Tushar Sharma

September 16, 2017 | 9 minute read

ABAP Core Data Services – Part 2(Types of CDS Views)

S.no	Topics
1.	Types of CDS views 1.1. Define View 1.1.1. Define View with Join 1.1.2. Define View with Association 1.1.3. Define View with Parameters 1.2. Extend View 1.3. Define Table functions

2.	More Blog's (Follow)
3.	Credits

This blog is continuation of my previous blog's :

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

ABAP Core Data Services – Part 1(ABAP CDS Entities) in this detailed introduction is given about the ABAP CDS Entities.

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

I, hope you have got the basic understanding about ABAP CDS views. In this blog, we'll see the different types of CDS views.

Let's Start !!

Types of CDS views

Let's see different types of ABAP based – CDS views in real world development scenario. To Create CDS view we need SAP development toolset in Eclipse (i.e ABAP development tool – ADT) [*To install ADT for Eclipse Folow 6.1.3 Installation Steps*]

Different types of CDS views (DDL Source)

1. Define View.
    - Define View with Join.
    - Define View with Association.
    - Define View with Parameters.
2. Extend View.
3. Define Table Function with Parameters.

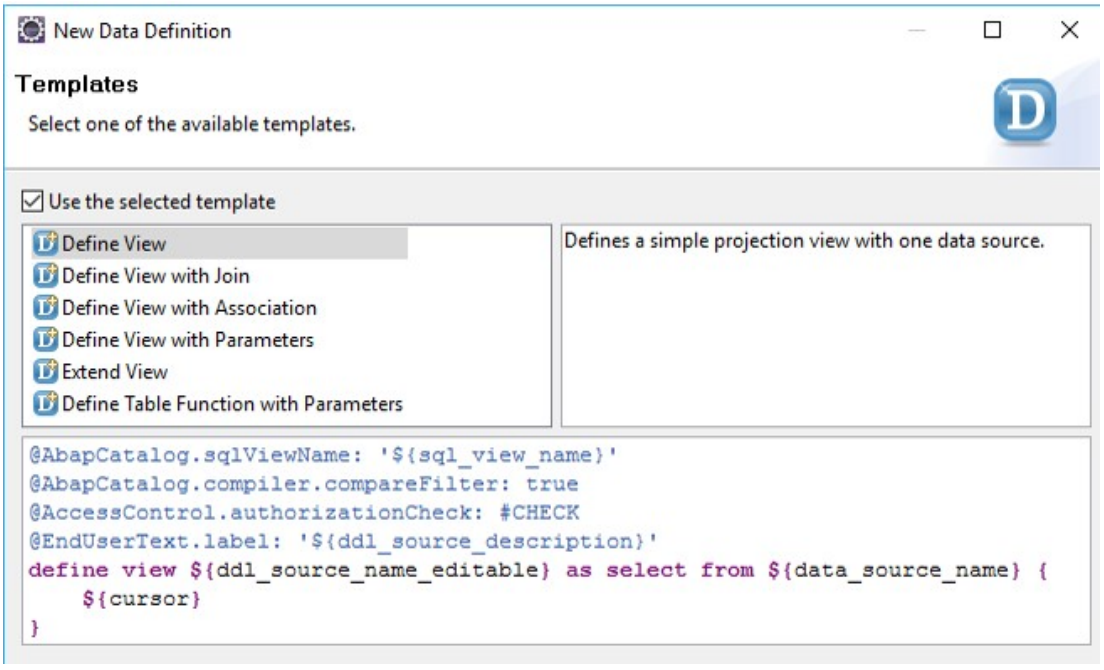


Figure 1: Basic View Templates

Define View

Defines a CDS view in the ABAP CDS in a CDS source code. A CDS is implemented using a query select_statement. The annotation AbapCatalog.sqlViewName must be specified before the view itself is defined using DEFINE VIEW.

Two objects are created for a CDS view that is defined using DEFINE VIEW. A name must be specified for each of the two objects:

- The name CDS_DB_VIEW of the CDS database view must be specified in quotation marks after the annotation @AbapCatalog.sqlViewName. This view is the technical foundation of the CDS view in ABAP Dictionary. The usual rules for ABAP Dictionary views apply to this name and it is not case-sensitive (it is transformed internally into uppercase letters). The associated SQL view is created under this name on the database. The name given to the database view can no longer be changed after the CDS view is transported into a follow-on system.

- The name cds_entity of the CDS entity is defined after the keywords DEFINE VIEW (DEFINE is optional). No quotation marks need to be specified. This name follows the rules of the CDS database view, but can have 30 characters. The CDS entity represents all properties of the CDS view.

```
ZCDS_VIEW   DDL source name
1  @AbapCatalog.sqlViewName: 'ZcdsView'   SQL view name
2  @AbapCatalog.compiler.compareFilter: true
3  @AccessControl.authorizationCheck: #CHECK
4  @EndUserText.label: 'Define View CDS_ENTITY'
5  define view zcds_View   CDS Entity name
6    as select from spfli   Data source (DDIC table)
7  {
8    key spfli.carrid,
9    key spfli.connid,
10       spfli.countryfr,
11       spfli.countryto
12 }
```

Figure 2: Example of Define View

- **Define View with Joins**

Defining a join between two data sources of a CDS view in ABAP CDS.using join expression.

Both inner and outer joins are possible:

- A join between two data sources using INNER JOIN or just JOIN selects all entries of the data sources whose fields meet the ON condition.
- A join between two data sources using LEFT OUTER JOIN selects all entries on the left side. A join between two data sources using RIGHT OUTER JOIN selects all entries on the right side. Entries that meet the ON condition have the same content as in the inner join. In entries that do not meet the ON condition, the elements on the right or left side have the null value that is set to a type-friendly initial value when the CDS view is used in Open SQL.

In nested join expressions, the order of the evaluation is specified by the arrangement of the ON conditions. From left to right, the most adjacent ON conditions are assigned to each JOIN and this expression is parenthesized implicitly. These implicit parentheses can be made explicit using actual parentheses, ( ). This is optional.

```
ZCDS_VIEW
1  @AbapCatalog.sqlViewName: 'ZcdsView'
2  @AbapCatalog.compiler.compareFilter: true
3  @AccessControl.authorizationCheck: #CHECK
4  @EndUserText.label: 'Define View CDS_ENTITY'
5
6  define view zcds View
7    as select from spfli
8      join         scarr on spfli.carrid = scarr.carrid
9  {
10   key spfli.carrid,
11   key scarr.carrname,
12   key spfli.connid,
13       spfli.countryfr,
14       spfli.countryto
15 }
```

Figure 3: Example of Define View with Join

In every join expression, a join condition cond_expr must be specified after ON. When specified, special rules apply to this condition.

```
Rules for conditions cond_exp in an ON condition of a join of a
in ABAP CDS:

1. All relational operators are allowed.
2. lhs expects a field of the data source data_source that repre
   the left side of the join.
3. A field of both data sources data_source of the join, a liter
   optional domain prefix, a parameter, or a session variable ca
   specified for rhs (with the exception of the operator LIKE).
4. At least one comparison for equality between a field of the l
   data source and a field of the right data source of the join
   be performed.
5. No path expressions or other expressions or function calls ca
   specified.
```

- **Define View with Associations.**

Defining an association of the name _assoc in a SELECT statement of a CDS view. An association connects the first elementary data source data_sources pecified as the initial data source (after FROM using the ON condition cond_exp) to the data source data_source specified as the target data source (in the definition of the association). The target data source cannot be built using joins.

An association of a SELECT statement can be accessed – in the same statement at all places where this is documented – by specifying the association name in path expressions. When a CDS view is activated with path expressions, the specified associations are converted to join expressions. The initial data source is shown on the left side and the target data source is shown on the right side. The ON condition of the association is added to the ON condition of the join. The type of join depends on the place where the path expression is used.

```
ZCDS_VIEW
1  @AbapCatalog.sqlViewName: 'ZcdsView'
2  @AbapCatalog.compiler.compareFilter: true
3  @AccessControl.authorizationCheck: #CHECK
4  @EndUserText.label: 'Define View CDS_ENTITY'
5
6  define view zcds_View
7    as select from spfli
8    association [1..1] to scarr as _scarr on $projection.carrid = _scarr.carrid
9  {
10   spfli.carrid,
11   _scarr.carrname,
12   spfli.connid,
13   spfli.countryfr,
14   spfli.countryto,
15   /*Associations*/
16   _scarr
17 }
```

_scarr is an alias name for table SCARR

Figure 4: Example of Define View with Association

When specifying the ON condition, the following special rules apply:

- Fields of the initial data source, which are specified in the ON condition, must also be listed in the SELECT list of the current SELECT statement. This ensures that a join expression can be built from the association (when used in a path expression). In the ON condition, the field name should be prefixed by $projection and not by the name of initial data source.
- The fields of the target data source must be prefixed in the ON condition by the name of the association (prefix assoc. separated by a period).

```
Rules for conditions cond_exp in an ON condition of
an association:

1. The relational operator IS [NOT] NULL is not allowed.
2. A field of one of the two data sources data_source of the
  association can be specified for lhs.
3. A field of one of the two data sources data_source of the
   association or a literal can be specified for rhs.
4. At least one comparison for equality between a field of the
   initial data source and a field of the target data source of
   the association must be performed.
```

```
5. No path expressions or other expressions or function calls
   can be specified.
```

- **Define View with Parameters.**

Defining a parameterized view with $parameters.pname: data type/data element in a SELECT statement of a CDS view. As a developer, you can now parameterize your CDS views. It means you can define one general views which produce context-specific result sets by using parameter values passed at execution time. This means that you do no longer need to create a view for each context!

Defines Input parameters pname1,pname2 in a CDS View in ABAP CDS in a comma separated list. Each input parameter must be typed with a data type parameter type.

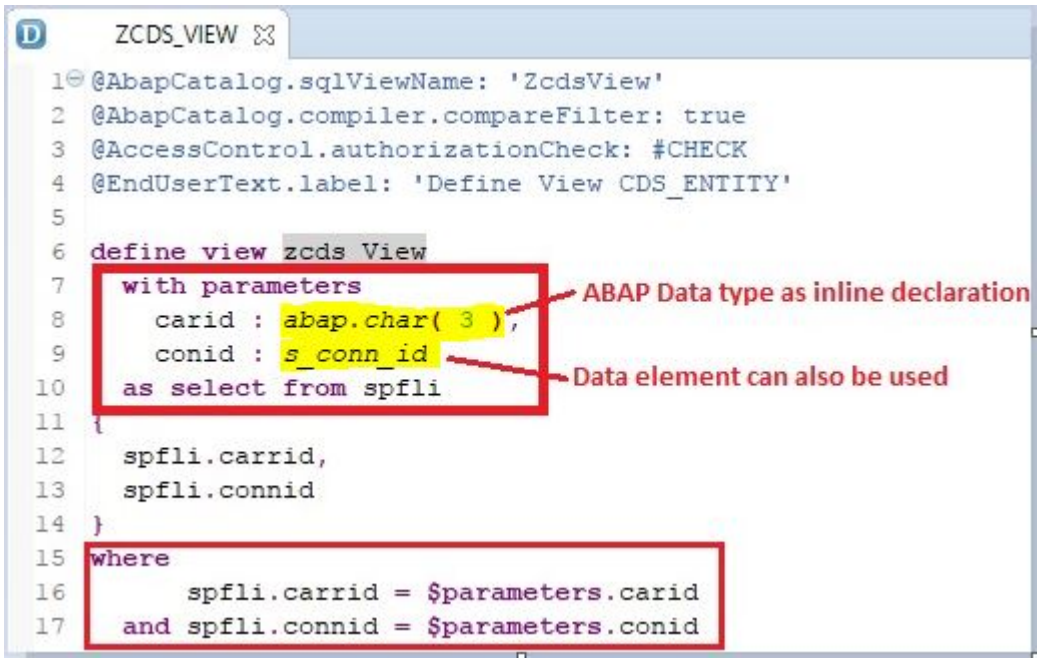


Figure 5: Example of Define View with Parameters

In above shown example, here in zcds_view CDS view carid and conid are used as parameters to get refined result based on filter values given by user at the time of execution. In WHERE clause the parameters are mapped to data source fields used in CDS view.

```
An input parameter called carid can be used as an operand in the
places in the SELECT statement of the view using the syntax $par

1. Right side of an expression cond_exp in a WHERE clause or HAV
2. Right side of an expression cond_exp in an ON condition in an
   or an association.
3. Right side of an expression cond_exp in a filter condition of
   expression.
```

Note:- The CDS views with parameters are not supported on all SAP certified databases – At least SAP HANA does support it 🙂 . However, the DDL of the ABAP CDS allows creating and accessing CDS views with parameters independent of the database system.

Post activation of Define View.

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

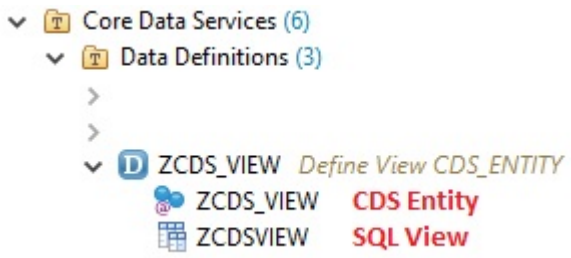


Figure 6: Post activation objects

a). The actual CDS entity (Zcds_View)

b). An SQL view (ZcdsView).

Extend View

Extend view is used to extend standard or custom CDS view without making any changes to the original CDS view. In SQL view of original CDS view the classical append view is created as .APPEND in ABAP Dictionary.(similar as append structure is used to extend DDIC tables.)

Extends an existing CDS view using a CDS view extension in CDS source code. The extension adds the following to the SELECT list of the available view without making changes:

- The elements of the specified extension list select_list_extension as view fields.
- Optional associations association1, association2, ... for the SELECT statement of the extended CDS view.

The annotation AbapCatalog.sqlViewAppendName must be specified before the view extension itself is defined using EXTEND VIEW.

```
ZCDS_VIEW   ZCDS_VIEW_EXT   ZCDSVIEW
1  @AbapCatalog.sqlViewAppendName: 'zview_ext'
2  @EndUserText.label: 'Extend view for zcds_view'
3
4  extend view zcds_View with Zcds_View_Ext
5  {   Original view that is extended
6    spfli.distance,                Extend view
7    spfli.distid as unit
8  }
9
```

Figure 7: Example of Extend View

In, extend view as shown above Zcds_view_ext is extend CDS view which is used for extension of zcds_view (original cds view) without making any changes to original view. After, activating the Extend view, spiral like symbol comes in original CDS view indicating that this view has been extended using other view as shown in figure 8.

```
ZCDS_VIEW   ZCDS_VIEW_EXT   ZCDSVIEW
1  @AbapCatalog.sqlViewName: 'ZcdsView'
2  @AbapCatalog.compiler.compareFilter: true
3  @AccessControl.authorizationCheck: #CHECK
4  @EndUserText.label: 'Define View CDS_ENTITY'
5
6  define view zcds_View
7    as select from spfli
8  {
9    spfli.carrid,
10   spfli.connid
11 }
12   zcds_View
13   Define View CDS_ENTITY

     Extended with
     Zcds_View_Ext
```

Figure 8: Original CDS view after extending using Extend view

As I mentioned above after extending any view using extend view it also adds classical append in SQL view in the ABAP dictionary. The difference between the SQL view before and after the extension can be seen in figure 9.

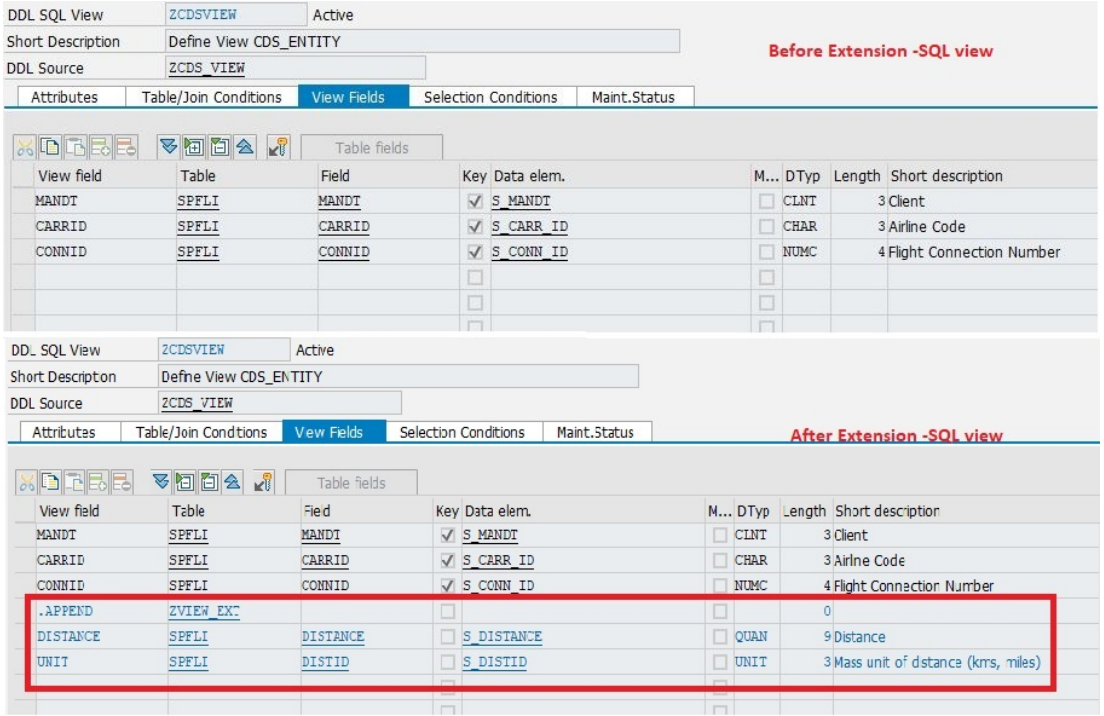
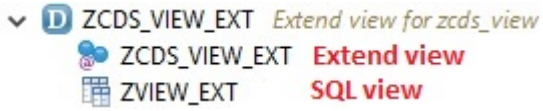


Figure 9: Comparison of SQL view before and after extending original CDS view.

Post activation of Extend View.

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



a). The extend CDS entity (Zcds_View_ext)

b). An SQL view (ZView_ext).

Note:-

The following CDS views cannot currently be extended:

- Views with an explicit name list.
- Views with aggregate expressions and a GROUP-BY clause.
- Views with a UNION clause for union sets.

The CDS view extensions themselves cannot be extended.

Define Table Functions with parameter.

A CDS table function is defined in CDS source code of a CDS data definition in the ABAP Development Tools (ADT) using the statement DEFINE TABLE_FUNCTION in the ABAP Core Data Services (CDS) DDL. A CDS table function includes the following:

- The CDS entity

  A CDS table function is declared as a CDS entity using DEFINE TABLE_FUNCTION. As a data type in ABAP Dictionary, the CDS entity represents a structured type with the elements of the CDS table function as components and can be used like any CDS entity.

- An AMDP function implementation

  CDS table functions are implemented in Native SQL in an AMDP method and the implementation is managed as an AMDP function by the AMDP framework in the database system.
  The AMDP method is specified after the addition IMPLEMENTED BY in the definition of the CDS table function using DEFINE TABLE_FUNCTION. It must be declared as a special AMDP function implementation for precisely one CDS table function using the addition FOR TABLE FUNCTION.

ZTEST_TABLEFUN | ZDEMO_AMDP_FUNCTIONS

```
1  @EndUserText.label: 'Test for Table funciton'
2  define table function ztest_tableFun
3  with parameters @Environment.systemField : #CLIENT
4                  clnt:abap.clnt
5  returns {
6              client:s_mandt;
7              carrname:s_carrname;
8              connid : s_conn_id;
9              cityfrom : s_from_cit;
10             cityto : s_to_city;
11 }
12 implemented by method zdemo_amdp_functions=>get_flights;
```

AMDP method

Figure 10: Example of Table function with parameters.

```
ZTEST_TABLEFUN    ZDEMO_AMDP_FUNCTIONS
ZDEMO_AMDP_FUNCTIONS ▸ GET_FLIGHTS
1  CLASS zdemo_amdp_functions DEFINITION
2    PUBLIC
3    FINAL
4    CREATE PUBLIC .
5    PUBLIC SECTION.
6
7      INTERFACES if_amdp_marker_hdb.
8      CLASS-METHODS get_flights FOR TABLE FUNCTION ztest_tablefun.
9
10   PROTECTED SECTION.
11   PRIVATE SECTION.
12
13 ENDCLASS.
14
15 CLASS zdemo_amdp_functions IMPLEMENTATION.
16
17   METHOD get_flights BY DATABASE FUNCTION
18                      FOR HDB
19                      LANGUAGE SQLSCRIPT
20                      OPTIONS READ-ONLY
21                      USING scarr spfli.
22     RETURN SELECT sc.mandt as client,
23              sc.carrname, sp.connid, sp.cityfrom, sp.cityto
24              from scarr as sc
25              inner join spfli as sp on sc.mandt = sp.mandt and sc.carrid = sp.carrid
26              order by sc.mandt, sc.carrname, sp.connid;
27
28   endmethod.
29
30 ENDCLASS.
```

Defining method for table function implementation

Syntax for defining the AMDP method for CDS Table function.

Figure 11: Definition of AMDP function for Table function with parameters.

Notes:-

- CDS table functions can only be used in a database system that supports AMDP.
- When a CDS table function is created, the CDS entity must be activated first, before the associated AMDP function implementation is created.
- When a CDS table function is transported, the CDS entity is first transported as part of the dictionary transport objects and then the AMDP function implementation as part of the ABAP transport objects. Depending on the size of the transport, there can be a considerable delay between these two phases where the CDS table function is not in a usable state.

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 Views).
- ABAP Core Data Services – Part 1 (ABAP CDS Entities).

- ABAP Core Data Services – Part 3 (Virtual Data Model Types)

Thank you.

Credits:-
ABAP CDS Development Guide

ABAP CDS – Data Definition

SAP Community Wiki – Core Data Services

Alert Moderator

Assigned tags

ABAP Development

SAP S/4HANA

ABAP CDS view

ABAP Core Data Service Views

beginner

Core Data Services (CDS)

sap hana

Similar Blog Posts

Getting Started with ABAP Core Data Services (CDS)

By Carine Tchoutouo Djomo Feb 01, 2016

Implementation Patterns for CDS Views in SAP S/4HANA at SAP TechEd 2017

By Carine Tchoutouo Djomo Jan 30, 2018

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

By Tushar Sharma Sep 09, 2017

Related Questions

SAP ABAP CDS @VDM.viewType usage.

By Prince Joshi Apr 19, 2020

ABAP CDS Views dynamic fields?

By End Stufe Feb 06, 2020

ABAP CDS View error with Annotation in Eclipse Editor.

By Con Pirzas Apr 15, 2020

6 Comments

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

Pushyami Koka
October 18, 2017 at 9:05 am

Hi Tushar,

Thanks for the article!! It's really helpful in understanding the basics from an ABAPer perspective.

I have a query here, Do you know how the memory consumption happens when we use very large CDS views(I almost have 15 tables in joins and unions and 330 fields in the output) ? View is working fine, however we have issue with very high memory consumption.

Is there any workaround or better way of implementing CDS views where memory consumption and response time can be less ?

Thanks in advance !!

Pushyami

Like 0 | Share

Tushar Sharma | Blog Post Author
November 14, 2017 at 7:49 am

Hi Pushyami,

Sorry, for replying late.

The memory consumption and about the performance, totally depends which Database you are using like SAP HANA will be much more faster w.r.t any other database.

May be this will help: SAP HANA Memory Usage

Thanks,

Tushar Sharma

Like 0 | Share

Sagar Muley
June 9, 2019 at 7:07 pm

Nice Blog

Like 0 | Share

pooja bakshi
July 14, 2019 at 11:45 am

Hi Tushar,

Your blog is very helpful.

But could you please add about cardinality in associations in detail.

also views with aggregations and Union can now be extended using

@AbapCatalog.viewEnhancementCategory: [#Projection, #Group_By, #Union]

Please correct me if i am wrong here, interested in learning 🙂

Thanks,

Pooja

Like 2 | Share

Tushar Sharma | Blog Post Author
July 29, 2019 at 9:56 am

Hi Pooja,

Thanks for feedback.

Yes, you are correct views having above annotation can be extended with aggregations & unions.

Regarding cardinality in associations, please check this.

Regards,

Tushar Sharma

Like 0 | Share

PavanKumar ReddyMaddhayyagari
August 31, 2021 at 6:31 am

Nice blog ,Thanks

Like 0 | Share

Find us on

Privacy	Terms of Use
Legal Disclosure	Copyright
Trademark	Cookie Preferences
Newsletter	Support