Tushar Sharma

September 9, 2017 | 8 minute read

ABAP Core Data Services – Part 1(ABAP CDS Entities)

Table of Content:

7.	Credits

This blog is continuation of my previous blog:

ABAP Core Data Services – Introduction (ABAP CDS view) in which a detailed introduction is given about the Core Data services.

Before, start reading this I would request you to go through above mentioned blog for better and smooth understanding.

In this blog we'll start about the ABAP Core Data services.

Before we Start...

Prerequisites

SAP NetWeaver Releases

- SAP NetWeaver 7.5 SP01, or higher
- No specific database is required, but SAP HANA DB is recommended

Development Environment

- It's recommended to use the latest version of ABAP Development Tools for SAP NetWeaver (in short: ADT).You can download the latest available ADT plugin from the update site https://tools.hana.ondemand.com/#abap

SAP Gateway

- SAP Gateway is properly configured in your ABAP system for activating and testing the resulting OData service. More on this on the SAP help portal: SAP Gateway Foundation Developer Guide.

Authorizations

To reproduce the steps described in above guide, the user on SAP NetWeaver Application Server with the following roles assigned is required:

- SAP_BC_DWB_ABAPDEVELOPER
- SAP_BC_DWB_WBDISPLAY
- /IWFND/RT_DEVELOPER (for SAP Gateway service development).

Hoping, all the above mentioned prerequisites are setup.

Let's Start !!

Overview

After the introduction of CDS in SAP HANA, SAP realised CDS can be introduced with ABAP application server which will allow ABAP to benefit from the enhanced capabilities that are offered by the data definition language of CDS compared to the form based ABAP Dictionary tool. So, SAP introduced ABAP CDS views in ABAP 7.40 SP05 release. Since the ABAP Dictionary already had the capabilty of defining tables, views and data types, the natural way of introducing CDS on the ABAP application server was to add it to the ABAP Dictionary. An ABAP Development Tools (ADT) based source code editor allows us to create Data Definition Language(DDL) sources in Eclipse or SAP HANA studio.

An ABAP-based CDS plays a significant role in the foundation of SAP Business Suite 4 SAP HANA (SAP S/4HANA). A large set of CDS artifacts — several thousand of CDS views — consisting of several hundred thousand lines of ABAP code, has been intoduced by SAP to represent the underlying core data model of the SAP S/4HANA solution. The standard ABAP CDS views, ERP tables and views can be reused in custom CDS views in ADT source editor.

The main motivation behind using ABAP CDS views from ABAP 7.50 SP00 where most of all the functionalities has been introduced for SAP S/4HANA is to provide a semantic layer on top of the traditional physical SAP ERP tables, which often act as physical data containers on the database and have a very complex inner structure which, in most cases, cannot be evaluated without traditional ABAP processing. The semantically rich data modeling provided by CDS enables this layer in SAP S/4HANA facilitating simplified, efficient access to the underlying data.

ABAP CDS Entities

ABAP CDS provides a framework for defining and consuming semantic data models on the central database of the application server AS ABAP. The specified data models are based on the data definition language (DDL) and the data control language (DCL) which are managed by ABAP Dictionary. So, a CDS entity or the enhancement of a CDS view is defined as source code in the CDS data definition.

To define an ABAP CDS entity, you first need to create a **DDL source** as the relevant development object with which you can use the standard functions of the ABAP

Workbench – such as syntax check, activation, or connecting to the Transport Organizer. A CDS entity is defined in the text-based **DDL editor** of ABAP Development Tools in Eclipse/SAP HANA studio.
The following types of ABAP CDS entities are supported:

- ABAP CDS Views
- ABAP CDS Table Functions

Defining ABAP CDS Views

A CDS view is defined for existing database tables and views, or for other CDS views in ABAP Dictionary, using the ABAP CDS statement `DEFINE VIEW`. A CDS view serves to define the structure of an SQL view and represents a projection onto one or several Dictionary tables or Dictionary views.

Note:- SQL views and CDS entities are part of one and the same namespace. Therefore, you must assign different names for an SQL view and the entity.

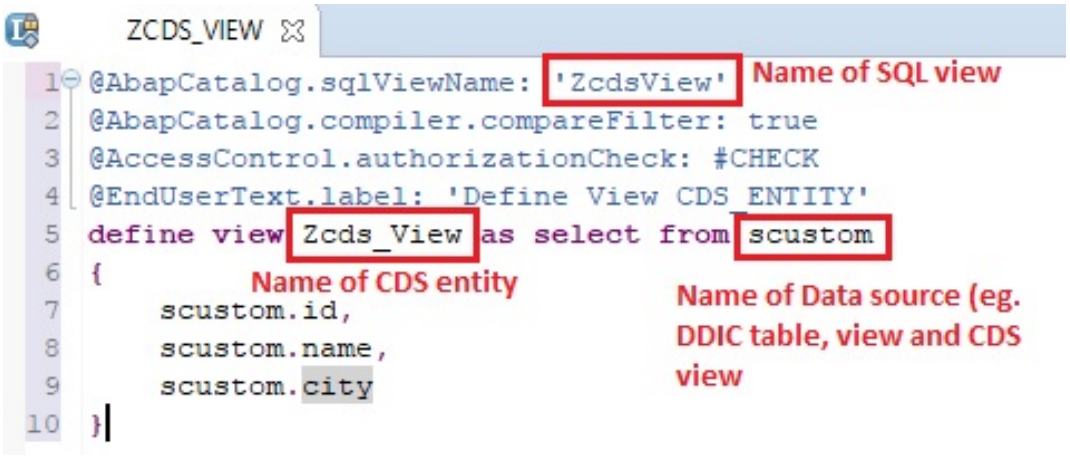

Figure 1: Defining the CDS view in the DDL editor.

In figure:1 the CDS entity `ZCDS_VIEW` defines a projection onto the database tables `scustom`. The generated SQL view (`ZcdsView`) comprises the ID, the name, and the city of all entries.

Post Activation

After activating a CDS view, the following objects are created in ABAP Dictionary:

a). The actual CDS entity (Zcds_View)

b). An SQL view (ZcdsView).

Defining ABAP CDS Table Functions

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.

Each CDS table function includes the following components:

- The actual **CDS entity** of the table function that is generated in the ABAP Dictionary.
- The **CDS table function implementation** (ABAP class library).

Note:- In contrast to the CDS views, the CDS table functions can be implemented using Native SQL. 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.

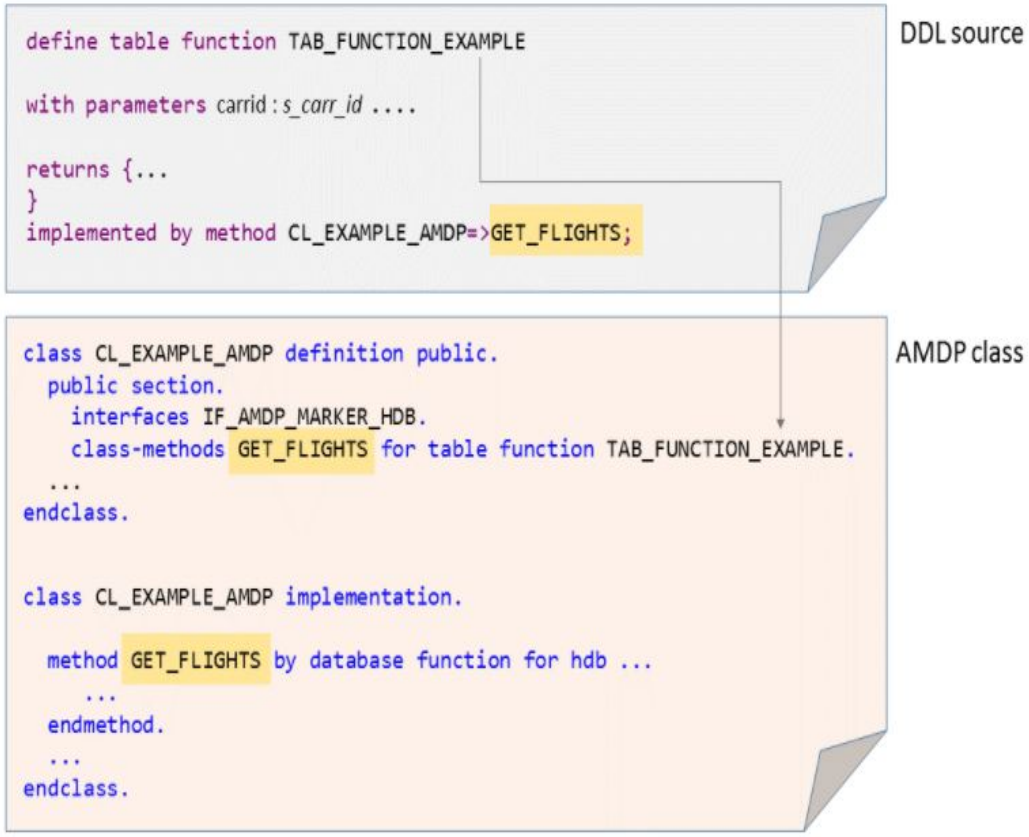

Figure 2: Defining and Implementing CDS table functions.

Note:- The name of the implementing AMDP method can only be specified in a single CDS table function (1: 1 relation).

Example 1:- **Table Function Definition.**

In the following listing, a client-specific ABAP CDS table function `TAB_FUNCTION_EXAMPLE` is defined using the DDL syntax. This table function declares two input parameters clnt (with the predefined value: #CLIENT)

and carrid, and a list of elements that provide the return values of the AMDP method that implements the table function. The table function is associated with the AMDP class `CL_EXAMPLE_AMDP`, where the method `GET_FLIGHTS` is used to implement the table function.

```
@ClientDependent: true
 @AccessControl.authorizationCheck: #NOT_REQUIRED
 define table function TAB_FUNCTION_EXAMPLE

    with parameters @Environment.systemField: #CLIENT
            clnt:abap.clnt, carrid : s_carr_id
    returns {
         client : s_mandt;
         carrname : s_carrname;
         connid : s_conn_id;
         cityfrom : s_from_cit;
         cityto : s_to_city;

        }

    implemented by method CL_EXAMPLE_AMDP=>GET_FLIGHTS;
```

Example 2:- **Table Function Implementation.**

The public ABAP class (AMDP class) in this example provides the AMDP method `get_flights`, which serves as the implementation of the table function `tab_function_example`. As with any other AMDP class, `cl_example_amdp` must implement the marker interface `IF_AMDP_MARKER_HDB`. The AMDP method `get_flights` implements the data selection using Native SQL code.

```
class cl_example_amdp definition public.

     public section.
        interfaces IF_AMDP_MARKER_HDB.
        class-methods get_flights  for table function tab_functi

     protected section.
     private section.
 endclass.


 class cl_example_amdp implementation.
```

```
    method get_flights by database function
        for hdb
        language sqlscript
        options read-only
        using scarr spfli.
        RETURN SELECT sc.mandt as client,
            sc.carrname, sp.connid, sp.cityfrom, sp.cityto
            FROM scarr AS sc
            INNER JOIN spfli AS sp ON sc.mandt = sp.mandt AND sc
            WHERE sp.mandt = :clnt AND
                  sp.carrid = :carrid
            ORDER BY sc.mandt, sc.carrname, sp.connid;
    endmethod.

endclass.
```

ABAP CDS Access Control

ABAP CDS enables access control based on a data control language (DCL). Access control in ABAP CDS further restricts the data returned from a CDS entity in ABAP CDS.

ABAP CDS access control is based on the following:

- CDS roles are defined using the DCL statement DEFINE ROLE. Currently, a CDS role is mapped to each user implicitly. This is why they are also known as mapping roles.
- Access conditions defined in a CDS role CDS entities. Access conditions can be the following:
  - Literal conditions that compare elements of a CDS entity with literal values.
  - PFCG conditions that associate elements of a CDS entity with authorizations in the SAP authorization concept.

If a CDS role is defined for a CDS entity, the access conditions are evaluated implicitly each time an object is accessed using Open SQL or using an SADL query (unless access control is disabled using the value #NOT_ALLOWED for the annotation @AccessControl.authorizationCheck. If access control is enabled, only that data comes which meets the access conditions.

Each CDS role is defined a separate piece of CDS source code. This CDS source code can only be edited in the ABAP Development Tools (ADT). When activated, the CDS role is characterized as a global internal object in ABAP Dictionary. The CDS source code of a CDS role is edited in a different editor from the CDS source code of a CDS entity (CDS view or CDS table function).

Imagine we have written a nice CDS view, e.g. as follows:

```
@AbapCatalog.sqlViewName: 'Z_T100_SABDEMOS'
@AccessControl.authorizationCheck: #CHECK.

define view z_t100_sabapdemos
  as select from t100
    { * }  where arbgb = 'SABAPDEMOS'
```

Now, we can create a **CDS role** (as shown below) in a DCL source code for above view.

```
@MappingRole: true

define role role_name {
  grant select on z_t100_sabapdemos
  where ( arbgb ) =  aspect pfcg_auth ( s_develop, objname,
                                        objtype = 'MSA
                                        actvt  = '03'
                     and sprsl= 'E' ; }
```

Now a question comes in mind:

What does this harmless looking code snippet using DEFINE ROLE can do?

A **CDS role** adds an **additional selection condition**, a so called **access condition**, to a **CDS view**! If you access a CDS view that is mentioned in a role, It implicitly consider the access conditions defined in each role.

In our case:

- A **literal condition** sprsl='E' restricts access to English only.
- A so called **PFCG condition** aspect pfcg_auth (s_develop ...) connects the CDS role to a classical authorization object s_develop and from that the CDS access control runtime generates an access condition that evaluates the authorizations of the current user for that object. Here, a **predefined aspect pfcg_auth** connects the authorization field objname to the view field arbgb. Additionaly, the user's authorization is checked if it complies with fixed values for authorization fields objtype and actvt.

Note: If you don't want any access restriction, you must decorate your view with the annotation @AccessControl.authorizationCheck: #NOT_ALLOWED. Then and only then CDS roles are ignored.

Post activation

When you activate a DCL source, SAP NetWeaver AS for ABAP generates the authorization views and fills the access control management tables with the required metadata. The roles are characterized as global internal objects in the ABAP Dictionary.

Notes

- Once created, CDS views can be used in ABAP programs using Open SQL read statements.
- A CDS database view is created for each CDS view and this database view supports only transparent tables, which means that pooled tables and cluster tables cannot be accessed using CDS views.
- For CDS views, CDS view enhancements are a separate way of making enhancements without making modifications.

To keep the focus on core objective (Core Data Services) of this blog, I tried to make it as small as possible.

Suggestions and questions are welcomed !!

Follow:-

- ABAP Core Data Services – Introduction (ABAP CDS view)
- ABAP Core Data Services – Part 2 (Types of CDS view)
- ABAP Core Data Services – Part 3 (Virtual Data Model Types)

Thank you.

Credits:-
ABAP CDS Development User Guide

Enhanced ABAP Development with Core Data Services (CDS)

ABAP CDS Access Control

Assigned tags

ABAP Development

SAP Access Control

SAP Access Control for SAP S/4HANA

SAP S/4HANA

ABAP CDS view

ABAP Core Data Service Views

beginner

View more... 〉

Similar Blog Posts

Getting Started with ABAP Core Data Services (CDS)

By Carine Tchoutouo Djomo Feb 01, 2016

CDS - One Concept, Two Flavors

By Horst Keller Jul 20, 2015

ABAP Core Data Services - Tooling FAQs

By Carine Tchoutouo Djomo Jun 22, 2016

Related Questions

ABAP CDS - Date Function - Function DATS_IS_VALID is unknown

By Former Member Mar 03, 2017

Core Data Services - Aggregation functions and general questions

By Former Member Sep 28, 2016

Other then ABAP, can CDS consumed by others reporting tools? e.g Bobj, Lumira

By Former Member Dec 03, 2015

3 Comments

You must be Logged on to comment or reply to a post.

DurgaPrasanth vemula
September 24, 2018 at 6:45 am

i have a requirement where i need to disable the field in the standard CDS View 'C_PurchaseReqnItem' and as per the below it is displaying the Field PurReqCreationDate in the "Manage Purchase Requisition Professional" Fiori App.

@UI: { fieldGroup: { qualifier: 'QuantityDate02', position: 20, importance: #HIGH } }

Document.PurReqCreationDate,

Now My Requirement i need to disable the field as i had to use annotation

@ObjectModel.readOnly: true

i had done the below code

@AbapCatalog.sqlViewAppendName: 'ZCPURREQNITM'

@EndUserText.label: '${ddl_source_description}'

extend view C_PurchaseReqnItem with zC_PurchaseReqnItem {

@ObjectModel.readOnly: true

Document.PurReqCreationDate }

But I am getting error "View field PURREQCREATIONDATE-C_PURCHASEREQNITEM already exists in parent object (DDL source)" could you please help how to handle this situation.

Like 0 | Share

Tushar Sharma | Blog Post Author
October 12, 2018 at 5:33 am

Hi *DurgaPrasanth vemula*,

Try using the Metadata extension view instead of Extend view and write all necessary annotations for particular field as per requirement.

Thanks,

Tushar Sharma

Like 0 | Share

Pavel Astashonok
December 1, 2020 at 1:46 pm

In the light of new CDS entity functionality (DEFINE VIEW ENTITY) released with ABAP on HANA 755 I see a lot of confusion between using *CDS entity* word alike in this blog for simple views/table funcs, and using CDS entities for real CDS entities after 2008 release. Possible it worth to correct old blogs about CDS for all the newcomers who is not yet familiar with CDS grain.

Like 1 | Share