S11 - BW Modeling - CDS Views

CDS Views	1. Datasource based on CDS View - Delta (Generic w/o ODQs) 2. Datasource based on CDS View - Delta (Generic with ODQs) 3. Enhancing datasource based on CDS views using extension CDS Views.	5th Sep: 8:30 AM - 10:30 AM

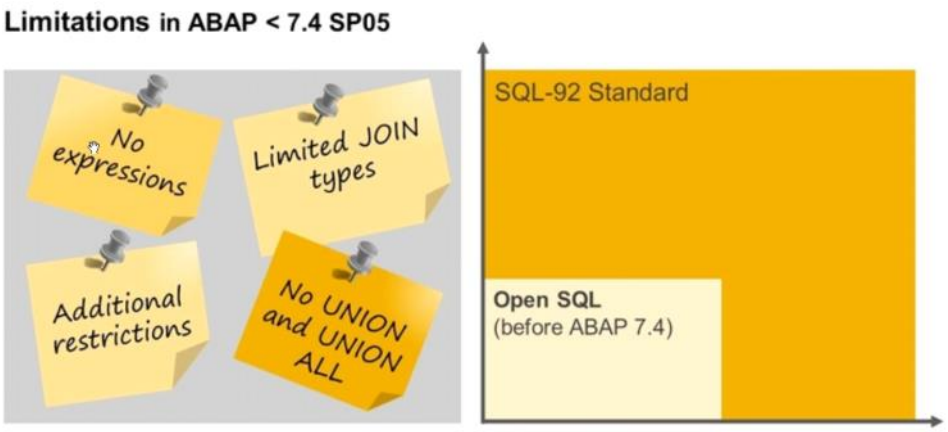
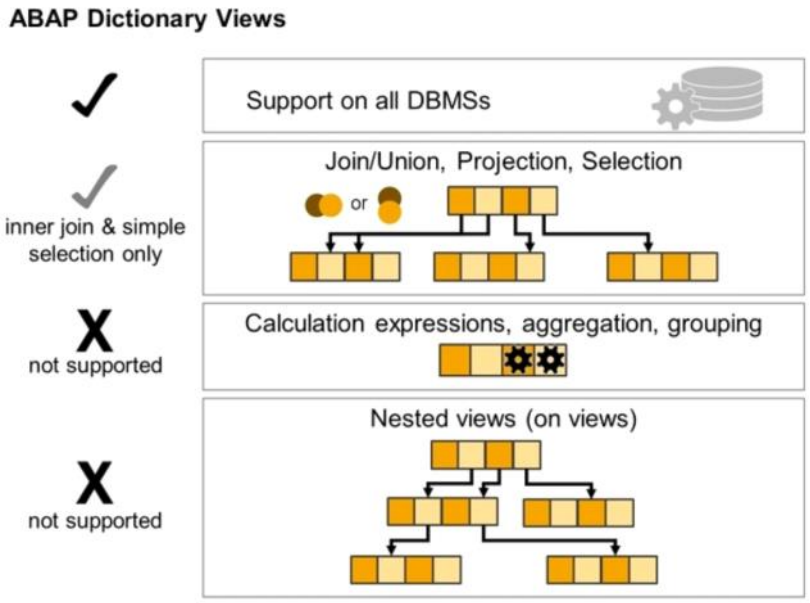
Q & A - Pre Session

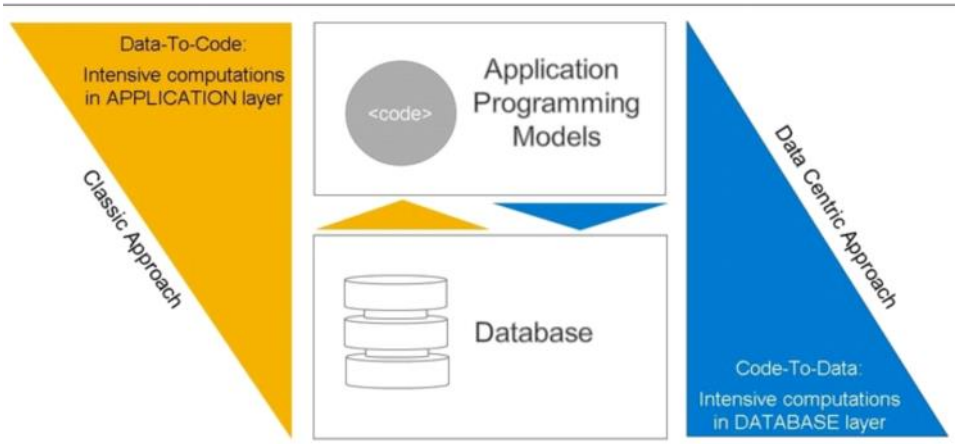
Shakthi	When and Why do we use field level annotations and view level annotations. Any practical examples? What are AMDP scripts? When do we use them? AMDP vs CDS. Can you explain about @AbapCatalog.compiler.compareFilter? What are path expressions? How to make 'Ctrl + <space>' work in eclipse? What are the commonly/ frequently used used Annotations? What is Access control? When do we opt for CDS view based extraction? Difference between Key_fields and the fields managed in 'Manage Key' in ADSO
Vijay	Any performance considerations we need to keep in mind while we extract data using CDS. Is it the same approach as earlier we choose for Full or Delta extraction? what about Psudo deltas?

How to find CDS Views

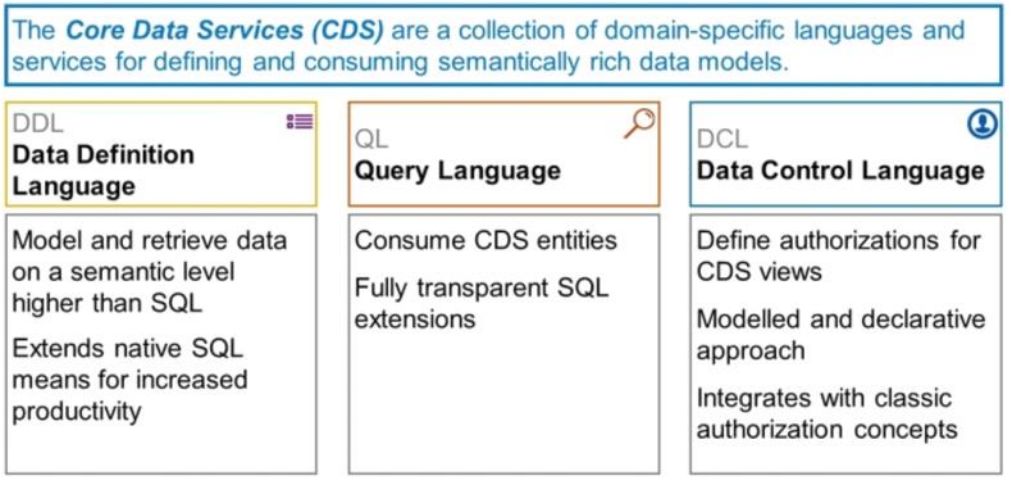
For a particular table	▪ Table DD26S. Pass base table name in column 'TABNAME (Basis Table)' Get the SQL View Name ▪ Table RSODPABAPCDSVIEW Pass the SQL view name in column 'SQLVIEWNAME' Get DDL Source file name and CDS View name ▪ Table DD25T. Pass the SQL view name in column 'VIEWNAME' Get the Description from column DDTEXT
Annotations: All	ABDOC_CDS_ANNOS SELECT * FROM ABDOC_CDS_ANNOS WHERE SPRAS = 'E'
Annotations for CDS View	CDSVIEWANNOPOS Select * from CDSVIEWANNOPOS where CDSNAME = 'I_MATERIAL' AND ANNOTATIONNAME LIKE '%.%'
Annotations for CDS View Fields	select * from CDS_FIELD_ANNOTATION WHERE CDSNAME = 'I_MATERIAL' and ANNOTATIONNAME LIKE '%.%'

CDS Views - Introduction

What is CDS	Combination of: 1. Language (OpenSQL) 2. Objects (CDS Views, CDS Tables, CDS functions)
What is CDS View	1. SELECT Query on tables. 2. Based on new ABAP OpenSQL Syntax. 3. Achieves Code pushdown to Database. 4. Embedded Analytics = Set of CDS Views 5. Semantics - Annotations (@..) 6. Association - Joins on Demand 7. Expressions 8. Parametrized
What was the need for CDS Views?	1. Limitations in ABAP  2. Limitations of ABAP Dictionary views:  3. Use of HANA DB capabilities and to push code to data:



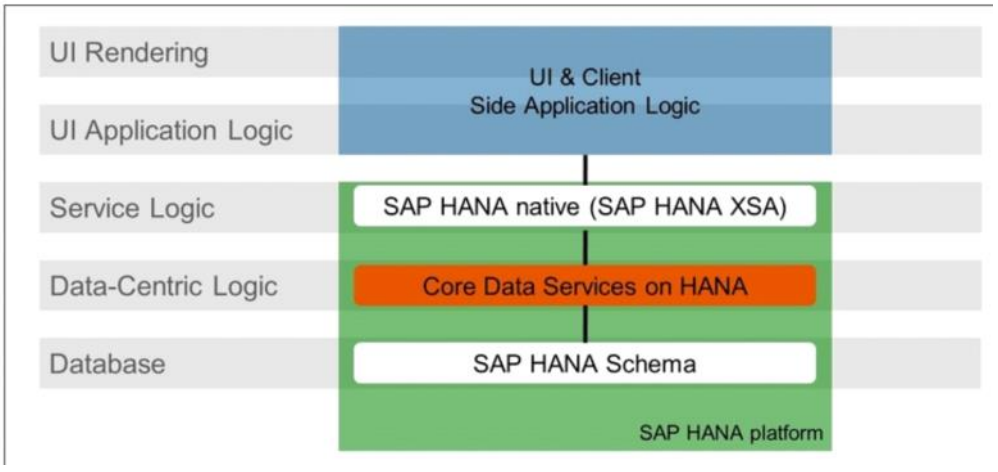
CDS



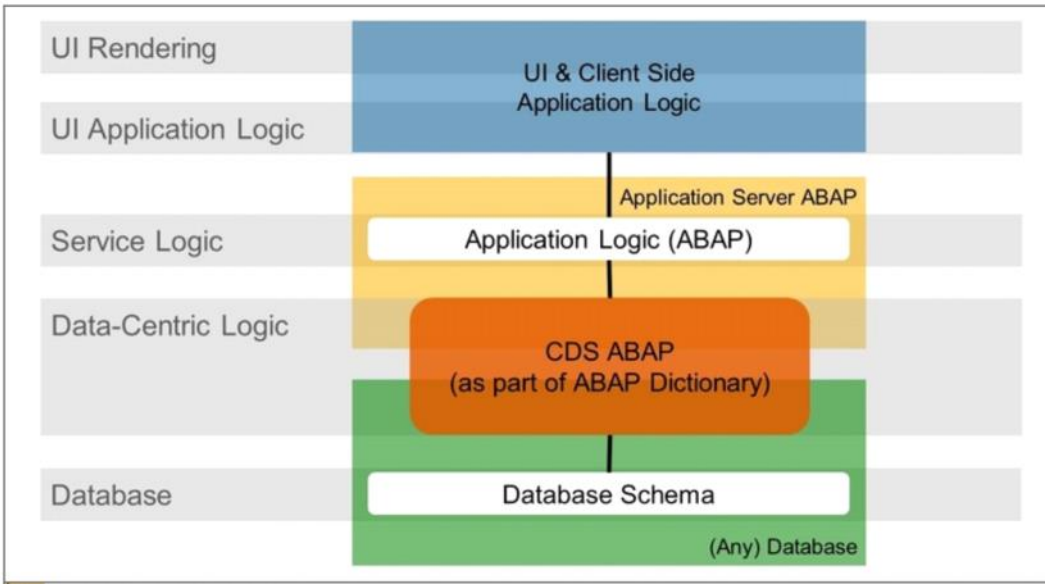
HANA CDS vs ABAP CDS

HANA CDS:

SAP HANA Core Data Services



ABAP CDS:

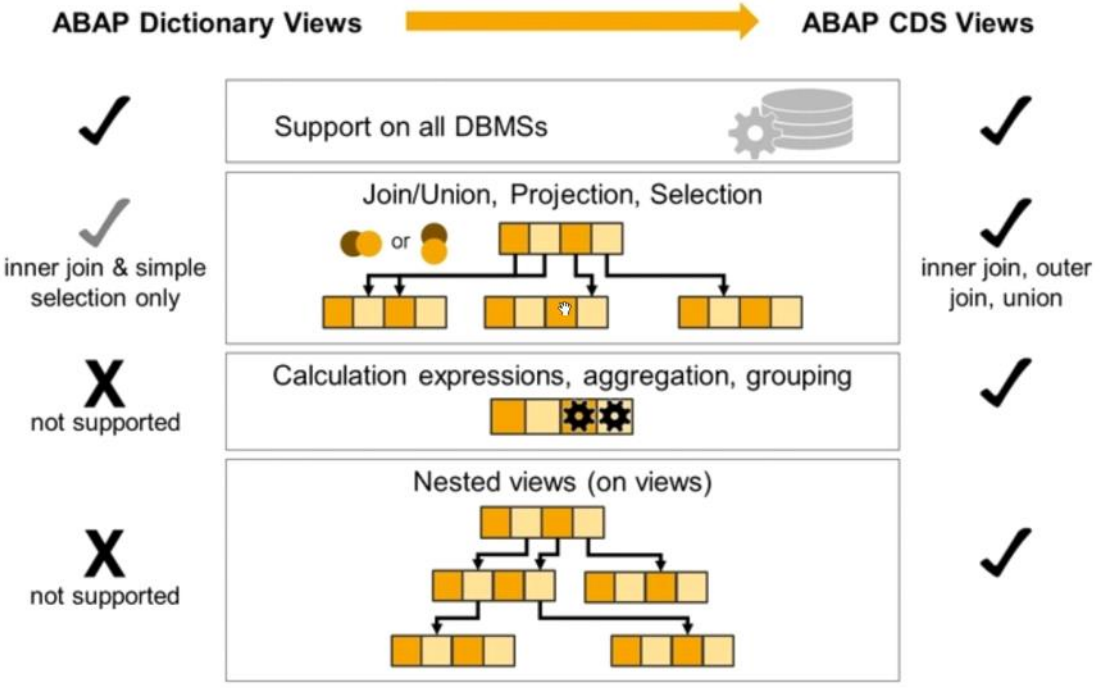
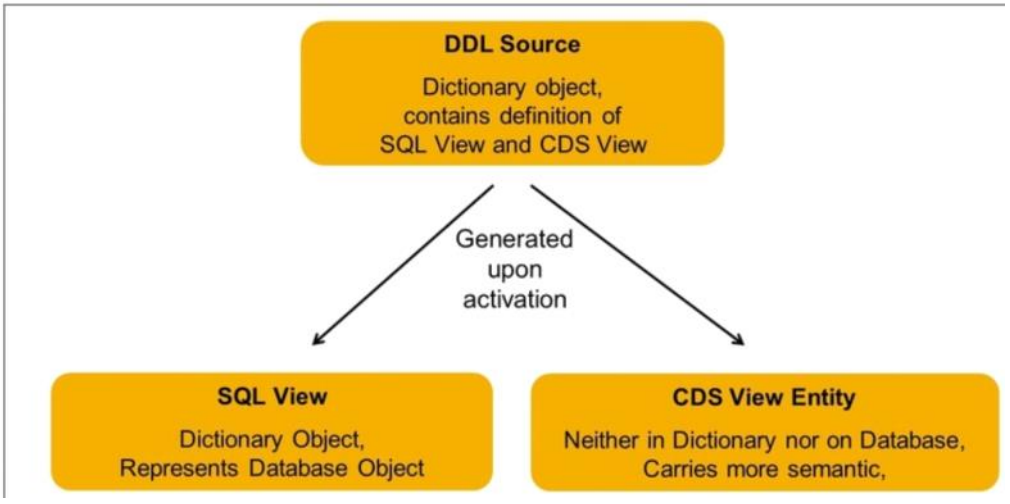


To take advantage of SAP HANA for application development, SAP introduced a new data modeling infrastructure known as core data services (CDS). This infrastructure is available in two flavors:

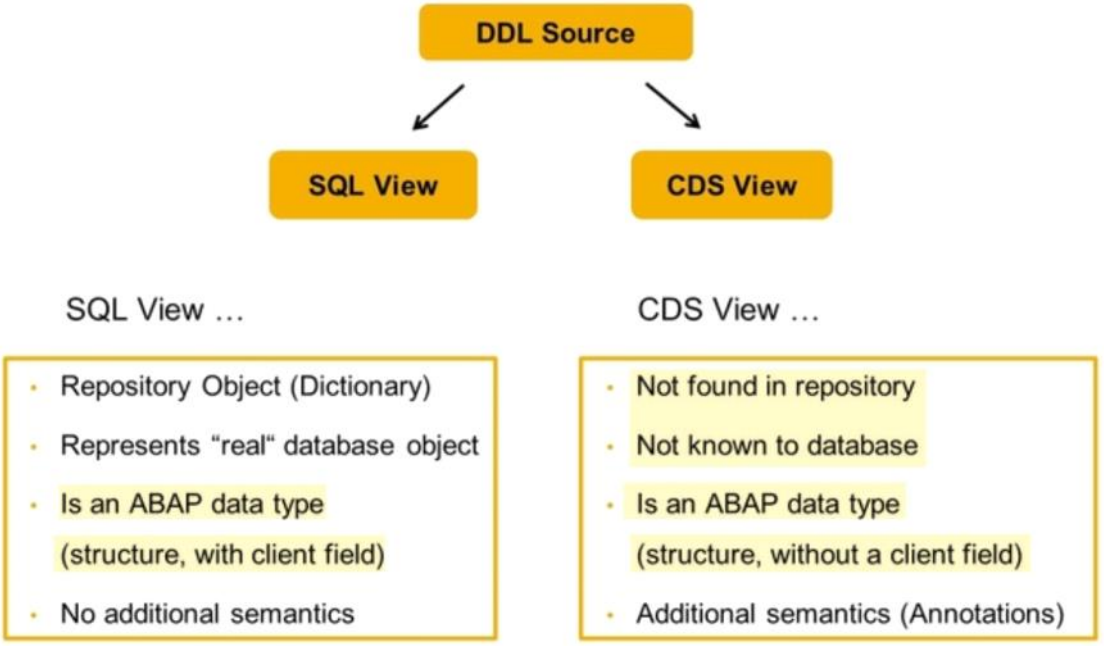
1. SAP HANA CDS provided by SAP HANA XS/XSA application server
2. SAP ABAP CDS provided by SAP NetWeaver ABAP in combination with SAP HANA

The framework relevant for SAP BW/4HANA focuses on ABAP CDS views and for this reason all following details are related this version of CDS. With ABAP CDS, data models are defined on the application server but they are processed and consumed on the database server. CDS also offers capabilities beyond the traditional data modeling tools, including support for conceptual modeling and relationship definitions, built-in functions, and extensions. The modeling concept is also fully implemented in SAP NetWeaver AS ABAP (also applies for SAP BW/4HANA AS ABAP), enabling developers to work in the ABAP layer with ABAP development tools while the code execution is pushed down to the database. The use of this technology means that no installation or activation of technical content is required, nor is there any need to load data. The data is available in real time and processed by a virtual data model without own data persistence.

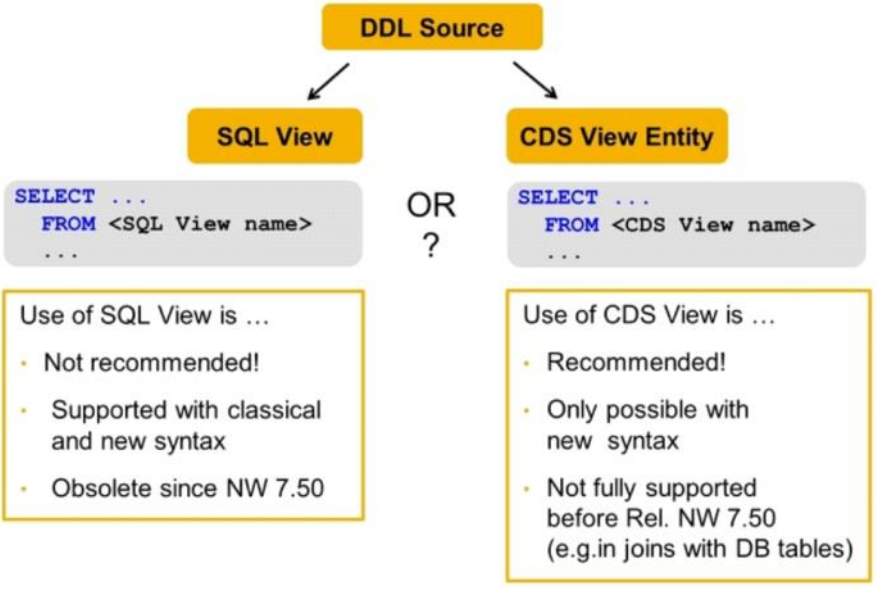
The 3 main musketeers of CDS	**Expressions** Used for calculations and queries in the data model **Associations** On a conceptual level, replacing joins with simple path expressions in queries **Annotations** To enrich the data models with additional (domain specific) metadata

Other Salient Points	▪ **Database Independent** – Use CDS ABAP with any database supported by SAP ▪ **Advanced View Building** – CDS Views provide much more SQL features than classical Dictionary Views ▪ **Annotations to add Semantic Information** – Add end user texts, currency keys, buffer settings, ... – Add semantic information for consumers (analytics, OData, SAP UI5, ...) ▪ **Implicit Authorization Checks** – Define authorization rules for CDS Objects ▪ **Associations instead of Joins** – Define relations between CDS Objects that will be translated into joins ▪ **CDS Table Functions** – Views based on scripted coding (Currently only supported for SAP HANA)
ABAP DDIC Views vs ABAP CDS Views	
CDS related objects	DDL Source, SQL View, and CDS View 

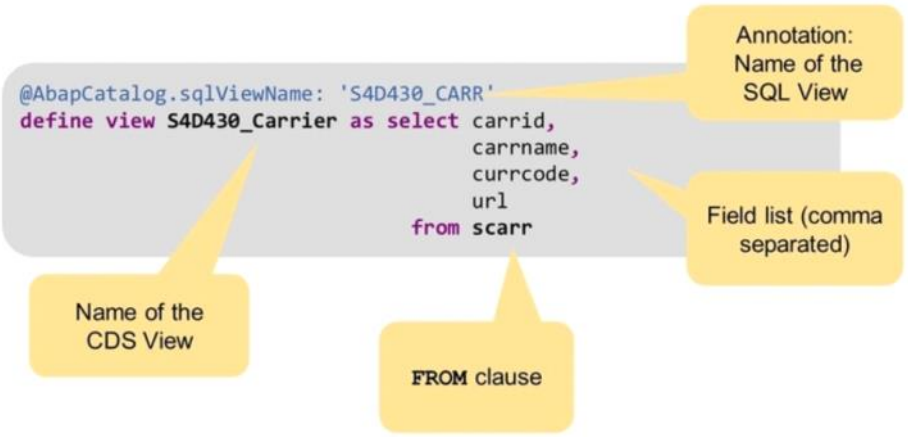
- A CDS View is defined in a DDL Source, which is a new type of repository object.
- There is no editor for DDL sources in the classical ABAP workbench. This new type of repository object has to be analyzed and developed in ABAP Development Tools in Eclipse.
- Upon activation of a DDL Source, two objects are created: the SQL View and the CDS View. Neither of them can be edited directly.
- The SQL View is visible as an object in the ABAP Dictionary where it cannot be edited and only reveals a fraction of the information available in the DDL source. It serves as a representative of the database object.
- The CDS View carries more semantics than its SQL view. It is not created on the Database and it is not visible in the ABAP Dictionary. It can, however, be consumed via open SQL. The new Open SQL syntax introduced with NW 7.40 SP5 is required to access CDS Views.



- The name of the CDS entity is specified after the DEFINE VIEW statement. The maximum length is 30 characters.
- It is recommended, though not technically necessary, that the name of the DDL source and the name of the CDS view are identical.
- The name of the SQL View is specified after the Annotation @ABAPCatalog.sqlViewName.
- It has to be different from the name of the CDS entity. Like for any Dictionary view, the maximum length is 16 characters

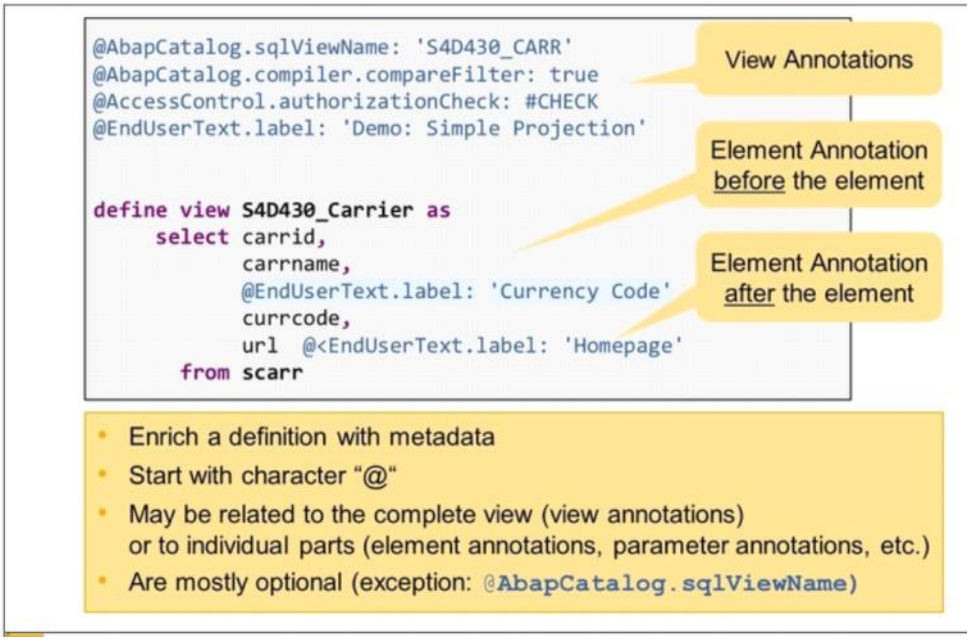


Syntax of CDS View



Annotations

Annotations



A CDS annotation (or annotation for short) enriches a definition in the ABAP CDS with metadata.

An annotation is identified by a simple or structured name after a leading character "@" or "@<".

Depending on its scope, an annotation can be found in different locations within the CDS Source.

View annotations

Relate to the view itself and are placed before the define view statement.

Element annotations

Relate to elements in the field list and can be found before or after the element.

Hint:
Element annotations after the element begin with leading characters "@<" .

SAP uses a set of predefined SAP annotations. Most of them are optional. But there is one exception: @AbapCatalog.sqlViewName is mandatory in every CDS View definition.

Associations

1. Join on Demand

Associations will only be triggered when user would access the required data which needs the Association of tables. For

example, your CDS view has 4 Associations configured and user is fetching data for only 2 tables, the ASSOICATION on other 2 tables will not be triggered and the system would return the results quickly, so it enables really high turn-around time as compared to regular SQL JOINS.

2. Association vs Join

View Definition with Association

```
@AbapCatalog.sqlViewName: 'S4D430_ASSO1'
define view S4d430_Association_1 as select
  from spfli association to scarr
    on spfli.carrid = scarr.carrid
   { key carrid,
     key connid,
         scarr.carrname
   }
```

Association target

Name of data source mandatory for all fields from association target

View Definition with Join

```
@AbapCatalog.sqlViewName: 'S4D430_JOININN'
define view S4d430_JOIN_INNER as select
  from spfli inner join scarr
    on spfli.carrid = scarr.carrid
     { key spfli.carrid,
       key connid,
           carrname
     }
```

Name of data source only mandatory for non-unique field names

3. Naming Convention

Hint:
It is recommended, but not a fixed rule, that names of associations begin with character "_". This corresponds to the naming rules for associations in OData.

A meaningful name for the association further improves the readability of the view definition.

Exposing Associations: (No join is executed at the first go)

-EXPOSED Association

```
1 @AbapCatalog.sqlViewName: 'ZSQL_VIEW_ASSTN'
2 @AbapCatalog.compiler.compareFilter: true
3 @AbapCatalog.preserveKey: true
4 @AccessControl.authorizationCheck: #CHECK
5 @EndUserText.label: 'CDS View with Associ
6 define view ZCDS_VIEW_ASSOCIATIONS as sel
7 association [1] to spfli as _flights
8     on sf.carrid = _flights.carrid {
9     //sf
10    key sf.carrid,
11    sf.connid,
12    sf.fldate,
13    sf.price,
14    sf.seatsocc_b,
15    sf.seatsmax_f,
16    sf.seatsocc_f,
17
18    _flights // Make association public
19 }
```

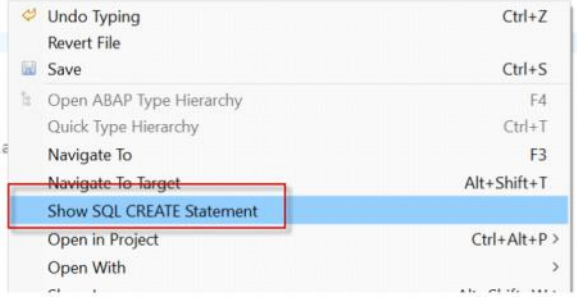
Similar to Join, we need key fields to Associate 2 different tables.

The key field which we used to Associate must be part of the selection

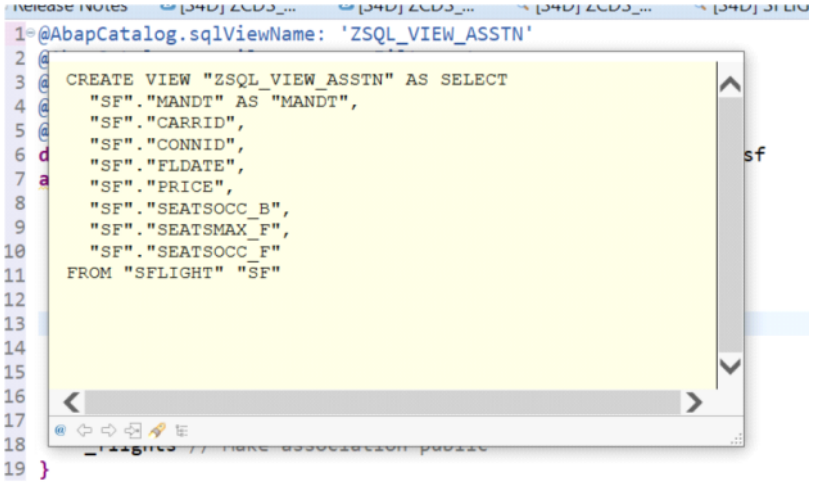
Make Association Public i.e. Expose the association. This will not create any Join beforehand but do it need basis.

Right click and select 'Show SQL CREATE Statement;

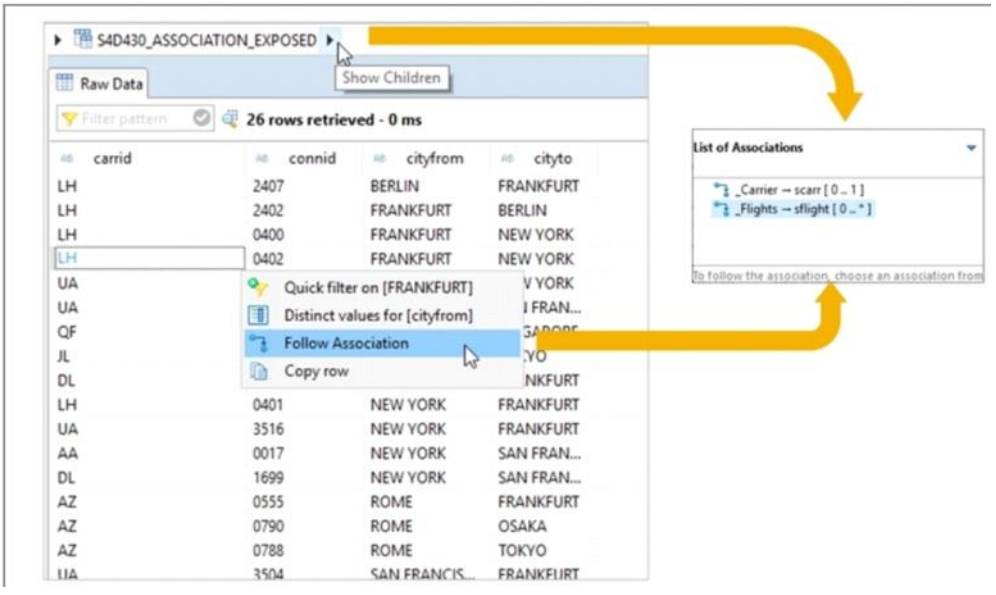
```
1 @AbapCatalog.sqlViewName: 'ZSQL_VIEW_ASSTN'
2 @AbapCatalog.compiler.compareFilter: true
3 @AbapCatalog.preserveKey: true
4 @AccessControl.authorizationCheck: #CHECK
5 @EndUserText.label: 'CDS View with Association concept'
6 define view ZCDS_VIEW_ASSOCIATIONS as select from sflight as sf
7 association [1] to spfli as _flights
8     on sf.carrid = _flights.carrid {
9     //sf
10    key sf.carrid,
11    sf.connid,
12    sf.fldate,
13    sf.price,
14    sf.seatsocc_b,
15    sf.seatsmax_f,
16    sf.seatsocc_f,
17
18    _flights // Make associa
19 }
```



As you can see, NO Join is created;



Exposed Associations in Data Preview



Now if we include fields from association:

```
1 @AbapCatalog.sqlViewName: 'ZSQL_VIEW_ASSTN'
2 @AbapCatalog.compiler.compareFilter: true
```

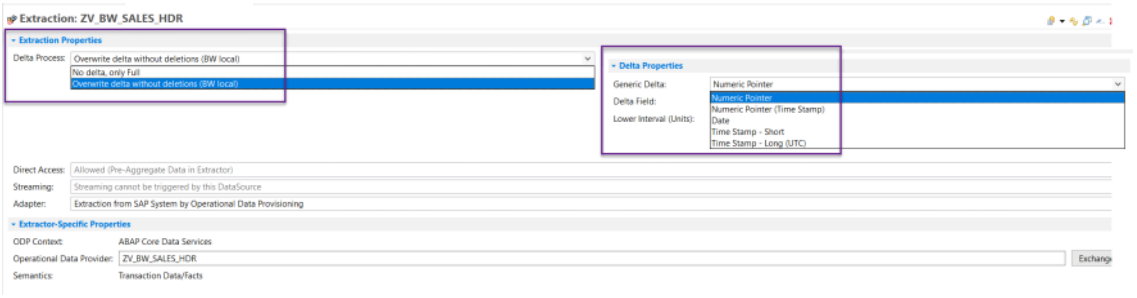
```
1  @AbapCatalog.sqlViewName: 'ZSQL_VIEW_ASSTN'
2  @AbapCatalog.compiler.compareFilter: true
3  @AbapCatalog.preserveKey: true
4  @AccessControl.authorizationCheck: #CHECK
5  @EndUserText.label: 'CDS View with Association concept'
6  define view ZCDS_VIEW_ASSOCIATIONS as select from sflight as sf
7  association [1] to spfli as _flights
8      on sf.carrid = _flights.carrid {
9      //sf
10     key sf.carrid,
11     sf.connid,
12     sf.fldate,
13     sf.price,
14     sf.seatsocc_b,
15     sf.seatsmax_f,
16     sf.seatsocc_f,
17
18     _flights.airpfrom, // Make association public
19     _flights.airpto
20 }
```

The join will be executed

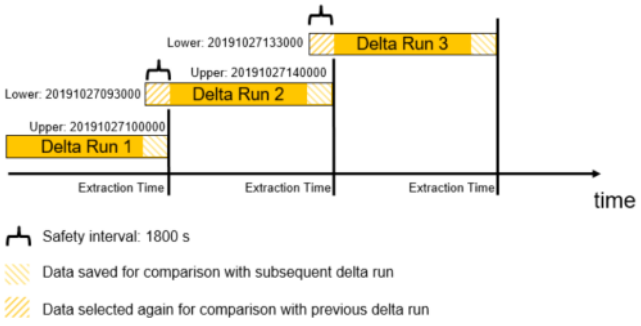
```
1  @AbapCatalog.sqlViewName: 'ZSQL_VIEW_ASSTN'
2  @
3  @
4  @
5  @
6  d
7  a
8
9
0
1
2
3
4
5
6
7
8
9
0 }
```

```
CREATE VIEW "ZSQL_VIEW_ASSTN" AS SELECT
  "SF"."MANDT" AS "MANDT",
  "SF"."CARRID",
  "SF"."CONNID",
  "SF"."FLDATE",
  "SF"."PRICE",
  "SF"."SEATSOCC_B",
  "SF"."SEATSMAX_F",
  "SF"."SEATSOCC_F",
  "=A0"."AIRPFROM",
  "=A0"."AIRPTO"
FROM "SFLIGHT" "SF" LEFT OUTER MANY TO ONE JOIN "SPFLI" "=A
  "SF"."MANDT" = "=A0"."MANDT" AND
  "SF"."CARRID" = "=A0"."CARRID"
)
```

CDS View - Delta Functionality

3 ways to do CDS View Delta	1. Generic Delta (without annotations and without using ODQs) 2. Annotations based delta using ODQs. 3. CDC based delta using SLT
Generic Delta	Controlled via DTP properties. 2 properties: 1. Delta Process 2. Delta Field (field to identify the change)  No Annotations required. ODQs not used at S4 side.

Annotations based delta using ODQs.	Delta activated via special annotations:
	
	@Analytics: {dataCategory: #FACT,
	dataExtraction: {
	enabled: true,
	delta.byElement: {
	name: 'LastChangedAt',
	maxDelayInSeconds: 1800,
	detectDeletedRecords: true
	}
	}
	}
	Streaming process chain supports the delta functionality.
	Default daemon check is 15 min Period and 15 seconds Takt time.
	For non-real-time delta subscriptions new records according to the delta criterion are directly pulled from the CDS view during extraction.
	
	Support Safety Interval via another set of special annotations:
	
	maxDelayInSeconds : 600
	
	Support identifying deleted records via another set of special annotations:
	
	detectDeletions : true
	
	Here the system stores the extracted records in a separate table.
	The next delta extraction records are compared to this table and records not found are marked as deleted.
	
	Analytics Annotations
	https://help.sap.com/viewer/cc0c305d2fab47bd808adcad3ca7ee9d/7.5.9/en-US/c2dd92fb83784c4a87e16e66abeeacbd.html

Safety Interval	The concept exists to make sure no data is missed from delta. Upper Safety Limit: The seconds subtracted from the time of delta trigger to have the 'Upto' time. Lower Safety Limit: The seconds subtracted from the previous delta's trigger to get the 'From' time. 
Delta relevant fields in ODP	0RECORDMODE ODP Calculation Based on 2 fields: 1. ODQ_CHANGEMODE Value | Description C | New Record (= Recordmode N) U | The data record was changed in the source. Depending on the delta procedure, the delta can contain up to two data records for one changed record in the source D | The data record was deleted from the source. 2. ODQ_ENTITYCNTR 0RECORDMODE is calculated based on these 2 fields: ODQ_CHANGEMODE | ODQ_ENTITYCNTR | RECORDMODE | Description

C	1	N	New-Image (i.e. After-Image without a preceding Before-Image)
U	-1	X	Before-Image (summable non-key components need to have reversed sign)
U	1		After-Image
D	-1	R	Reverse-Image (i.e. Before-Image without a subsequent After-Image)
D	0	D	(Incomplete) Delete-Image (i.e. only key components need to be specified)
U	0	A	Additive Image (Aggregation of Before- and After-Image for summable non-key components; After-Image for other non-key components)

CDS Performance Annotations

@ObjectModel.usageType.serviceQuality

quality of service with respect to the expected performance of the CDS view

@ObjectModel.usageType.dataClass

type of data in CDS view (transactional data, master data, ...)

@ObjectModel.usageType.sizeCategory

set of data whichhas to be searched through in order to compute the result set

CDS views: performance annotations

serviceQuality	Each CDS view shall be assigned to one of the following quality categories: **A**: the view may be consumed within business logic for **high volume transactions** or background processing **B**: the view may be consumed within business logic for **transactions** or background processing **C**: the view may be consumed **from the UI** in transactions for single object retrieval **D**: the view may be consumed for **analytical reporting** **X**: the view is built to **push down application code** to HANA
sizeCategory	Each CDS view shall have assigned a size category. The size category enables the consumer to judge the possible result set. It reflects the number of rows that has to be searched through to get a result. The labels correspond to the following size categories (expected number of rows in customer production systems): **S** < 1000, **M** < 100.000, **L** < 10.000.000, **XL** < 100.000.000, **XXL** > 100.000.000
dataClass	To support the decision on cache strategies for higher layers and to enable client side statement routing using these caches, each CDS view shall have assigned a data class. The different data classes correspond to different life time cycles. **TRANSACTIONAL** data is written or changed in high volume transactions **MASTER** data is read, but not written or changed in high volume transactions **ORGANIZATIONAL** data describes the organizational structure of a company and its business processes **CUSTOMIZING** data describes how a concrete business process is executed at the customer **META** data specifies how the system is configured or describes the technical structure of entities **MIXED** data shall be chosen if the CDS-View contains tables with several different of the above types

CDS View performance guidelines

1. Left outer joins.
2. No join on calculated columns.
3. User Parameters and where conditions in the CDS Views.
4. Take only required columns.
5. Use performance related annotations, wherever possible

CDS View - System Demo

CDS Views - Full extraction

*[S4H] ZI_BW_SALES_HDR

```
@AbapCatalog.sqlViewName: 'ZV_BW_SALES_HDR'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Tran: Sales Header'

@Analytics.dataCategory: #FACT
@Analytics.dataExtraction.enabled: true

define view ZI_BW_SALES_HDR
  as select from I_SalesDocumentBasic
{
    //I_SalesDocumentBasic
    key SalesDocument,
    SDDocumentCategory,
    SalesDocumentType,
    LastChangeDate,
    LastChangeDateTime,
    LastCustomerContactDate,
    SalesOrganization,
    DistributionChannel,
    OrganizationDivision,
    SalesGroup,
    SalesOffice,
    SoldToParty,
    AdditionalCustomerGroup1,
    AdditionalCustomerGroup2,
    AdditionalCustomerGroup3,
    AdditionalCustomerGroup4,
    AdditionalCustomerGroup5,
    CreditControlArea,
    CustomerRebateAgreement,
    SalesDocumentDate,
    SDDocumentReason,
    SDDocumentCollectiveNumber,
    CustomerPurchaseOrderType,
    CustomerPurchaseOrderDate,
```

Code Blocks

CDS View	@AbapCatalog.sqlViewName: 'ZV_SALES_ITM'
	@AbapCatalog.compiler.compareFilter: true
	@AbapCatalog.preserveKey: true
	@AccessControl.authorizationCheck: #CHECK
	@EndUserText.label: 'Tran: SD Sales Item'
	
	@Analytics: {dataCategory: #FACT,
	dataExtraction: {
	enabled: true,
	delta.byElement: {
	name: 'LastChangedAt',
	maxDelayInSeconds: 1800,
	detectDeletedRecords: true
	}
	}
	}
	define view ZI_BW_SALES_ITEM
	as select from vbap as p
	left outer join vbak as k on p.vbeln = k.vbeln
	{
	key p.vbeln as SalesDoc,
	key p.posnr as SalesItem,
	p.matnr as Material,
	
	@Semantics.quantity.unitOfMeasure: 'SalesUnit'
	p.kwmeng as OrderQty,
	
	@Semantics.unitOfMeasure: true
	p.vrkme as SalesUnit,
	
	@Semantics.systemDateTime.lastChangedAt: true
	k.upd_tmstmp as LastChangedAt
	}

Sales Header	@AbapCatalog.sqlViewName: 'ZV_SALES_HDR'
	@AbapCatalog.compiler.compareFilter: true
	@AbapCatalog.preserveKey: true
	@AccessControl.authorizationCheck: #CHECK
	@EndUserText.label: 'Tran: SD Sales Header'
	@Analytics: {
	dataCategory: #FACT,
	dataExtraction.enabled: true
	}
	define view ZI_BW_SALES_HDR
	as select from I_SalesDocument as SD //I_SalesDocument
	{
	
	//I_SalesDocument
	
	key SD.SalesDocument,
	SD.SDDocumentCategory,
	SD.SalesDocumentType,
	SD.CreatedByUser,
	SD.LastChangedByUser,
	SD.CreationDate,
	SD.CreationTime,
	SD.LastChangeDate,
	cast(cast(substring(cast(SD.LastChangeDateTime as abap.char(25)),1,14) as abap.numc(14)) as abap.dec (15, 0)) as LastChangedAt,
	SD.SalesOrganization,
	SD.DistributionChannel,
	SD.OrganizationDivision,
	SD.SoldToParty,
	SD.SalesDocumentDate,
	SD.FiscalYear,
	SD.FiscalPeriod
	}

Header Extn	@AbapCatalog.sqlViewAppendName: 'ZV_SALES_HDR_EXT'
	@EndUserText.label: 'Tran: SD Sales Header Extension'
	extend view ZI_BW_SALES_HDR with ZI_BW_SALES_HDR_EXT
	association to kna1 as _kn
	on SD.soldtoparty = _kn.kunnr
	{
	_kn.name1,
	_kn.land1
	}