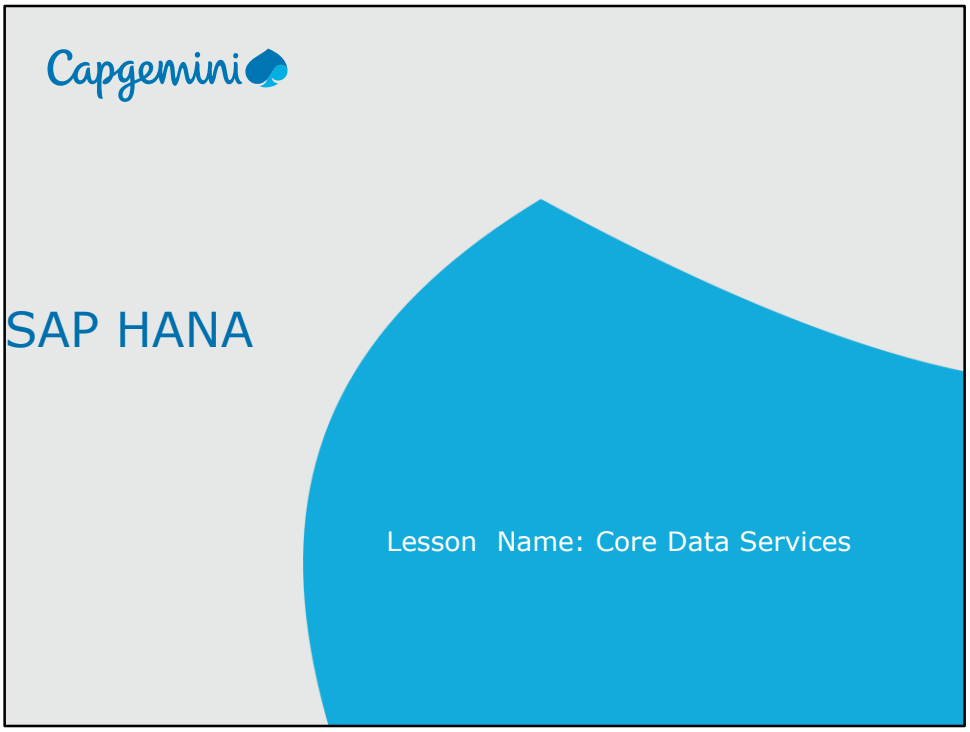
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.

Lesson Objectives

After completing this lesson, participants will be able to –

- Basics of Core Data Services (CDS)
- Demo on CDS
- CDS view Definition Features

Instructor Notes:

Contents

Introduction to CDS
CDS in ABAP
Demo on CDS
CDS View Definition Features

Instructor Notes:

Introduction to Core Data Services (CDS) View

CDS stands for Core Data Services.

A view is an entity that is not persistent; it is defined as the projection of other entities.

CDS View is reusable data models on the database.

It is a **data** model that represents framework of what relationships are in a database.

To take advantage of SAP HANA for application development, SAP introduced a new data modeling infrastructure known as core data services.

With CDS, data models are defined and consumed on the database rather than on the application server.

The rule-of-thumb is simple:

Do as much as you can in the database to get the best performance.

Instructor Notes:

Introduction to Core Data Services (CDS) View

A **CDS view** is defined for existing database tables and any other **views** or **CDS views** in ABAP Dictionary .

CDS is a data modeling infrastructure for defining and consuming semantic and reusable data models on the database, rather than on the ABAP server, regardless of the database system used

Technically, it is an enhancement of SQL which provides you with a data definition language (DDL) for defining semantically rich database tables/views (CDS entities) and user-defined types in the database.

CDS entities and their metadata are extensible into the ABAP Data Dictionary and the ABAP language.

 5

Instructor Notes:

Features of Core Data Services (CDS) View

Core Data Services

- Next generation of data definition and access for database-centric applications
- Optimized application programming model for all domains (transactional, analytical,…)
- Technically an extension to SQL:
  - Expressions
  - Domain-specific metadata
  - Associations
- CDS includes
  - Data Definition Language (**DDL**)
  - Query language (**QL**)
  - Data Manipulation Language (**DML**)
  - Data control language (**DCL**)

User Interface

UI Abstractions

Application Programming

Core Data Services

Database

Instructor Notes:

Features of Core Data Services (CDS) View

Code-to-Data paradigm

- Supported through extended view functionality

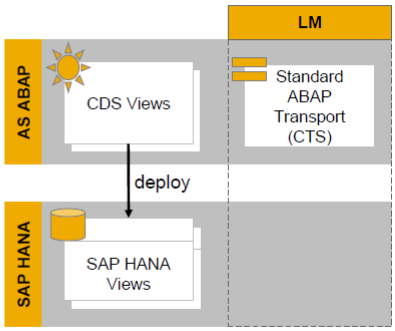
Definition of semantically rich data models in the ABAP Dictionary

- ABAP 'view entities' in DDL source objects (`R3TR DDLS`)

Fully integrated into the ABAP infrastructure

- Consistent lifecycle management with all other ABAP artifacts

Consumption via Open SQL on view entities



Instructor Notes:

Advantages of CDS

Semantically rich data models

CDS builds on the well-known entity relationship model and is declarative in nature, very close to conceptual thinking.

Compatibility across any database platform

CDS is generated into managed Open SQL views and is natively integrated into the SAP HANA layer.

Efficiency

CDS offers a variety of highly efficient built-in functions — such as SQL operators, aggregations, and expressions — for creating views.

Extensibility

Customers can extend SAP-defined CDS views with fields that can be automatically added to the CDS view

Instructor Notes:

Definition of CDS VIEW

The statement **DEFINE VIEW** is used to create the CDS DDL in ABAP.

This is done in the CDS source code of a CDS data definition in the ABAP Development Tools (ADT)

Definition is only possible with ABAP Development Tools in Eclipse/HANA Studio .

CDS view cannot be created via transaction SE11.

CDS views can be developed and maintained in SAP HANA studio and in ABAP in Eclipse

Instructor Notes:

CDS View Definition

Definition of the CDS consists of

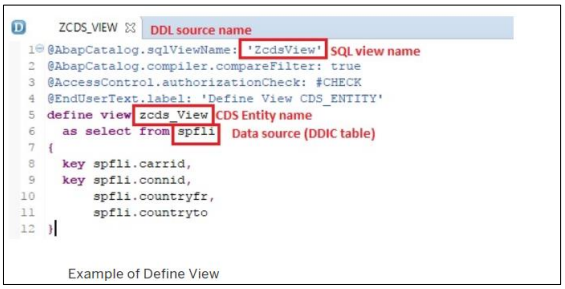
- View name
- Semantic information (key field)
- Projection List
- Aliases

Projection List :

- Client Dependency
- Semantic Information (Key)
- Aliases
- Aggregation
- Literals
- Arithmetic Expressions
- Conditional Expressions

Instructor Notes:

Simple CDS View



Example of Define View

Instructor Notes:

CDS View with Join

ZCDS_VIEW

```
@AbapCatalog.sqlViewName: 'ZcdsView'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Define View CDS_ENTITY'

define view zcds View
  as select from spfli
    join         scarr on spfli.carrid = scarr.carrid
{
  key spfli.carrid,
  key scarr.carrname,
  key spfli.connid,
      spfli.countryfr,
      spfli.countryto
}
```

Example of Define View with Join

Instructor Notes:

Demo

Create Simple CDS View, Preview it and consume it via Open SQL

Instructor Notes:

CDS View Definition

Literal values:
- C-sequence literals (Max length: 1333)
- Signed integer literals (4-Byte)

Aggregation functions:
- MIN, MAX, COUNT, AVG, SUM
- Alias required for function results

String functions:
- LPAD,SCORE,LEFT,LTRIM,SUBSTRING
- Alias required for function results

Instructor Notes:

Type of CDS Views

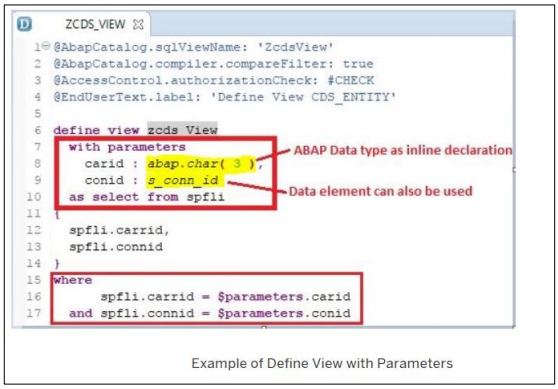
CDS View with Input Parameters
CDS View Extensions
View-on-View
CDS View without Input Parameters

Instructor Notes:

Type of CDS Views

CDS View with input parameters

- To create Views with parameters key word used is **With PARAMETER.**
- Comma-separated list of scalar input parameters and their corresponding type is to be mentioned.
- Supported parameter types are as follows:
  - Predefined data type like abap.char(char_len)
  - Name of a data element



Example of Define View with Parameters

Instructor Notes:

Demo

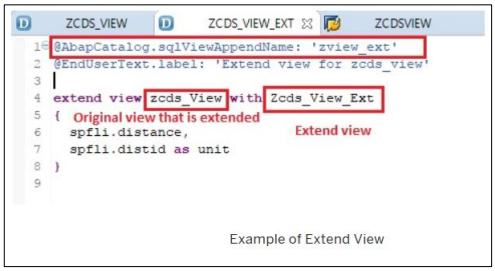
CDS View with Input Parameter

Instructor Notes:

Type of CDS Views

CDS View Extensions

- A CDS view can be extended by adding new fields .
- To extend a CDS Base view, the key word used is **EXTEND VIEW .**
- In the extended view, mention the new fields to be added separated by comma.



Example of Extend View

Instructor Notes:

Demo

Create Base CDS View

Extend the above CDS View

Instructor Notes:

Type of CDS Views

View-on-View

- Another important type of CDS view is View on View.
- You can create CDS view on another CDS View(called as the Base View).
- There is no restriction on the number of layers.

```
@AbapCatalog.sqlViewName: 'ZDDLS_CDS_13A'
define view zcdsv_base as select
from snwd_so as so
{
 key so.so_id as order_id,
 so.buyer_guid,
 so.currency_code,
 so.gross_amount
}
```

```
@AbapCatalog.sqlViewName: 'ZDDLS_CDS_13B'
define view zcdsv_view_on_view as select
from zcdsv_base
inner join snwd_bpa as bpa
  on bpa.node_key = zcdsv_base.buyer_guid
{
 key bpa.bp_id,
 bpa.company_name,
 zcdsv_base.currency_code,
 zcdsv_base.gross_amount
}
```

Instructor Notes:

Demo

Create Base CDS View

Create View on the above Base View

Instructor Notes:

Consumption of CDS View

Consumption of CDS can be done in the following ways:

- In a CDS View
- By Open SQL
- Data Preview (context menu in ADT)
- SAP List Viewer
- SAP NetWeaver Gateway (OData Model)

Instructor Notes:

Consumption of CDS View

Consumption of CDS View in CDS View:

- You can create CDS view on another CDS View(called as the Base View)
- View on View is nothing but the consumption of CDS View in another View

```
@AbapCatalog.sqlViewName: 'ZDDLS_CDS_14B'
define view zcdsv_consume_param_view as select from
zcdsv with input parameters( customer_name : 'SAP' ) as vwp
{
  vwp.param_customer_name
}
```

```
@AbapCatalog.sqlViewName: 'ZDDLS_CDS_14A'
define view zcdsv with input parameters
  with parameters customer_name : abap.char(80)
as select
from snwd_so as so
join snwd_bpa as bpa
  on bpa.node_key = so.buyer_guid
{
  key so.so_id as order_id,
  $parameters.customer_name as param_customer_name,

  case
    when bpa.company_name = $parameters.customer_name
     then 'Found it!'
    else  'Not found'
  end as found_customer
}
where bpa.company_name = $parameters.customer_name
```

Instructor Notes:

Consumption of CDS View

CDS View is consumed via OpenSQL using below 4 steps

- Check if the feature is supported :**abap_true**
- Provide (mandatory) input parameter(s) :**Customer_name**
- Suppress syntax warning using the pragma **##**
- Provide a "fallback" implementation / some error handling : **ELSE**

```
REPORT zr_cds_01_consumption_vwp.

DATA lv_cust_name TYPE c LENGTH 80 VALUE 'SAP'.

"awesome application logic

DATA(lv_feature_supported) =
 cl_abap_dbfeatures=>use_features(
  EXPORTING
   requested_features  =
    VALUE #( ( cl_abap_dbfeatures->views_with_parameters ) )
 ).

IF lv_feature_supported = abap_true.
  SELECT *
  FROM zcdsv_with_input_parameters( customer_name = 'SAP' )
  INTO TABLE @DATA(lt_result)
  ##DB_FEATURE_MODE[VIEWS_WITH_PARAMETERS].
ELSE.
  "do some alternative coding here
ENDIF.

"even more awesome application logic
cl_demo_output=>display_data( lt_result ).
```

Instructor Notes:

Add instructor notes here.

Summary

In this lesson, you have learnt:

- Basic Concepts of Open SQL
- Features of Open SQL
- Open SQL Syntaxes and Statements
- Performance Rules and Limitations of Open SQL
- About Core Data Services
- CDS in ABAP
- Demos on CDS
- CDS View Definition Features

Add the notes here.

Instructor Notes:

Review Questions

OPEN SQL Statements are those statements which are used to -------- or --------- database table data.

For OPEN SQL statements insertion in database table is possible in -------- way/ways.

Open SQL in ABAP application server is the ------- ---------layer calling an SQL like syntax.