How to use the flexibility of AMDP along with effectiveness of CDS view

AccreteConsultingSolutions

HANA And Code Pushdown Technique Using Top-down Approach

As SAP is moving towards code push down technique using top-down approach, both CDS views and AMDP are very powerful tools to achieve this framework. As both CDS view and AMDP have their own pros and cons, choosing one of these is governedby the business requirement as well involved landscape.

Few factors which may influence the decision while choosing between AMDP over CDS view:

- In AMDP, we can call one function inside the other, it is helpful in returning multiple result set on complex logics. Whereas, CDS is dedicated for single set of logic and return only one result set..
- CDS views can be created to read and process data at DB layer. Whereas AMDP can be created to process and modify data at DB layer.
- AMDP is used to work with stored procedures, which further go to HANA DB layer and execute that. This functionality can't be achieved by Open SQL and CDS.

Some factors which may result in choosing CDS view over AMDP

- Ability to reuse the SAP HANA database artifact in CDS view.
- It has more advanced features like Associations and Annotations in CDS view.
- Client handling feature in CDS view.
- CDS views can be directly consumed by other SAP products such as SAC, BODS, BI etc. to fetch data from SAP business suite database layer.
- CDS can be exposed as OData Service which can then be used even by non-SAP systems to fetch data from SAP business suite database layer.

As mentioned above, it's a very common scenario these days that limit fetching data from SAP database layer using only CDS view but the business requirement is so complex that the coding flexibility of AMDP is sorely missed.

However, SAP has provided tools to handle such scenarios as well. To address this restriction and to provide power of AMDP to CDS views, SAP has provided ABAP CDS Table function.

What Is ABAD CDS Table Function?

ABAP CDS table functions define table functions that are implemented natively on the database and can be called in CDS.

A CDS table function is defined using the ABAP CDS statement DEFINE TABLE FUNCTION and can be used as the data source in Open SQL read statements.

This implementation is done within an AMDP method of an AMDP class and is managed as an AMDP function by the AMDP framework in the database system.

Real world scenario:

Let's take a look at real world example to better understand how to create/define a Table function and how to use it handle complex logic in CDS view with quite an ease.

The requirement is to fetch Order details from "SAP Retail business suite (also known as CAR)" tables where field "Order Timestamp" is between date passed as parameter upto next 45 days.

A typical CDS would view looks like this:

```
@AbapCatalog.sqlViewName: 'ZIORDPLAN2'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Order plan report'
define view ZCDS_I_ORD_PLAN_REP2
with parameters
p_date : datum
as select from /dmf/ordpln_itm as a
left outer join /dmf/prod_ext_xr as b on a.prod_id = b.prod_id
left outer join /dmf/loc_ext_xr as c on a.locto = c.loc_id
left outer join /dmf/opln_trcopi as d on a.orderplanitemuuid = d.orderplanitemuuid
left outer join /dmf/opln_trcitm as e on a.orderplanitemuuid = e.orderplanitemuuid
and e.is_selected = 'X'
left outer join /xrp/c_rop_ogmda as f on a.orderplanitemuuid = f.orderplanitemuuid

{
a.orderplanitemstatus,
a.orderdatetime,
a.deliverydatetime,
a.order_quantity_proposed,
b.ext_prod_id,
c.ext_loc_id,
d.stock_avail_tstmp_qty,
d.sales_fcst_demand_period,
d.standard_deviation_dmnd_period,
e.lost_sales_qty,
e.inventory_eop_qty,
f.opengoodsmovementquantity
}
```

Where

```
a.orderdatetime between dats_tims_to_tstmp( $parameters.p_date, '000000',
abap_system_timezone( $session.client, 'NULL' ), $session.client, 'NULL' ) and
dats_tims_to_tstmp( DATS_ADD_DAYS( $parameters.p_date, 45, 'INITIAL' ), '000000',
abap_system_timezone( $session.client, 'NULL' ), $session.client, 'NULL' )
```

```
and(
      a.orderplanitemstatus =     '00'
or a.orderplanitemstatus =      '20'
or a.orderplanitemstatus =      '25'
or a.orderplanitemstatus =      '26'
)
```

Focus on the Yellow highlighted section. Here the field “orderdatetime” has datatype “timestamp” and we have to fetch all records between “p_date” (parameter of CDS view having data type YYYYMMDD) and (p_date + 45 days). Just look how complex the code has become just to fetch this simple requirement.

Alternatively, if we use AMDP capabilities of CDS Table Function to meet the same requirement, the coding become way simpler without sacrificing the performance: -

Table Function definition

A typical Table Function definition to replicate above CDS view business logic:

```
@EndUserText.label: 'Order plan report using table function.'
@ClientHandling.type: #CLIENT_DEPENDENT
@ClientHandling.algorithm: #SESSION_VARIABLE
define table function ZUTL_ORD_PLAN_TABLE_FUNCTION
 with parameters
   p_date  : abap.dates,
   @Environment.systemField: #CLIENT
   p_client : abap.clnt
returns
{
Key client : abap.clnt;
key ext_prod_id : /DMF/ext_prod_id;
key ext_loc_id : /dmf/ext_location_id;
orderplanitemstatus
orderdatetime
deliverydatetime
order_quantity_proposed
stock_avail_tstmp_qty
sales_fcst_demand_period
standard_deviation_dmnd_period : /dmf/stdv_demand_period;      : /dmf/kpi_lost_sales_qty;
lost_sales_qty                                                 : /dmf/kpi_inventory_eop_qty;
inventory_eop_qty                                              : /dmf/kpi_inventory_eop_qty;
OpenGoodsMovementQuantity                                     : /dmf/goods_movement_open_qty;
}
```

```
 implementedby method
zcl_order_plan_report=>get_plan_report;
```

It looks similar to a CDS view with the difference being logic to fetch data is implemented by an AMDP class (highlighted using yellow above).

Let us now look at the implementation which is in the method of an AMDP class:

Table Function implementation (Class implementing the table function)

```
CLASS zcl_order_plan_report IMPLEMENTATION.
 METHOD get_plan_report BY DATABASE FUNCTION
                        FOR HDB LANGUAGE SQLSCRIPT
                        USING /dmf/ordpln_itm
                              /dmf/prod_ext_xr
itab1 = SELECT
a.mandt,
b.ext_prod_id,
c.ext_loc_id,
a.orderplanitemstatus,
a.orderdatetime,
a.deliverydatetime,
a.order_quantity_proposed,
/dmf/loc_ext_xr
/dmf/opln_trcopi
/dmf/opln_trcitm
/xrp/c_ropi_demand_open_gm_agg.
```

```
  declare lv_date_from date;
    declare lv_date_to date;
    declare lv_date_to_n nvarchar( 8 );
    declare lv_datetime_from dec(15);
    declare lv_datetime_to dec(15);
    lv_date_from = p_date;
    lv_date_to = add_days( lv_date_from, 45 );
    lv_date_to_n = to_dats( lv_date_to );
    lv_datetime_from = concat( p_date, '000000' );
    lv_datetime_to  = concat( lv_date_to_n, '000000' );
```

```
itab1 = SELECT
a.mandt,
b.ext_prod_id,
c.ext_loc_id,
a.orderplanitemstatus,
a.orderdatetime,
```

```
a.deliverydatetime,
a.order_quantity_proposed,
d.stock_avail_tstmp_qty,
d.sales_fcst_demand_period,
d.standard_deviation_dmnd_period,
e.lost_sales_qty,
e.inventory_eop_qty,
f.OpenGoodsMovementQuantity
from "/DMF/ORDPLN_ITM" as a
left outer join "/DMF/PROD_EXT_XR" as b on a.prod_id = b.prod_id
left outer join "/DMF/LOC_EXT_XR" as c on a.locto = c.loc_id
left outer join "/DMF/OPLN_TRCOPI" as d on a.orderplanitemuuid = d.orderplanitemuuid
left outer join "/DMF/OPLN_TRCITM" as e on a.orderplanitemuuid = e.orderplanitemuuid
                                and e.is_selected      = 'X'
left outer join "/XRP/C_ROPI_DEMAND_OPEN_GM_AGG" as f on a.orderplanitemuuid =
f.orderplanitemuuid
where

  a.orderdatetime between lv_datetime_from and lv_datetime_to

 and(
 a.orderplanitemstatus =      '00'
 or a.orderplanitemstatus =  '20'
 or a.orderplanitemstatus =  '25'
 or a.orderplanitemstatus =  '26'
 );
 RETURN SELECT
 mandt AS client,
 ext_prod_id,
 ext_loc_id,
 orderplanitemstatus,
 orderdatetime,
 deliverydatetime,
 order_quantity_proposed,
 stock_avail_tstmp_qty,
 sales_fcst_demand_period,
 standard_deviation_dmnd_period,
 lost_sales_qty,
 inventory_eop_qty,
 OpenGoodsMovementQuantity
 FROM :itab1;

 ENDMETHOD.

 ENDCLASS.
```

Focus on the yellow highlighted section. We can write code within AMDP class using native HANA SQL language similar to our ABAP language thereby making data manipulations so very simple. Also, unlike CDS, you can debug this to rectify the bug!!

This table function can be consumed in a CDS view just like you would consume any transparent table (see yellow highlighted section):

```
@AbapCatalog.sqlViewName: 'ZCORDPLAN3'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Order plan report using table function'
define view ZCDS_C_ORD_PLAN_REP3
as select from ZUTL_ORD_PLAN_TABLE_FUNCTION(p_date: $session.system_date, p_client:
$session.client)
{
  ext_prod_id,
  ext_loc_id,
 orderplanitemstatus,
  orderdatetime,
 deliverydatetime,
 orderquantityunit,
  @Semantics.quantity.unitOfMeasure: 'orderquantityunit'
  order_quantity_proposed,
  @Semantics.quantity.unitOfMeasure: 'orderquantityunit'
 stock_avail_tstmp_qty,
  @Semantics.quantity.unitOfMeasure: 'orderquantityunit'
  lost_sales_qty,
  @Semantics.quantity.unitOfMeasure: 'orderquantityunit'
 OpenGoodsMovementQuantity,
  @Semantics.quantity.unitOfMeasure: 'orderquantityunit'
 inventory_eop_qty
 }
```

This way, we can implement complex business logic using the flexibility of AMDP without compromising on the benefits of CDS view utility. For detail discussion and questions, please reach out to our team at corp@acnsol.com.

USA
7116 252nd Avenue NE
Redmond, WA 98053

Noida
The Iconic Corenthum
1st & 2nd floor, Sector
62, Noida-201301

South Africa
609 Lanseria Corporate
Estate, Falcon Lane,
Lanseria, Gauteng

AccreteConsultingSolutions