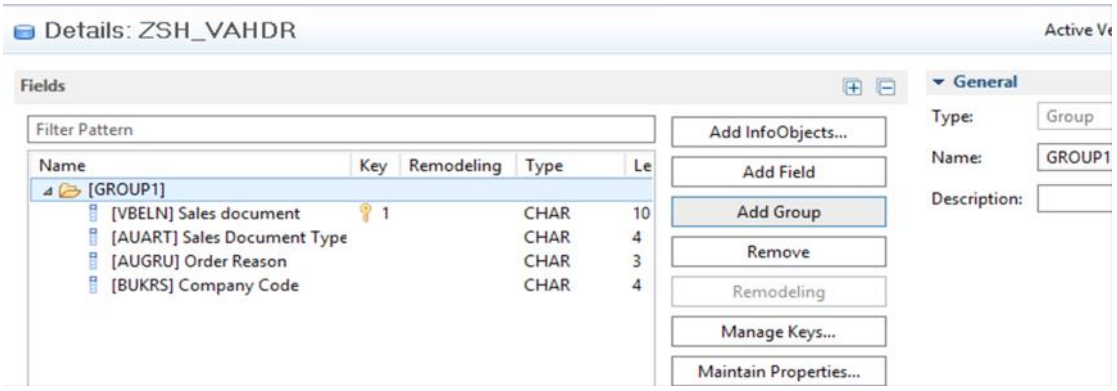
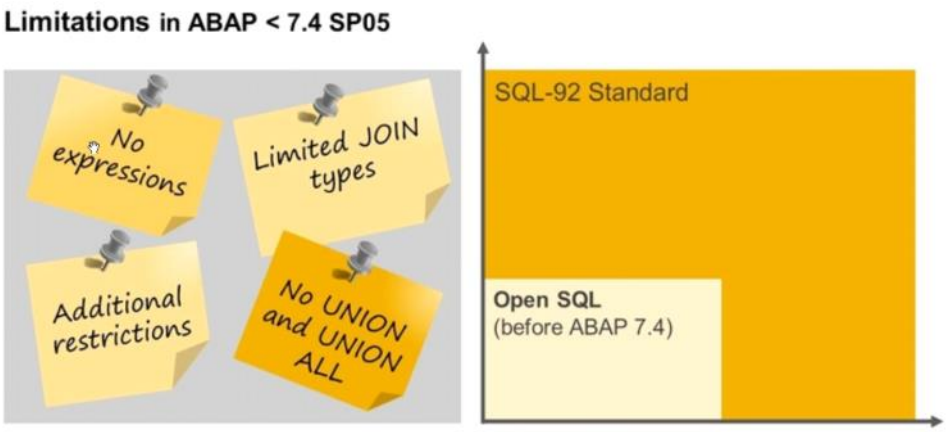
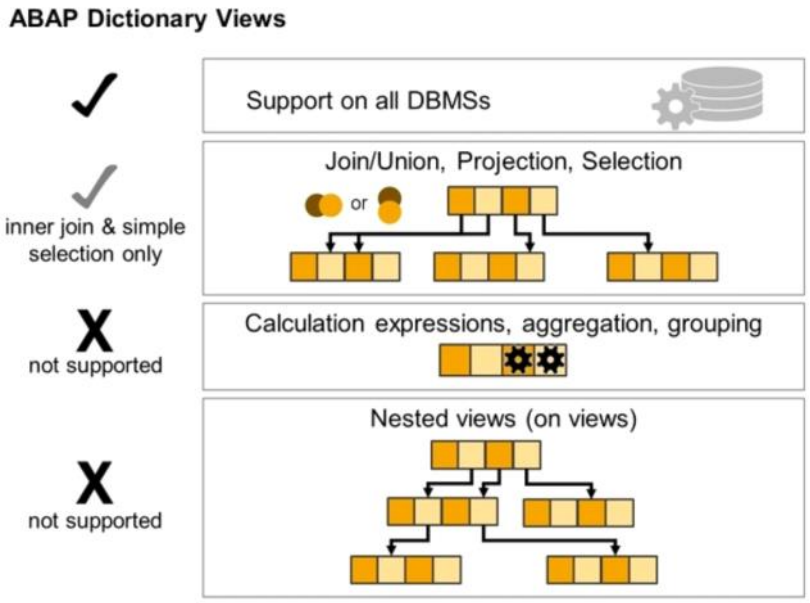
S10 - BW Modeling - CDS Views

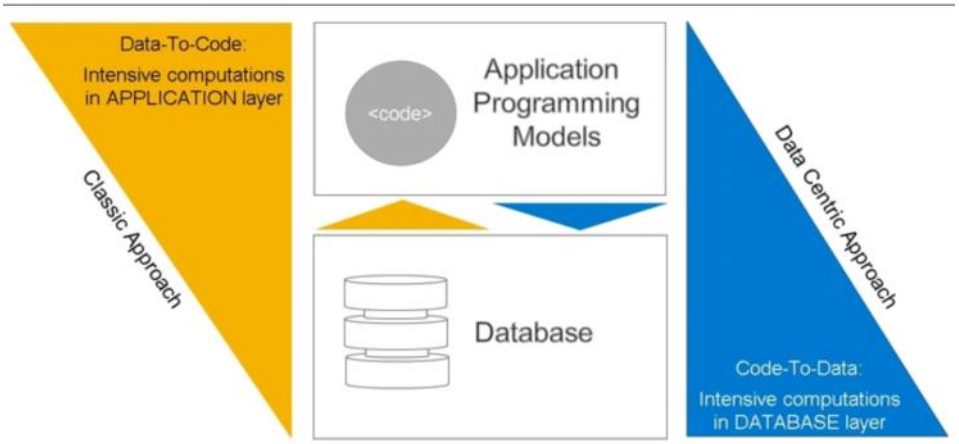
CDS Views	1. Datasource based on CDS View - Delta (Generic w/o ODQs) 2. Datasource based on CDS View - Delta (Generic with ODQs) 3. Enhancing datasource based on CDS views using extension CDS Views.	30th Aug: 8:30 AM - 10:30 AM
BW Statistics in BW4HANA	1. Concept of BW Statistics in BW4HANA. 2. CDS Views of type Query and Cube. 3. Datasource out of Operational Data providers for BW Statistics	30th Aug: 8:30 AM - 10:30 AM

Q & A - Pre Session

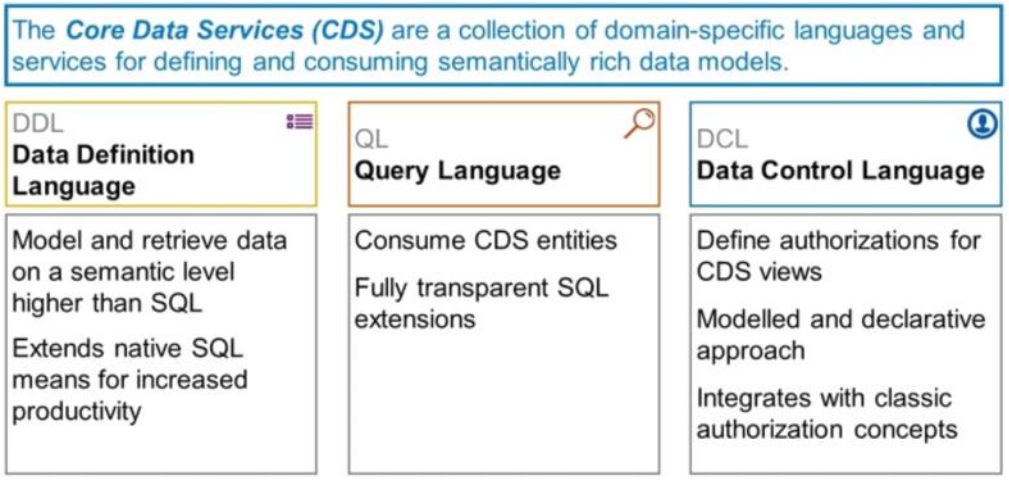
Shakthi	Created a basic standard ADSO with change log enabled with few fields as in the screenshot. When ran init, the initial 55 records from setup table got fetched. But when tried to run delta after making changes and running v3 job, the data is not getting picked up.  How ODQ_ENTITYCTR gets populated Datasource Propery

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:



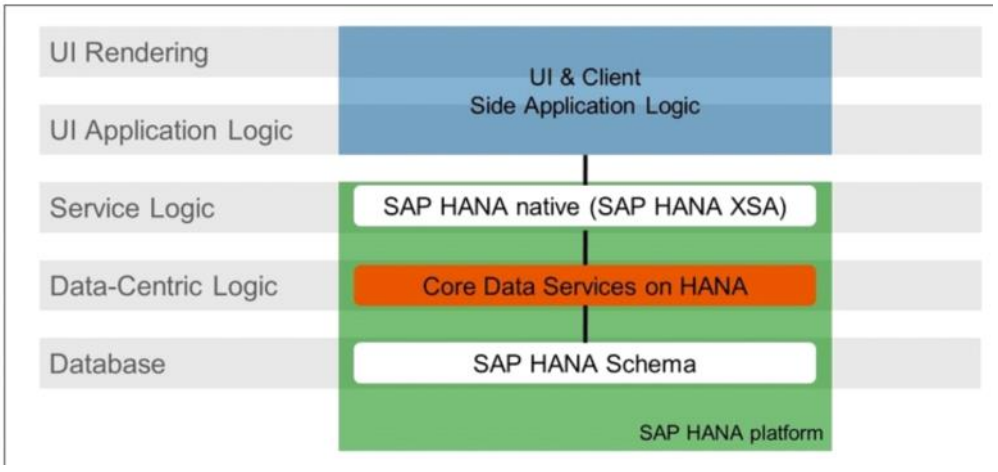
CDS



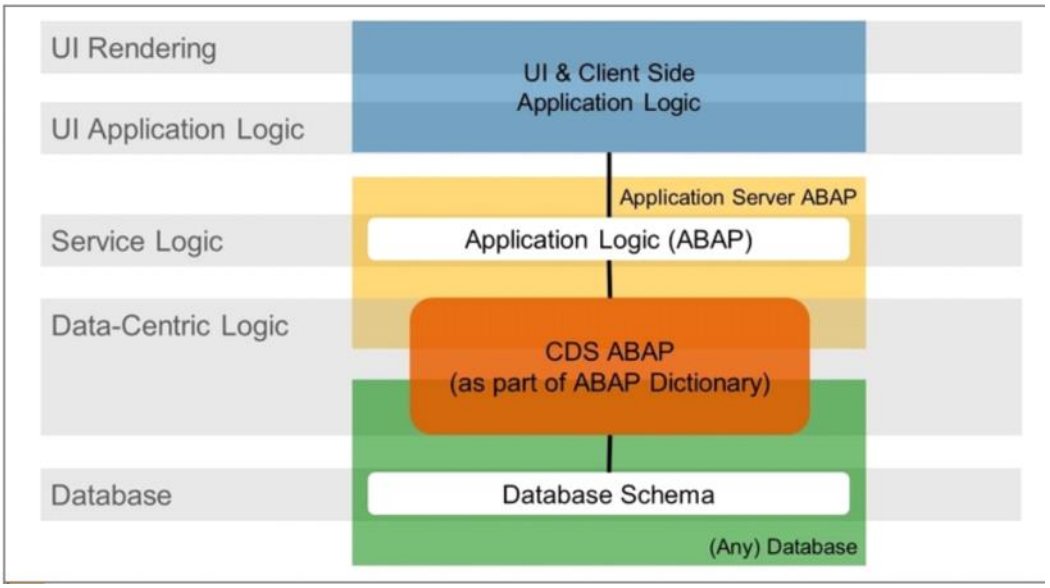
HANA CDS vs ABAP CDS

HANA CDS:

SAP HANA Core Data Services



ABAP CDS:

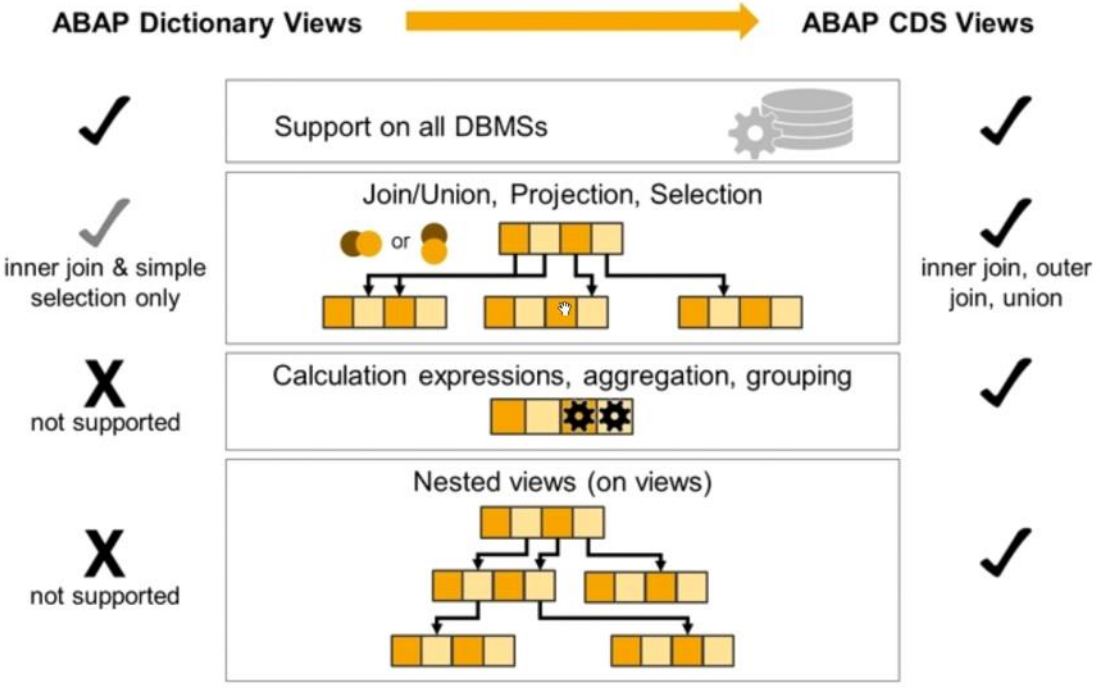
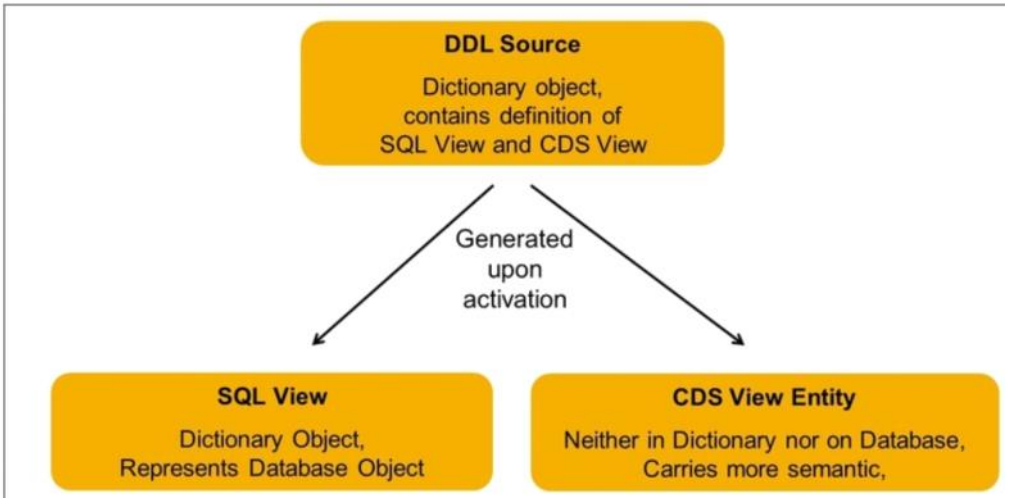


To take advantage of SAP HANA for application development, SAP introduced a new data modeling infrastructure known as core data services (CDS). This infrastructure is available in two flavors:

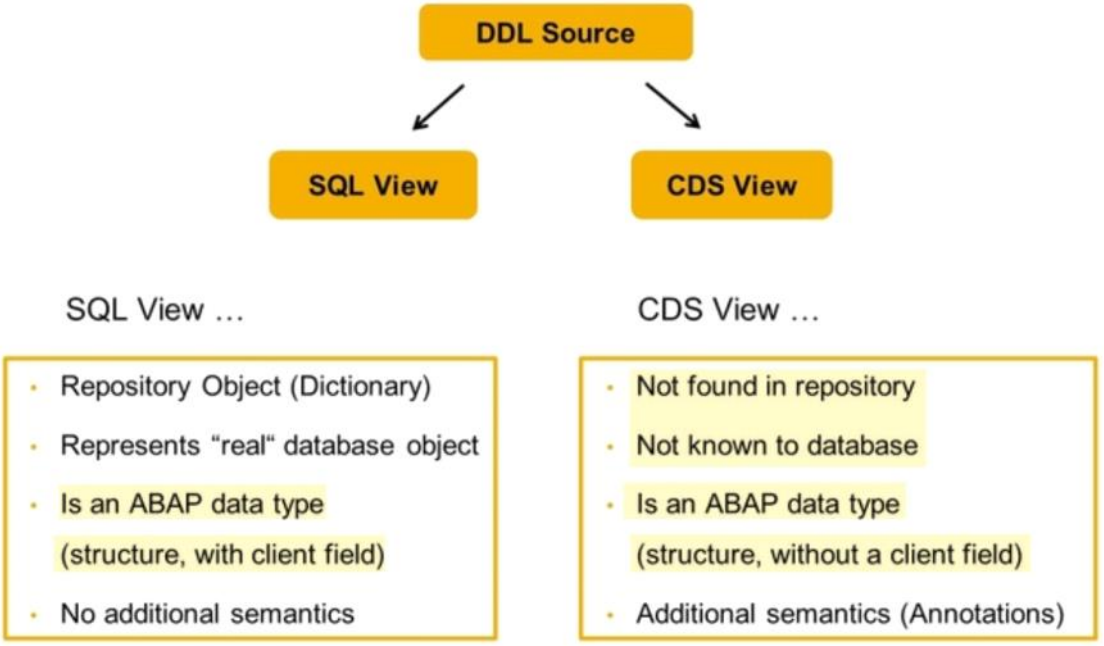
1. SAP HANA CDS provided by SAP HANA XS/XSA application server
2. SAP ABAP CDS provided by SAP NetWeaver ABAP in combination with SAP HANA

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.

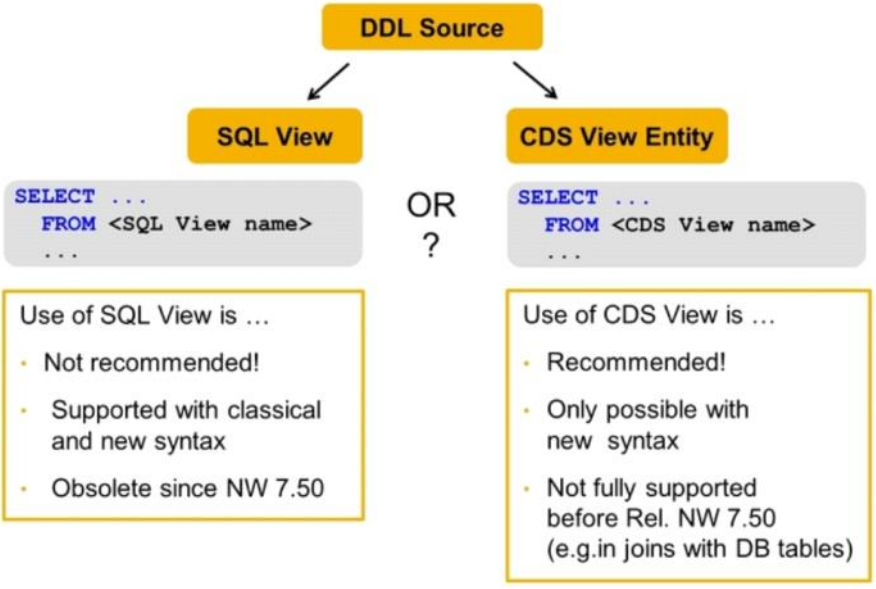
The 3 main musketeers of CDS	**Expressions** Used for calculations and queries in the data model **Associations** On a conceptual level, replacing joins with simple path expressions in queries **Annotations** To enrich the data models with additional (domain specific) metadata

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 

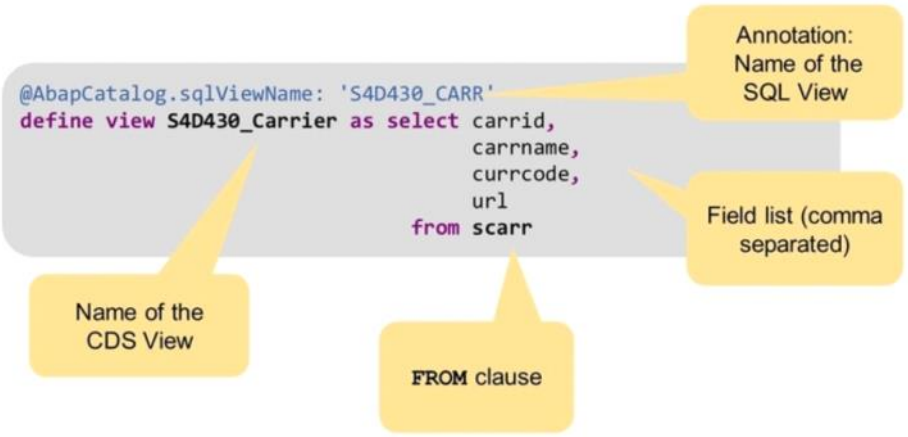
- A CDS View is defined in a DDL Source, which is a new type of repository object.
- There is no editor for DDL sources in the classical ABAP workbench. This new type of repository object has to be analyzed and developed in ABAP Development Tools in Eclipse.
- Upon activation of a DDL Source, two objects 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

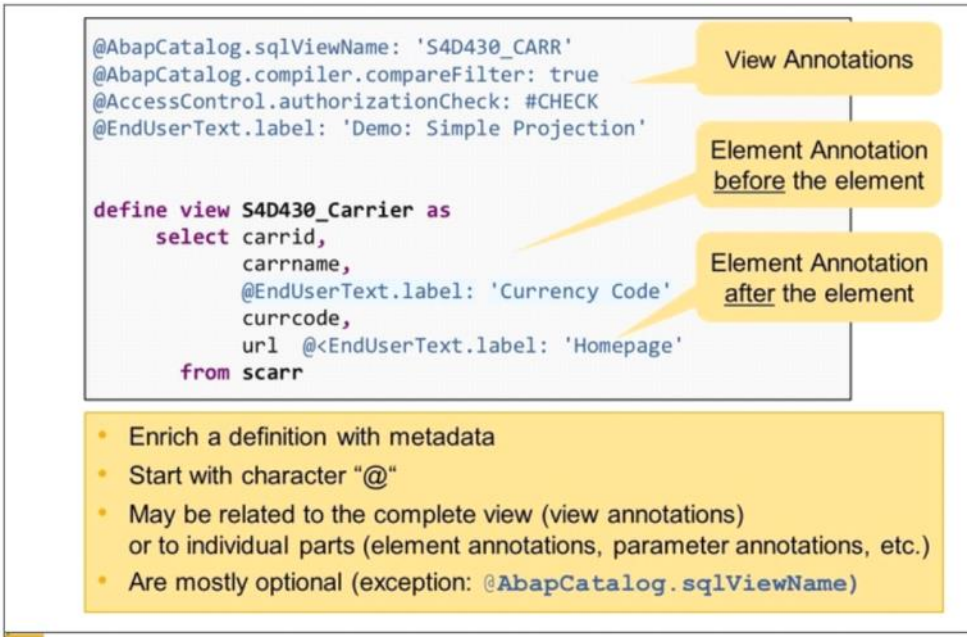


Syntax of CDS View



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 }
```

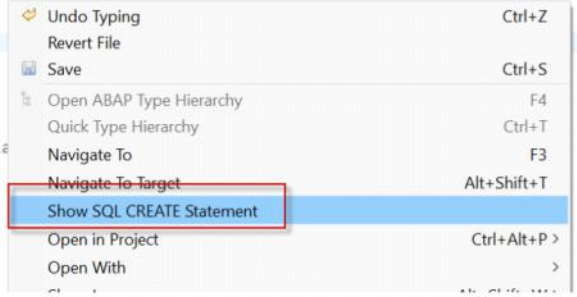
Similar to Join, we need key fields to Associate 2 different tables.

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

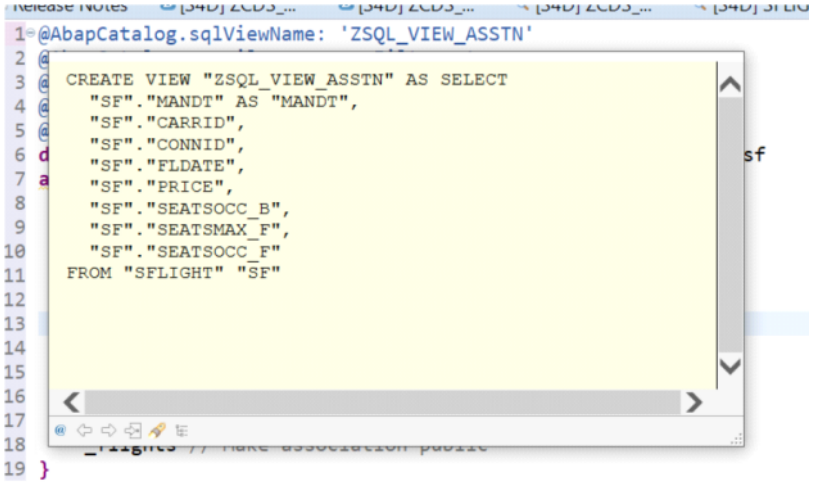
Make Association Public i.e. Expose the association. This will not create any Join beforehand but do it need basis.

Right click and select 'Show SQL CREATE Statement;

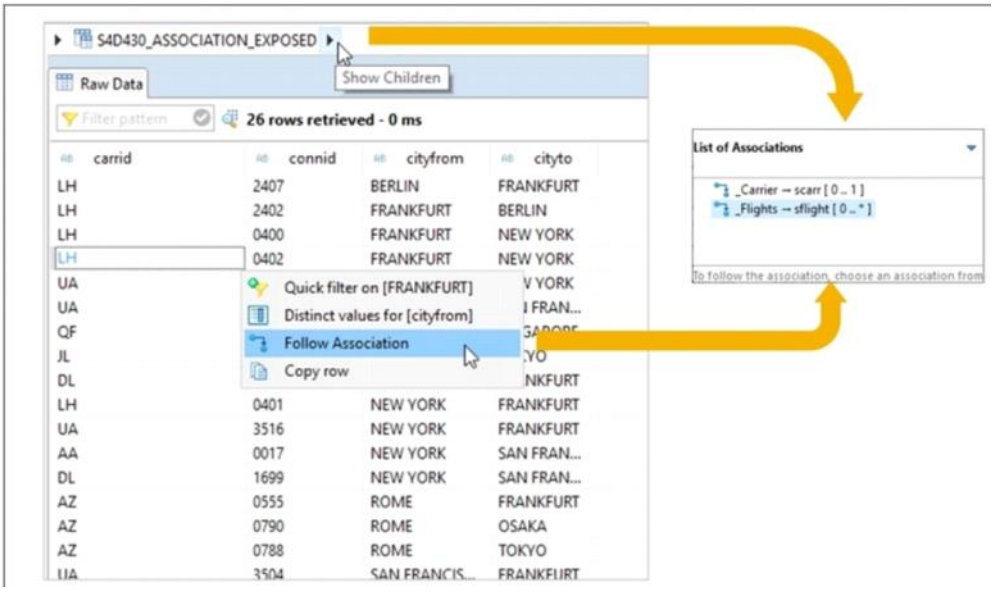
```
1 @AbapCatalog.sqlViewName: 'ZSQL_VIEW_ASSTN'
2 @AbapCatalog.compiler.compareFilter: true
3 @AbapCatalog.preserveKey: true
4 @AccessControl.authorizationCheck: #CHECK
5 @EndUserText.label: 'CDS View with Association concept'
6 define view ZCDS_VIEW_ASSOCIATIONS as select from sflight as sf
7 association [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"
)
```

Introduction

Statistical Analyses of the Data Warehouse Infrastructure

Provide feedback

There are various options available to you for statistical analyses of the data warehouse infrastructure. These queries are based on the core data services technology (CDS).

For the various areas, analytic queries based on query CDS views (CDS views with the analytics annotation `@Analytics.query: true`) are available. These serve as proposals for default analysis with the most important information, and can be executed in your BI client. You can also define queries based on TransientProviders derived from cube views (CDS views with analytics annotation `@Analytics.dataCategory: #CUBE`).

The use of CDS 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.

You can use CDS technology to analyze the following areas of the Data Warehouse:

- Data Loading
- Data Volume
- Query Runtime
- Process chains

SAP Help Link

https://help.sap.com/viewer/107a6e8a38b74ede94c833ca3b7b6f51/1.0.9/en-US/1e596b288f494f5d815c86cf94c3fbbb.html

BW Statistics in BW/4HANA

Introducing Statistical Analysis in SAP BW/4HANA

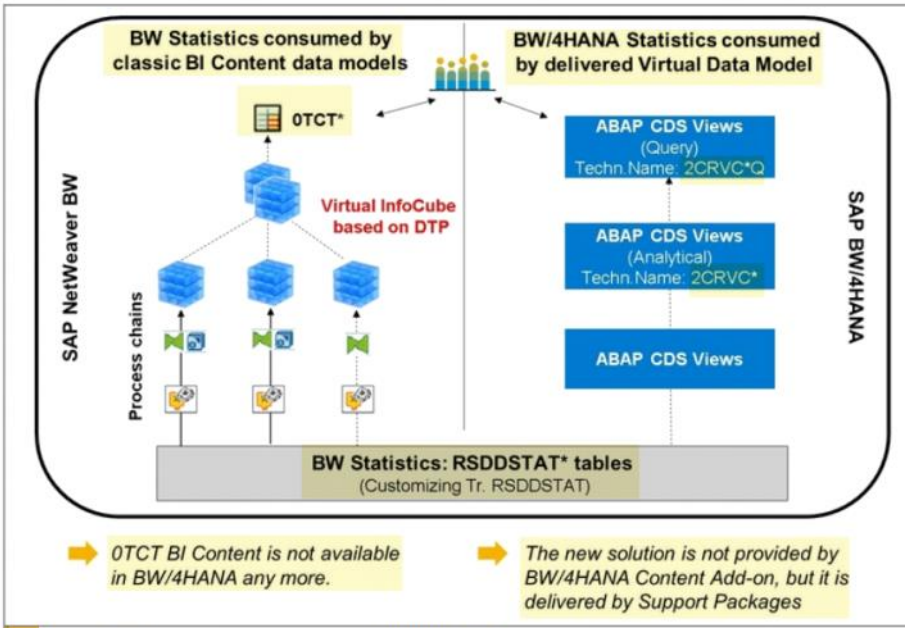


Figure 188: Reporting on BW Statistics with SAP BW/4HANA

CompositeProviders, SAP delivers a pure virtual data model (VDM) only. This VDM is based on **SAP ABAP CDS views** . Because it is a proven technology from SAP S/4HANA embedded analytics (see unit 9) the same concept is now leveraged for SAP BW/4HANA for state-of-the-art reporting of the still-existing BW statistics. The back-end tables and their administration (transaction `RSDDSTAT`) are still the same, but the whole reporting architecture has changed to a pure virtual one. This solution is rolled out in SAP BW/4HANA support packages, rather than the Content Add-On. Starting from SAP BW 7.5, you will find some first examples based on technical name 2CRVC* in a dedicated InfoArea called `SAP BW/4HANA (2O-BW4)` in the Query Monitor.

Query Monitor (Tr. RSRT)

Data Management (Runtime)	2O-BW4-DM
Runtime Statistics	2O-BW4-DM-STAT
HANA Object Data Volume Statistics (obsolete)	2CRVCTABSIZE
HANA Object Data Volume Statistics (obsolete)	2CRVCTABSIZQRY
Request Statistics (Data Target View)	2CRVCREQUEST
Request Statistics (Data Target View)	2CRVCREQQRY
Query Statistics (OLAP)	2CRVCOLAPSTATAC
Query Statistics (OLAP)	2CRVCOLAPSTATAQ
Cold Store Data Volume Statistics	2CRVCNLSTAT
Cold Store Data Volume Statistics	2CRVCNLSTATQ
Data Volume Statistics (current + history view)	2CRVCHNDBSTATALL
Data Volume Statistics (current + history view)	2CRVCHNDBSTAT...
Data Volume Statistics (current view)	2CRVCHNDBSTAT
Data Volume Statistics (current view)	2CRVCHNDBSTATQ
HANA Data Volume Statistics	2CRVCDVHDTABSIZE
HANA Data Volume Statistics	2CRVCDVHDTABSI...
Request Statistics (Process View)	2CRVCDTPLOAD
Request Statistics (Process View)	2CRVCDTPLOADQ
Logical Cube view for Cockpit Monitor	2CBW4_DVM_CU
Query for data volume statistics	2CBW4_DVM_Q1

Figure 189: Scope of Available Statistics in SAP BW/4HANA

InfoArea `SAP BW/4HANA (2O-BW4)` is not visible in the BW/4HANA Modeling Tools; however, it is still possible to define BW queries on this virtual data model. The CDS views generate **TransientProviders**, which serve as InfoProviders for BW/4HANA; hence, BW queries can be defined on them. When creating a new query, make sure you enhance the scope for TransientProviders so you are able to find them (technical name starts with `2CRV*`).

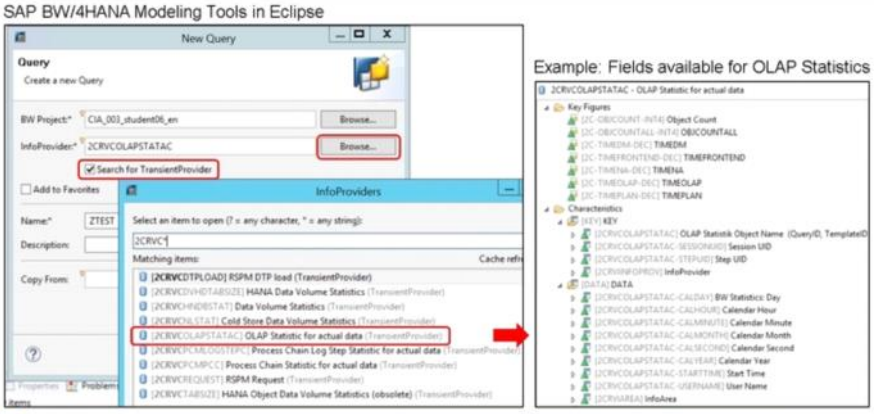


Figure 190: Creating Additional Queries on Top of the CDS Views Generated as Transient Providers for BW/4HANA

The following BW/4HANA statistics are currently available:

1. **Query Runtime Statistics**

   Query Runtime Statistics : The query returns information about the query runtime statistics. This displays the average query execution time for various layers, such as the front end, OLAP or data manager. Source tables: RSDDSTAT_OLAP (view), RSDDSTATHEADER, RSDDSTATINFO, RSDDSATEVDATA

   Aggregation of Query Runtime Statistics : Query runtime statistics in SAP BW / 4HANA are recorded by handle types and event IDs. This results in a large amount of data per navigation step, as information about every query runtime event is recorded. In order to simplify the analysis of this data, this detailed data is aggregated to the following key figures: 1. Time spent in the front end, 2. Time spent in the Analytical Engine (OLAP und Planning), 3. Time spent in the data manager.

2. Process Chain Statistics

Statistics for Process Chain Status : The query shows the current status of all process chains in the system with the same information as the process chain monitor or the monitoring of periodic process chains (transaction RSPCM). The statistics provide a snapshot of the current status information and no history. You can use the statistics for a customer-specific process chain monitoring, which is based on current information about the process chain status. Source tables: RSPCCHAINATTR, RSPCRECENTRUN

Statistics for Status and Runtime Information of Process Instances : With these statistics, status and runtime information are written for all process instances of all runs of all process chains. You can use the statistics for example to find out which processes in which chains typically take the most time during runs, or which often terminate. You can also use these to perform analyses over a period of time, thus gaining insights which you could not gain from the process chain status statistics, which only offer a snapshot of the process chain. Source tables: RSPC*

3. Data Loading Statistics

RSPM Request Statistics : The query returns information on requests for a BW target object (if the requests were successful, for example). Advanced DataStore objects, InfoObjects, and open hub destinations are all supported as target objects. Source table: RSPMREQUEST

RSPM DTP Load Statistics : The query returns information for requests that are executed during DTP execution. Advanced DataStore objects, InfoObjects, and open hub destinations are all supported as target objects. Source tables: RSPMREQUEST, RSPMXREF, RSPMPROCESS, RSBKDTP

4. Data Volume Statistics

Statistics for the combined SAP HANA/cold store data volume : The statistics for the combined data volume provide you with an entry point for analyzing the data volume and provide a view of the entire volume of data in the system at the current point in time. The query returns information about the data volume, which was moved by advanced DataStore objects into the cold store and stored there, together with data volumes from the SAP HANA database of SAP BW/4HANA. The statistical information for advanced DataStore objects includes: Table size in SAP HANA and the cold store, raw size in SAP HANA and the cold store, number of records in SAP HANA and in the cold store, and archiving rate (percentage share of archived data). The statistics for SAP HANA online data volume and the statistics for cold store data volume provide you with more detailed information.

Statistics for SAP HANA online data volume : The query provides information about the data volume of database tables in the SAP HANA database of the SAP BW/4HANA system. Source SAP HANA views: M_CS_TABLES, M_RS_TABLES

Statistics for cold Store data volume : The query returns information about the data volume, which was moved by advanced DataStore objects into the cold store and saved there. The statistical information includes: Table size in the cold store, raw size of the transferred data, number of records, and compression rate. Source View: RSDANLSTAT_VQuery

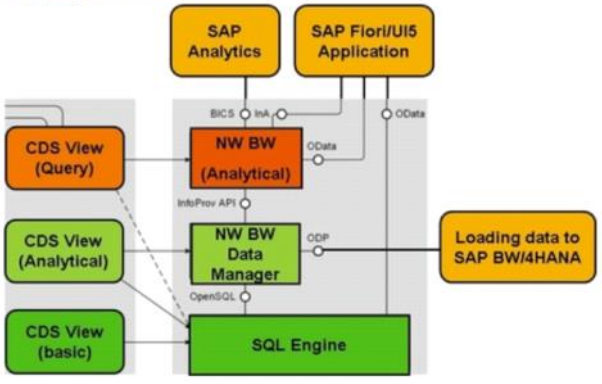
CDS View and BW Analytic Manager

SAP BW/4HANA Statistics is delivered based on ABAP CDS views

The CDS views to be consumed are defined with annotation "@Analytics.query: true"

Based on this property an ABAP CDS view becomes an Analytic Query (Default Query Name = 2C<SQL-view name>)

- View is interpreted as transient BW query and can be executed by the BW Analytic Engine.
- View can be consumed by SAP Analytics clients like Lumira, Analysis Office, or Analytics Cloud.
- View can leverage built-in functions of the BW Analytic Engine (e.g., exception aggregation).
- View can leverage hierarchies for display or as filter.



For the various areas in the SAP BW/4HANA statistics, queries based on query CDS views (CDS views with the analytics annotation `@Analytics.query: true`) are available. default technical query name follows the pattern 2CRV*Q. These serve as proposals for default analysis with the most important information of the statistics available in SAP BW/4HANA, and can be executed in all SAP Analytics BI clients.

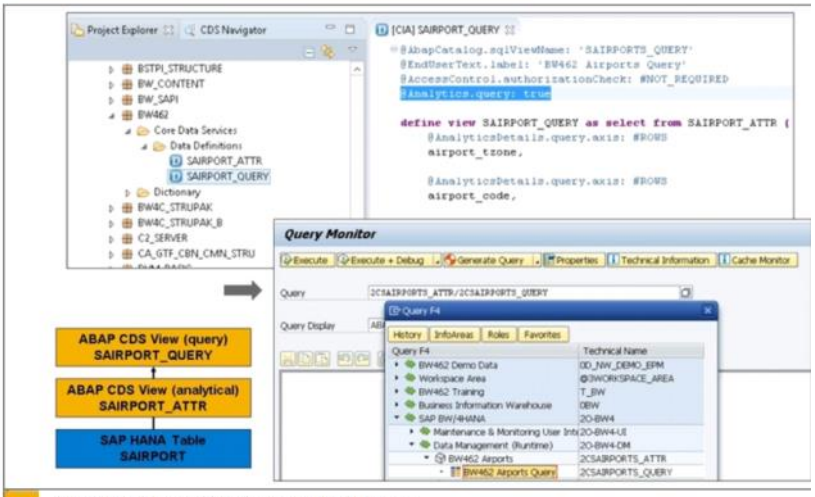


Figure 193: Simple Example of an Analytical Query

You can also define new BW queries based on the TransientProviders derived from the analytical CDS views (CDS views with analytics annotation `@Analytics.dataCategory: #CUBE`).

So, there are 2 approaches:

1. Use SAP-given Stnd queries (Naming convention: 2CRV*Q) -
   These are based on query CDS views (Analytics.query: true). These are referred to as 'Analytic Query'.
   In BW/4HANA, these are treated as 'Transient BW Query' which can be consumed by AO, Lumira or Analytics Cloud.

2. Create your own queries - These are based on transientProviders, which are generated out of Analytical CDS views (Analytics.dataCategory: #CUBE)

Example

ABAP CDS View for Transient Provider (known as analytical CDS View) is different and
ABAP CDS View for BW Query (known as query CDS View) for Statistics is different.

Cube CDS View Generates:

1. Operational Data Provider (Datasource creation)
2. Transient provider (RSRT , InfoArea: 2O-BW4) - Used to create user specific queries

Query CDS View Generates:

1. Analytic Query (Used for as it is reporting for BW Statistics)
2. Built upon the Cube CDS View.

Two important analytic annotations:

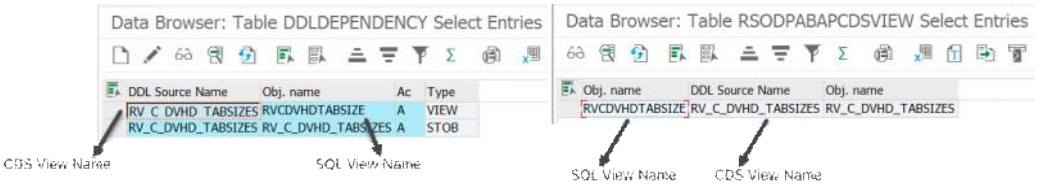
- CDS views with the annotation @Analytics.query: true are transient queries which can be interpreted by the Analytic Engine.
- CDS views with the analytic annotation @Analytics.dataCategory:#CUBE are transient BW providers.

TRANSIENT PROVIDER	Source ABAP CDS VIEW for TransientProvider	SQL VIEW for ABAP CDS VIEW	BW QUERY	Source ABAP CDS VIEW for BW Query	SQL VIEW for ABAP CDS VIEW for TransientProvider
2CRVCOLAPSTATAC	RV_C_OLAPSTATACUBE	RVCOLAPSTATAC	2CRVCOLAPSTATAQ	RV_C_OLAPSTATAQUERY	RVCOLAPSTATAQ
2CRVCDVHDTABSIZE	RV_C_DVHD_TABSIZES	RVCDVHDTABSIZE	2CRVCDVHDTABSIZEQ	RV_C_DVHD_TABSIZESQUERY	RVCDVHDTABSIZEQ

Query CDS View builds upon, meaning does 'SELECT FROM', analytical CDS view.

TransientProvider naming convention: 2C<SQL view name of analytical ABAP CDS View'
BW Analytic Query naming convention: 2C<SQL view name of query ABAP CDS View'

The connection between DDLSOURCENAME and SQLVIEWNAME can also be found in table RSODPABAPCDSVIEW or table DDLDEPENDENCY.



Wiki Link for CDS View based reporting	https://wiki.scn.sap.com/wiki/display/BI/CDS+views%3A+HowTo+use+in+BW+contexts

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
	}