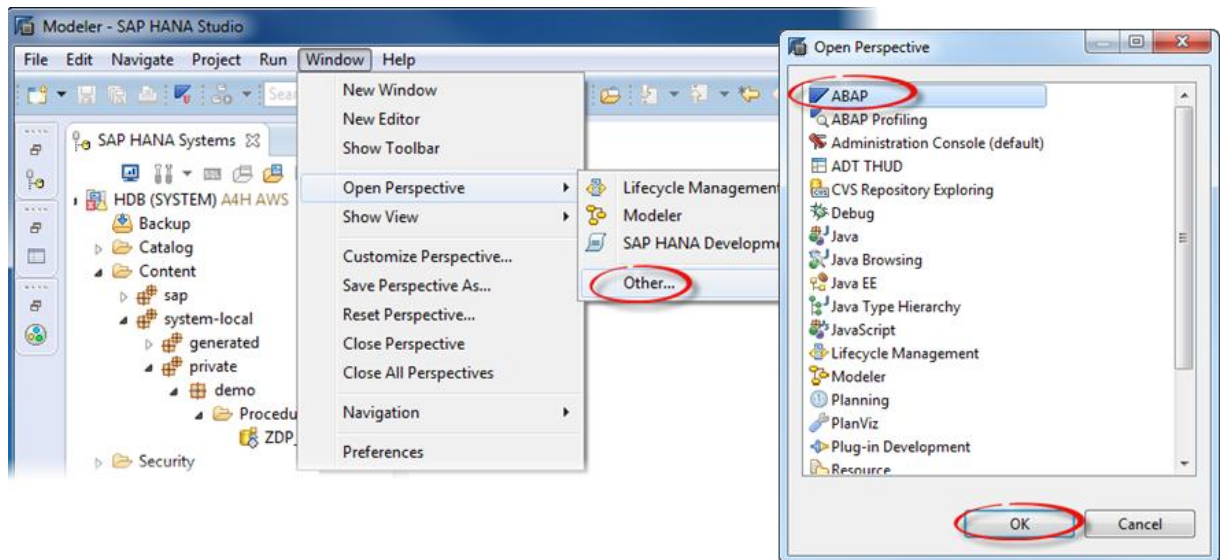
b. Select the package of your choice (e.g. your *$TMP* package under *Favorite Packages*), right-click on it and select *New -> Other ABAP Repository Object* from the context menu.

   Filter for "*Database Procedure Proxy*", select it and press on *Next*.
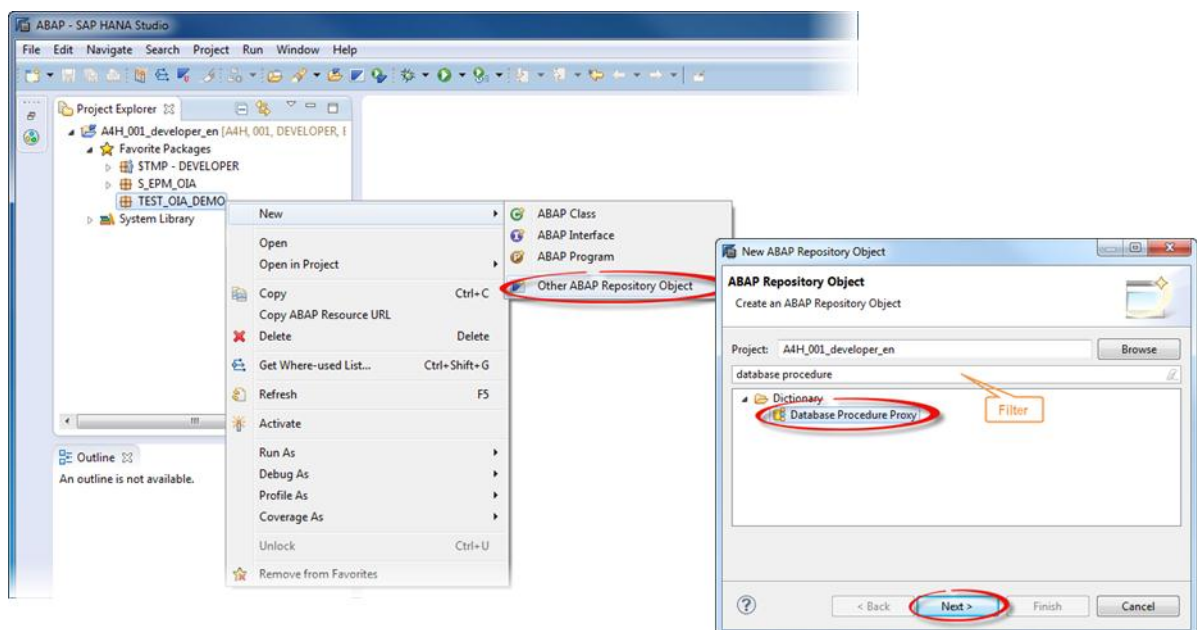
c. Enter a name (e.g. *ZDPP_OIA_TOPANDFLOP*) and a description (e.g. "*Database Procedure Proxy for ZDP_OIA_TOPANDFLOP*") for the new Database Procedure Proxy.

Enter the name of the SAP HANA Procedure previously created in Step 1 (e.g. *system-local.private.demo.ZDP_OIA_TOPANDFLOP*).

Note: You can use shortcut Ctrl+Space for the Search-Help (💡 *Content Assist* Available)

A name is proposed for the *Parameter Type Interface* which contains the data type definition of the input and output parameters defined in the HANA Procedures. The interface will automatically be created together with the database procedure proxy.

Now press on *Next* and then on *Finish* on the next dialog step.
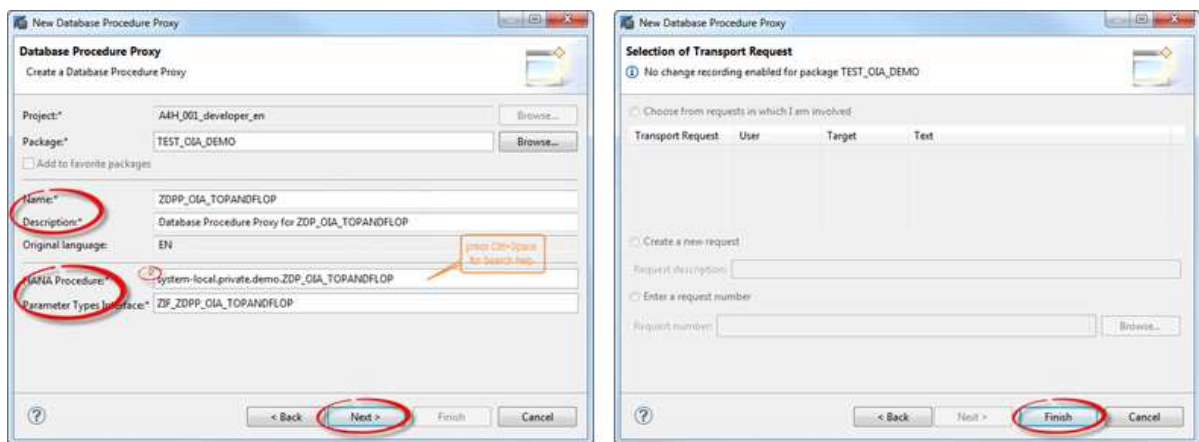
d. You can now have a look at the created objects: Database Procedure Proxy and its interface.

Activate both of them (🗒 or *Ctrl+Shift+F3*).

Note: Press the *"Synchronize"* button to reload the metadata of the Database Procedure if the procedure was changed in the SAP HANA repository and do not forget to activate the updated dictionary object afterwards.
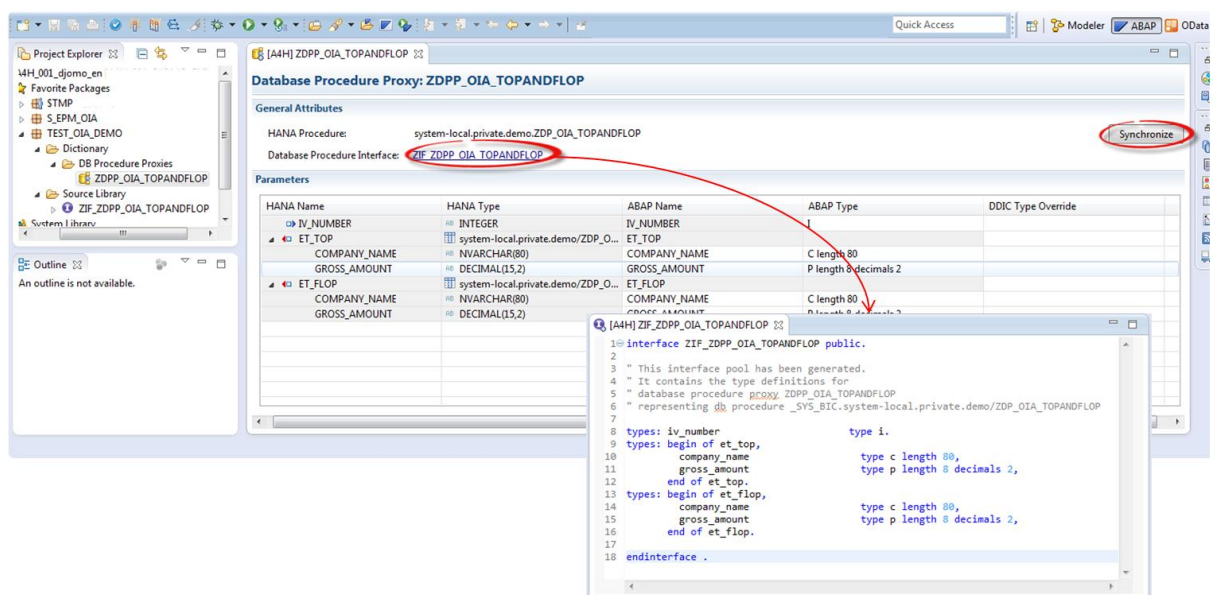
Step 3: Create and implement an ABAP Report consuming Procedure

We will now call the database procedure proxy from an ABAP program. We will implement a simple report which will just output the result with a WRITE statement.

The following steps are performed in the ABAP perspective.

a. Create a new ABAP Program in the package of your choice by just right-clicking on it and selecting "*New -> ABAP Program*" from the context menu.
Enter a name (e.g. *ZR_OIA_TOPANDFLOP*) and a description (e.g. "*Display Top and Flop Customers*").

Press on *Next* and then on *Finish* on the dialog step.
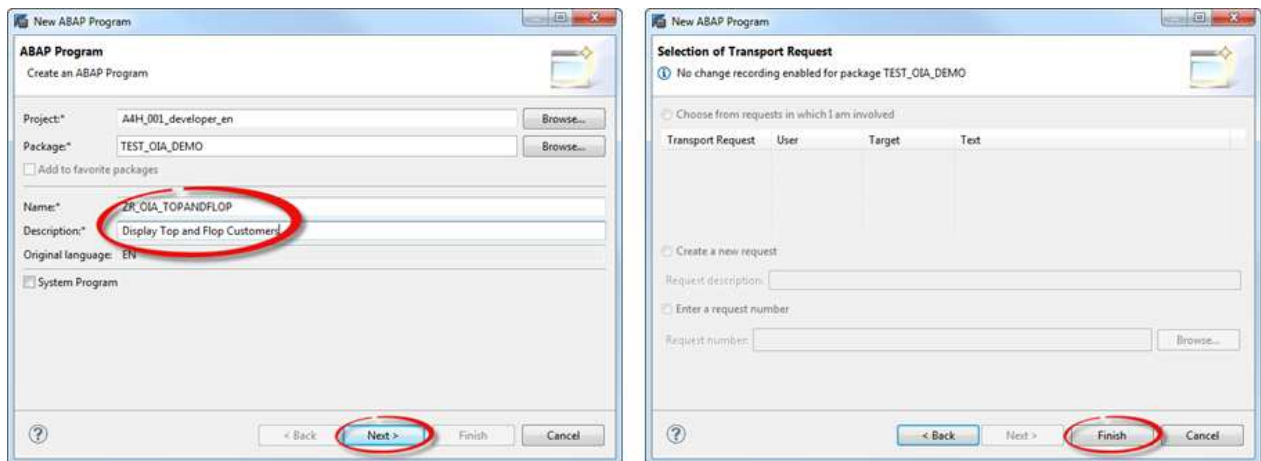
b. Implement the report.
Copy & paste the source code below into the ABAP Editor.

<div align="center">

Report *ZR_OIA_TOPANDFLOP*

</div>

```abap
REPORT ZR_OIA_TOPANDFLOP.

* input parameter: how many tops/flops should be displayed
PARAMETER pnumber TYPE i DEFAULT 10.

* define parameters using the created DPP interface ZIF_ZDPP_OIA_TOPANDFLOP
DATA: lv_number TYPE zif_zdpp_oia_topandflop=>iv_number,
      lt_top    TYPE STANDARD TABLE OF zif_zdpp_oia_topandflop=>et_top,
      lt_flop   TYPE STANDARD TABLE OF zif_zdpp_oia_topandflop=>et_flop.

* set the value of the procedure input parameter
lv_number = pnumber.
* call the created database procedure proxy
CALL DATABASE PROCEDURE zdpp_oia_topandflop
    EXPORTING   iv_number = lv_number
    IMPORTING   et_top    = lt_top
                et_flop   = lt_flop .

* display the returned itab with TOP customers
WRITE: / 'Tops:' COLOR COL_POSITIVE.
LOOP AT lt_top ASSIGNING FIELD-SYMBOL(<f>).
  WRITE:/ <f>-company_name , <f>-gross_amount  .
ENDLOOP.
* display the returned itab with FLOP customers
ULINE. WRITE: 'Flops:' COLOR COL_NEGATIVE.
LOOP AT lt_flop ASSIGNING FIELD-SYMBOL(<g>).
  WRITE:/ <g>-company_name , <g>-gross_amount .
ENDLOOP.
```
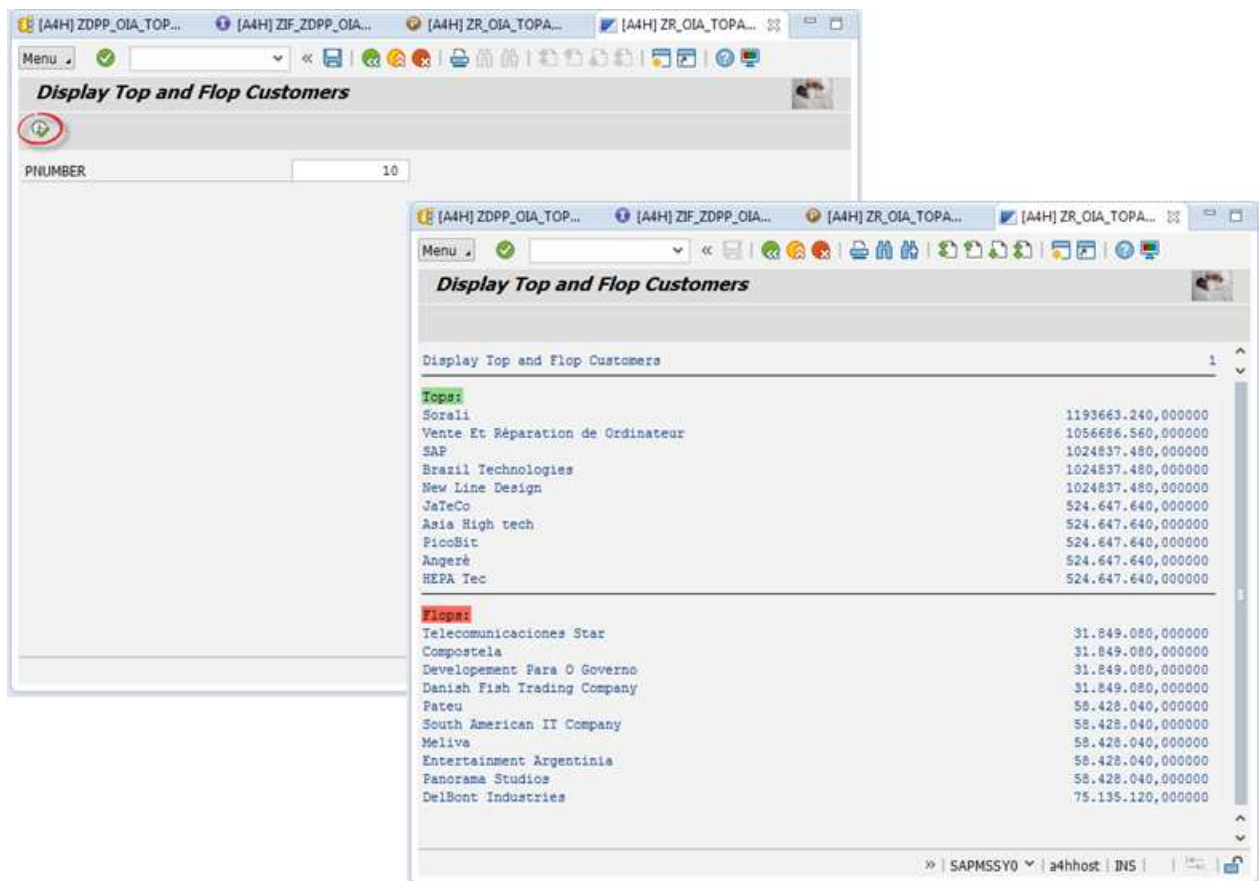
> **Note**: The new ABAP statement "*CALL DATABASE PROCEDURE <NAME>"* is used to read data from the SAP HANA database. The name of the *Database Procedure Proxy* defined in the ABAP dictionary is expected. *Database Procedure Proxy* can be called in the same manner from other development objects such as classes, function modules and Web Dynpro applications.

c. Save ⊞ and Activate ⊞ your report

d. You can now run the report (press on F8) – and by the way see the result of your effort ☺

Summary

Congratulations, you have just experienced how easy it is to consume SAP HANA procedures from an ABAP programs. You can now create a SAP HANA Database Procedure, expose it as Database Procedure Proxy in the ABAP dictionary and easily consume it in an ABAP program.

The following illustration provides the high level architecture underlying this tutorial.
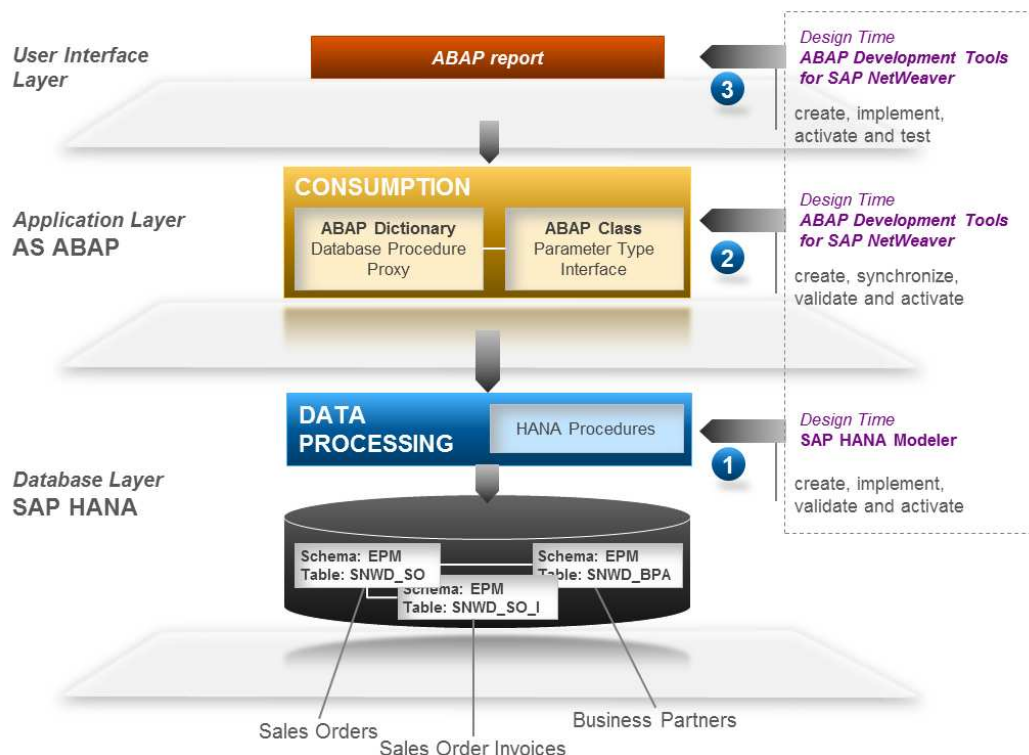
Related Content

ABAP for SAP HANA Reference Scenario, SCN document, http://scn.sap.com/docs/DOC-35518

SAP HANA Developer Guide, SAP Help Guide, http://help.sap.com/hana/hana_dev_en.pdf

SAP HANA SQLScript Reference, SAP Help Guide, http://help.sap.com/hana/hana_dev_sqlscript_en.pdf

www.sap.com