



# NoR GmbH

~ for example stories ~

## Core Data Services (CDS)

Dr. Roland Schwaiger  
© 2025 NoR GmbH

# Agenda

- System
- Part 2
  - CDS - Basics
    - VDM
    - CDS - Views
    - Data Preview
    - Association Navigation
    - SQL Console
    - Annotations
    - Dependencies
  - Creating CDS View
  - Inner Join, Projection, Selection
  - Annotations
  - Identifiers, translatable texts, methods
  - SQL Expressions
  - SQL Functions
  - Nested Views
  - Aggregations
  - Input parameter
  - Associations
  - ABAP Testing Program for CDS Usage
  - Access control
  - Table Functions (AMDP)



# Part 2

# CDS Data

# CDS - Core Data Services

CDS is a collection of domain-specific languages and services for defining and consuming semantically rich data models

DDL = Data Definition Language

- Modeling and Retrieving Data
- Extension of native SQL

DQL = Data Query Language

- Use of CDS entities
- SQL Extension

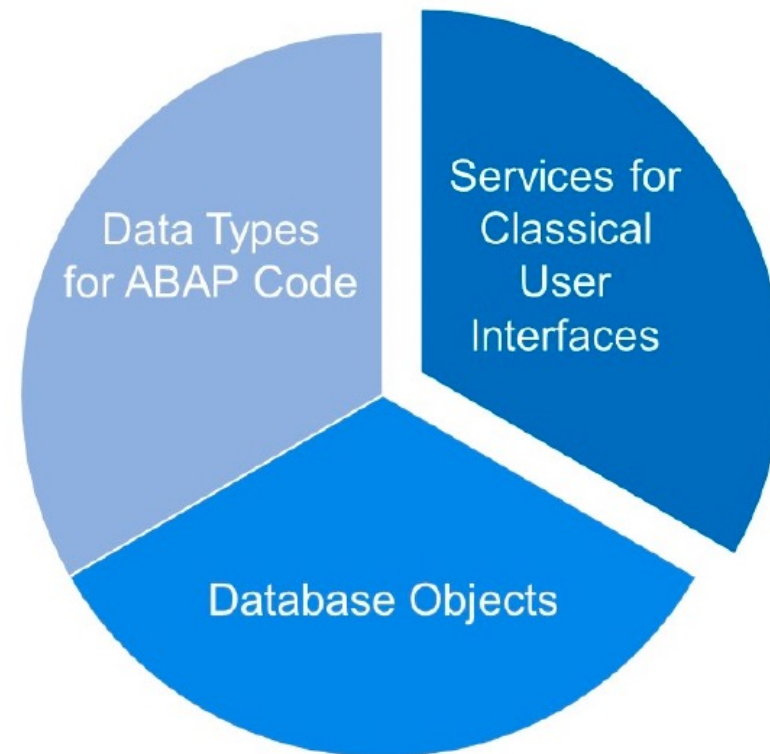
DCL = Data Control Language

- Defining Permissions for CDS Views
- Integrated into the classic SAP authorization concept

# ABAP Dictionary

## Tasks of the ABAP Dictionary

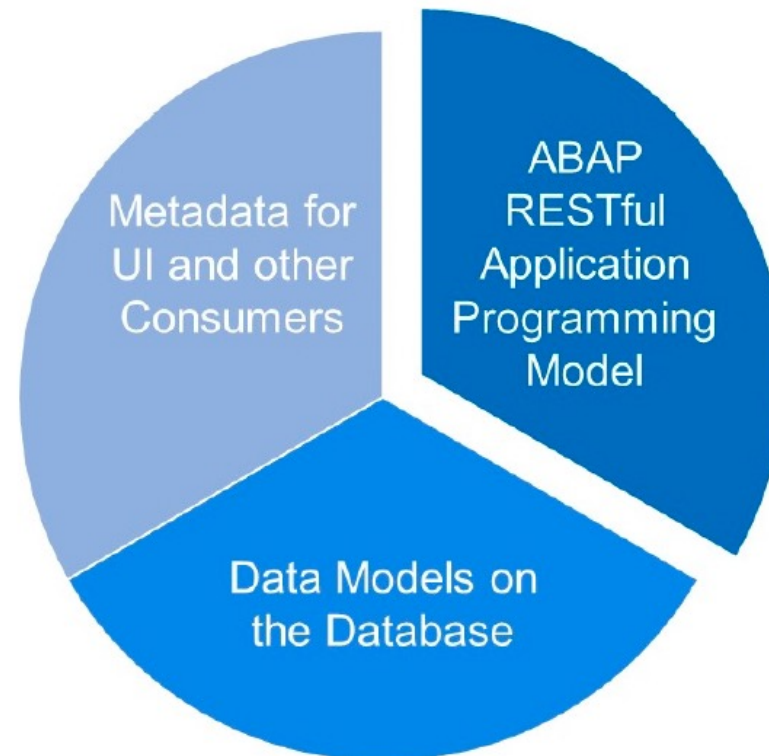
- **Defining Database Objects**
  - Database tables and views
  - Independent from actual database system
  - For use in ABAP SQL
- **Providing Data Types for ABAP Code**
  - Data elements, structures, table types
  - For use in data declarations and definition of interface parameters
  - globally visible in the system
- **Providing Services for Classical UIs**
  - Labels, documentation, value helps, and generated value checks, ...



# ABAP CDS

## Tasks of the ABAP Core Data Services

- **Defining Data Models on the Database**
  - CDS objects (views and functions)
  - Including SQL logic for code-to-data
  - For use in ABAP SQL
- **Providing Metadata**
  - Semantically enriched data models
  - Support modern UI technologies (SAP Fiori, OData, and so on)
- **Support the ABAP RESTful Application Programming Model**
  - Special CDS objects
  - define the behavior of business objects



# CDS Objects

Development Object	Purpose	Remark
CDS Data Definition	Define a CDS Entity	CDS Entities are: <ul style="list-style-type: none"> <li>• CDS View Entity</li> <li>• CDS DDIC-based Views (obsolete)</li> <li>• CDS Projection Views</li> <li>• CDS Abstract Entities</li> <li>• CDS Table Functions</li> <li>• CDS Custom Entities</li> <li>• CDS Hierarchies</li> </ul>
CDS Access Control	Define a CDS Role	Control read access to data
CDS Metadata Extension	Add Metadata to a CDS Entity	Store metadata for a CDS entity outside its data definition
CDS Behavior Definition	Define the Behavior for a Business Object	Used for transactional processing in the ABAP RESTful Application Programming

## Quick Reference

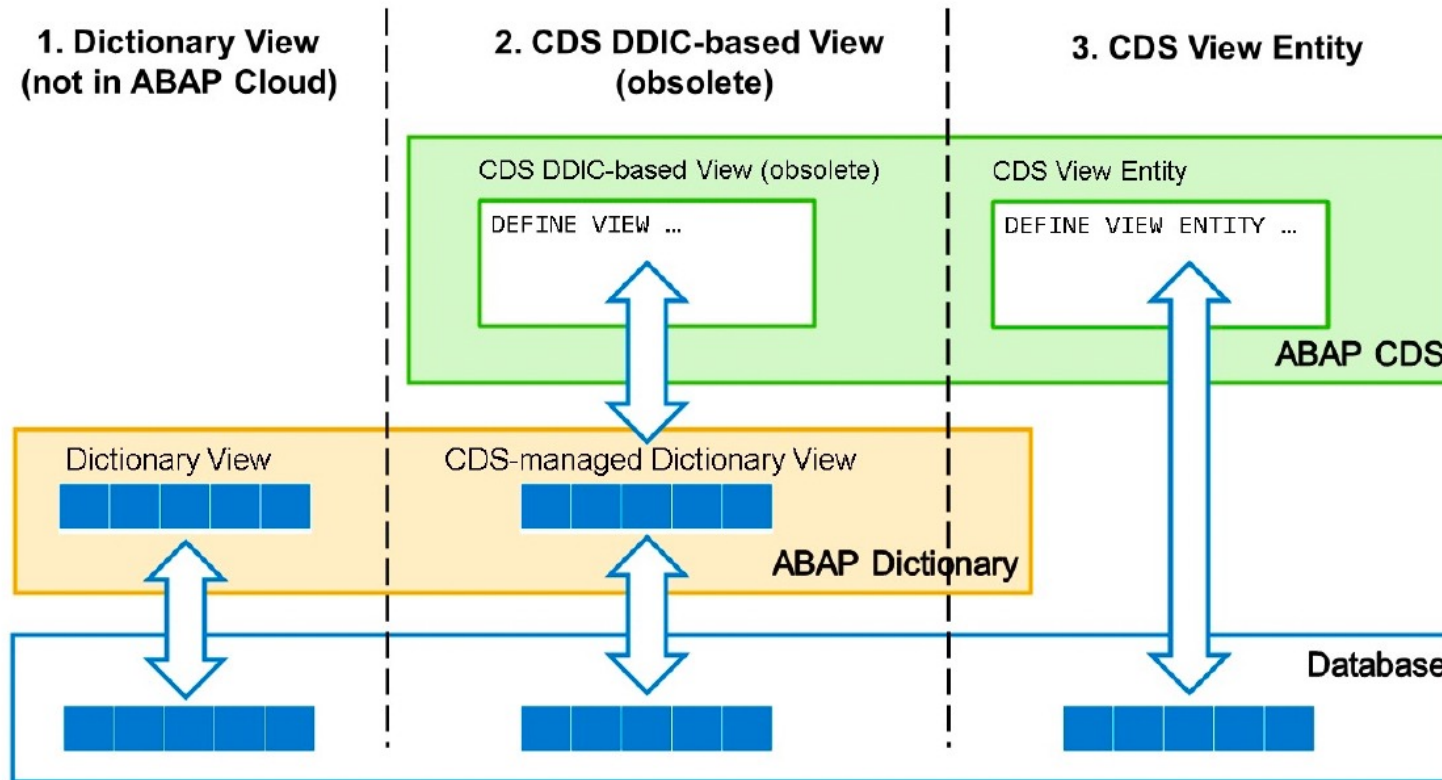
<https://community.sap.com/t5/technology-blog-posts-by-members/cds-views-7-5x-quick-reference/ba-p/13978469>

Name	Explanation
CDS DDIC-Based View	SQL access to data sources, technically implemented as a DB view. Introduced in ABAP 7.40 SP5. DEFINE VIEW ...
CDS Table Function	CDS entity implemented as an AMDP table function. Introduced in ABAP 7.50. DEFINE TABLE FUNCTION ...
CDS Abstract Entities	Defines an abstract CDS entity without a data source. Introduced in ABAP 7.53. DEFINE ABSTRACT ENTITY ...
CDS Hierarchies	CDS entity that realizes an SQL hierarchy on a data source. Introduced in ABAP 7.53. DEFINE HIERARCHY ...
CDS Custom Entities	Implemented as an ABAP class. Supports non-persistent data for UI scenarios. Introduced in ABAP 7.54. DEFINE CUSTOM ENTITY ...
CDS Projection Views	Access to a subset of a CDS view. Useful for service exposure with limited fields. Introduced in ABAP 7.54. DEFINE VIEW ENTITY ... AS PROJECTION
CDS View Entities	SQL access to data sources using the new CDS syntax. Modern replacement for DDIC-based views. Introduced in ABAP 7.55. DEFINE VIEW ENTITY ...

# CDS Subobjects in the Repository

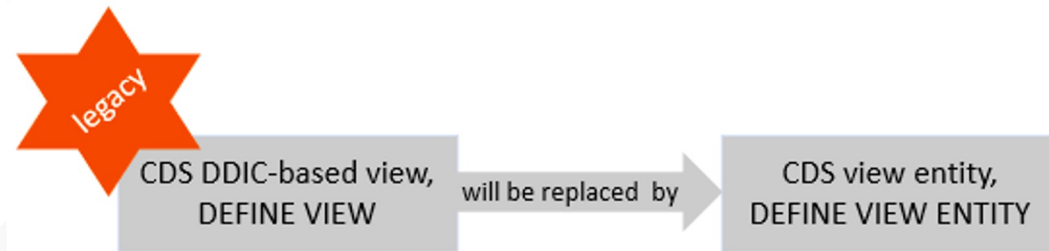
- Access Control (DCL)
  - Must be maintained in Eclipse ADT
- Data Definition (DDL)
  - Must be maintained in Eclipse ADT
  - Defines CDS View or CDS Table Function
- Data Query Language (DQL)
  - Similar to Standard SQL-92
- Metadata Extension (@ Annotation)
  - Must be maintained in Eclipse ADT
  - Annotated CDS View or CDS Table Function

# CDS Views



# SAP CDS Innovations

## CDS view entity



With ABAP release 7.55, a new type of CDS view is available: in official terminology, it's called CDS view entity. And it has come to replace the "classic" CDS DDIC-based views that have been around for years.

- No additional ABAP Dictionary view is created on activation.
- Improved performance during view activation.
- Optimization and simplification of syntax.
- Stricter syntax checks indicate problematic situations more explicitly, for example, annotation checks.

# Naming conventions SAP

Name of the DDL Source:

- Max. 30 characters
- System-wide unique (Customer namespace!)
- Always uppercase

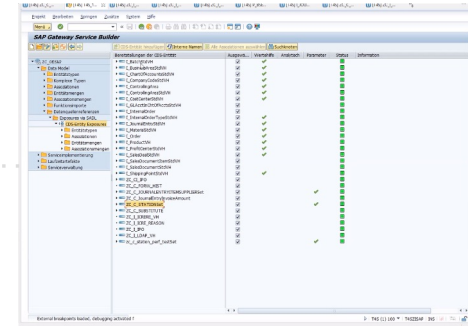
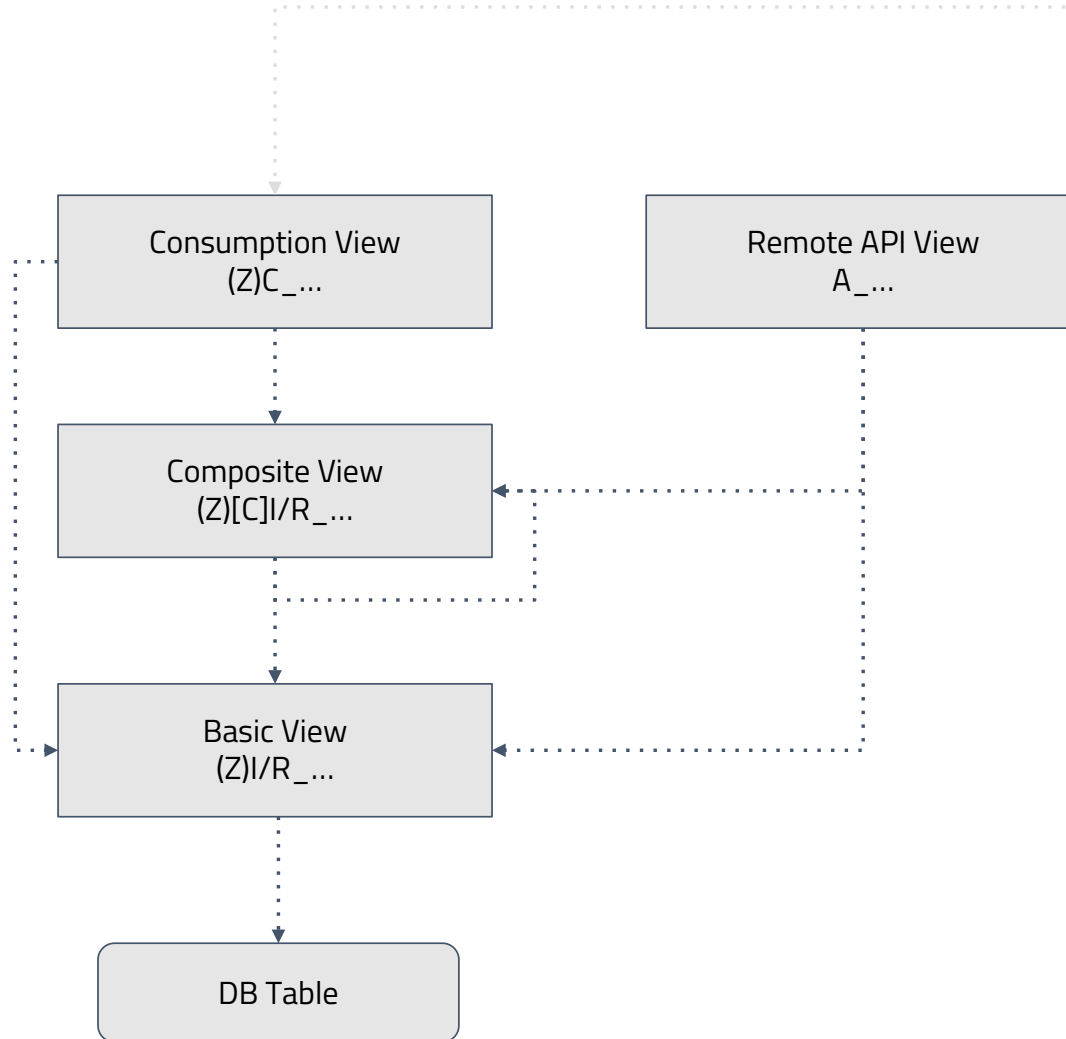
Name of the CDS View:

- Max. 30 characters
- System-wide unique (Customer namespace!)
- Unique in the ABAP Dictionary (global ABAP data type)
- Case-insensitive (upper/lowercase is not distinguished)
- Can differ from the name of the DDL source (not recommended!)

Name of the SQL View:

- Max. 16 characters (standard Dictionary object)
- System-wide unique (Customer namespace!)
- Unique in the ABAP Dictionary (global ABAP data type)
- Case-insensitive (will be transferred in uppercase)
- Must differ from the name of the CDS View

# VDM - Virtual Data Model



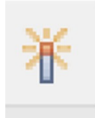
# VDM cont.

Identification
@VDM.viewType: #BASIC
@VDM.viewType: #COMPOSITE
@VDM.viewType: #TRANSACTIONAL
@VDM.viewType: #CONSUMPTION
@VDM.viewType: #CONSUMPTION @ObjectModel.transactional ProcessingDelegated:true
@VDM.lifecycle.contract.type: #PUBLIC_REMOTE_API
@VDM.private: true
@VDM.viewType: #EXTENSION
@VDM.viewExtension: true

# An example ...

Attention:  
SQL View Name  
deprecated with 7.57

Annotation:  
SQL View Name  
(classic)



```
[GNX] ZI_RS_SBOOK x
1 @AbapCatalog.sqlViewName: 'ZIRSSBOOK'
2 @AbapCatalog.compiler.compareFilter: true
3 @AbapCatalog.preserveKey: true
4 @AccessControl.authorizationCheck: #NOT_REQUIRED
5 @EndUserText.label: 'Basic View SBOOK'
6
7 @VDM.viewType: #BASIC
8
9 define view ZI_RS_SBOOK as select from sbook {
10   key carrid as Carrid,
11   key connid as Connid,
12   key fldate as Fldate,
13   key bookid as Bookid,
14   customid as Customid,
15   custtype as Custtype,
16   smoker as Smoker,
17   luggweight as Luggweight,
18   wunit as Wunit,
19   invoice as Invoice,
20   class as Class,
21   forcuram as Forcuram,
22   forcurkey as Forcurkey,
23   loccuram as Loccuram,
24   loccurkey as Loccurkey,
25   orderdate as OrderDate,
26   counter as Counter,
27   agencynum as Agencynum,
28   cancelled as Cancelled,
29   reserved as Reserved,
30   passname as Passname,
31   passform as Passform,
32   passbirth as Passbirth
33 }
34
```

Data origin

CDS View Name

Field list  
Alias

Attention:  
define view deprecated  
mit 7.57

# @Annotations

@AbapCatalog.sqlViewName: 'XXXX' — Auto creates SQL view corresponding to CDS view with given name 'XXXX'. (CDS DDIC-based View)

@AbapCatalog.preserveKey: true — Preserves the CDS key fields, note the Keyword 'Key' next to SCARR. Preserve key true ignores defaults keys from the database table/view and preserves keys defined explicitly in the View. Possible Values — true/false

@AccessControl.authorizationCheck: #CHECK — Tells whether an authorization check should be performed or not, how can we perform an authorization check in CDS? Check upcoming blogs. Possible Values — #CHECK, #NOT\_ALLOWED, #NOT\_REQUIRED, #PRIVLEDGE\_ONLY.

@EndUserText.label: 'CARDINALITY' — It's a free text which provides info to the end user. This has its own advantage in FIORI, UI5 apps etc. can be used as field catalogues. 40 Characters is the max length.

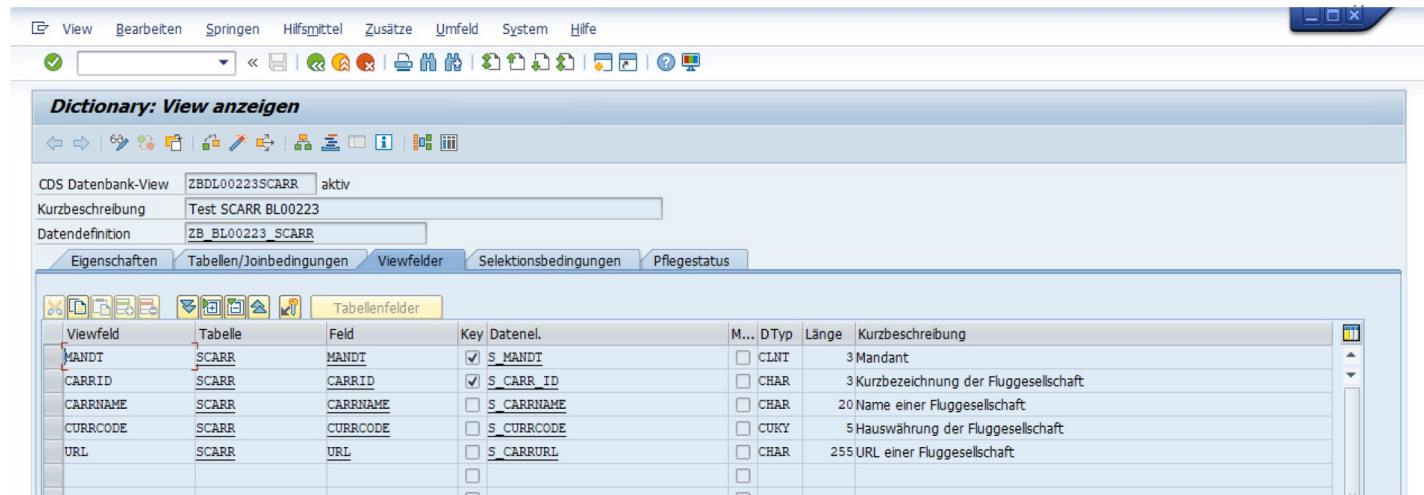
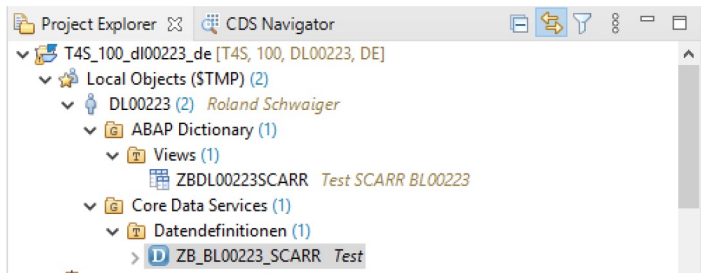
@AbapCatalog.compiler.compareFilter: true — Defines the evaluation of filter conditions in path expressions of the CDS view. Possible Values — true/false

@ClientDependent: true — Defines Client handling when Open SQL is used to access the CDS View. Is optional and the default is "true". Possible Values — true/false.

# Annotations

	ABAP Annotations (Evaluated by ABAP runtime)	Framework-specific (Evaluated by Framework like OData, VDM, UI, Analytics, ...) **
View Annotations	@AbapCatalog.viewEnhancementCategory @AccessControl.authorizationCheck @EndUserText.label	@VDM.viewType @UI.headerInfo.typeName @Analytics.query: true
Element Annotations	@Semantics.Amount.currencyCode @EndUserText.label	@UI.hidden: true @AnalyticsDetails.query.axis @Semantics.address.city
Parameter Annotations*	@Environment.systemField @EndUserText.label	...
Extension Annotations*	@AbapCatalog.sqlViewAppendName	...
Function Annotations*	@ClientDependent @EndUserText.label	...

# CDS DDIC-based View ... SQL-View in DDIC



# Views

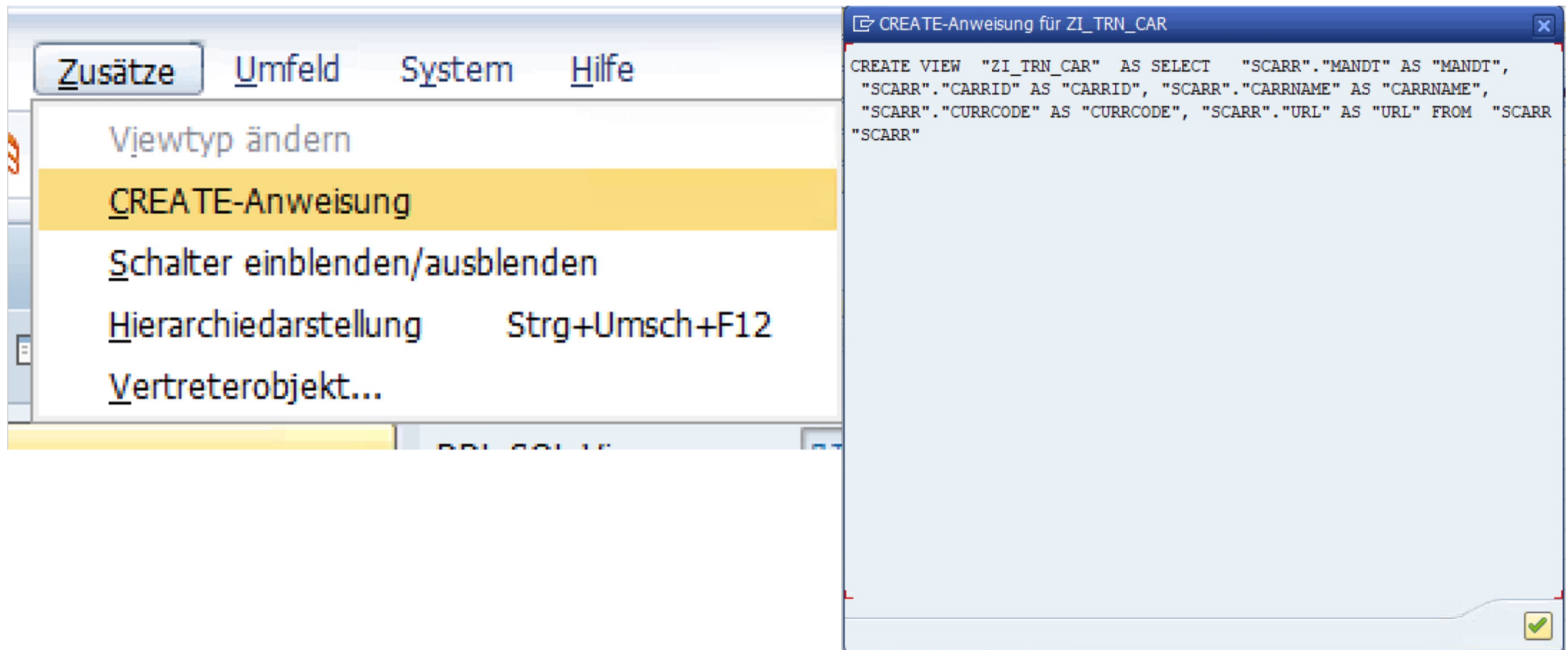
## SQL-View:

- Is Repository Object
- Is Database Object
- Is ABAP data type
- No additional semantics

## CDS-View:

- Not available in the repository
- Not in DB
- Is ABAP data type
- Additional semantics (annotations)

# CREATE Instruction



# Data Preview

The screenshot shows the SAP IDEAS interface. On the left, a tree view displays the project structure under 'Core Data Services' > 'Data Definitions', with 'ZI\_TRN\_CARRIER' selected. A context menu is open over this object, listing actions such as 'New Data Definition', 'Copy', and 'Delete'. The 'Open With' submenu is expanded, and 'Data Preview' is highlighted. On the right, a code editor shows the SQL definition for the view 'ZI\_TRN\_CARRIER'.

```
define view ZI_TRN_CARRIER as  
  mandt as Mandt,  
  carrid as Carrid,  
  carrname as Carrname,  
  currcode as Currcode,  
  url as Url  
}
```

# Association Navigation

The screenshot shows a database tool interface with a table of carrier data. A dialog box titled 'List of Associations' is open, displaying the message 'No association is defined'. Below the dialog, a tooltip reads 'To follow the association, choose an association from the list'. The table below contains the following data:

Carriid	Carrier	Currency	URL
AA	American Airlines	USD	http://w...
AB	Air Berlin	EUR	http://w...
AC	Air Canada	CAD	http://w...
AF	Air France	EUR	http://w...
AZ	Alitalia	EUR	http://w...
BA	British Airways	GBP	http://w...
CO	Continental Airl...	USD	http://w...
DL	Delta Airlines	USD	http://w...
FJ	Air Pacific	USD	http://w...
JL	Japan Airlines	JPY	http://w...
LH	Lufthansa	EUR	http://w...
NG	Lauda Air	EUR	http://w...
NW	Northwest Airli...	USD	http://w...
QF	Qantas Airways	AUD	http://w...
SA	South African Air.	ZAR	http://w...
SQ	Singapore Airlin...	SGD	http://w...
SR	Swiss	CHF	http://w...
UA	United Airlines	USD	http://w...

# SQL Console

The screenshot displays an SQL console interface. On the left, a query window shows the following SQL code:

```
SELECT
  ZI_TRN_CARRIER~CARRID,
  ZI_TRN_CARRIER~CARRNAME,
  ZI_TRN_CARRIER~CURRCODE,
  ZI_TRN_CARRIER~URL
FROM
  ZI_TRN_CARRIER
```

Below the query window, a status bar indicates: "2021-02-17 22:54:12 767 - Query successfully executed in 3.5510000 ms (server processir)". Below this, a log window shows the execution details:

```
SELECT
  ZI_TRN_CARRIER~CARRID,
  ZI_TRN_CARRIER~CARRNAME,
  ZI_TRN_CARRIER~CURRCODE,
  ZI_TRN_CARRIER~URL
FROM
  ZI_TRN_CARRIER
INTO TABLE @DATA(LT_RESULT)
UP TO 100 ROWS .
```









On the right, the "Raw Data" tab is active, showing a table with 18 rows. The table has four columns: CARRID, CARRNAME, CURRCODE, and URL. The data is as follows:

CARRID	CARRNAME	CURRCODE	URL
AA	American Airlines	USD	http://www...
AB	Air Berlin	EUR	http://www...
AC	Air Canada	CAD	http://www...
AF	Air France	EUR	http://www...
AZ	Alitalia	EUR	http://www...
BA	British Airways	GBP	http://www...
CO	Continental Airlines	USD	http://www...
DL	Delta Airlines	USD	http://www...
FJ	Air Pacific	USD	http://www...
JL	Japan Airlines	JPY	http://www...
LH	Lufthansa	EUR	http://www...
NG	Lauda Air	EUR	http://www...
NW	Northwest Airlines	USD	http://www...
QF	Qantas Airways	AUD	http://www...
SA	South African Air.	ZAR	http://www...
SQ	Singapore Airlines	SGD	http://www...
SR	Swiss	CHF	http://www...
UA	United Airlines	USD	http://www...

# Active Annotations

## Active Annotations for Entity ZI\_TRN\_CARRIER

type filter text

Annotated Elements	Annotation Value	Translated Text	Origin Data Source	Origin Data Element
<ul style="list-style-type: none"> <li>▼  ZI_TRN_CARRIER</li> <li>▼  Entity annotations           <ul style="list-style-type: none"> <li>▼  @AbapCatalog               <ul style="list-style-type: none"> <li>sqlViewName 'ZI_TRN_CAR'</li> <li>▼ compiler                   <ul style="list-style-type: none"> <li>compareFilter true</li> <li>preserveKey true</li> </ul> </li> </ul> </li> <li>▼  @AccessControl               <ul style="list-style-type: none"> <li>authorizationCheck #CHECK</li> </ul> </li> <li>▼  @EndUserText               <ul style="list-style-type: none"> <li>label Carrier</li> </ul> </li> </ul> </li> <li>▼  Carrid           <ul style="list-style-type: none"> <li>▼  @EndUserText               <ul style="list-style-type: none"> <li>quickInfo Airline Code SCARR S_CARR_ID</li> <li>label Airline SCARR S_CARR_ID</li> <li>heading ID SCARR S_CARR_ID</li> </ul> </li> </ul> </li> <li>▼  Carrname</li> </ul>				

# SQL Dependency Tree

SQL Name	SQL Relation	Object Type	Entity Name	Database Object
<ul style="list-style-type: none"> <li> <ul style="list-style-type: none"> <li>ZI_TRN_CAR</li> <li>SCARR</li> </ul> </li> </ul>	From	Database Table (TABL)	ZI_TRN_CARRIER	Information not available

SQL Dependency Graph

```

graph TD
    ZI_TRN_CAR[ZI_TRN_CAR] --> SCARR[SCARR]
    
```

Find:

SQL Dependency Tree | SQL Dependency Graph

Next Previous

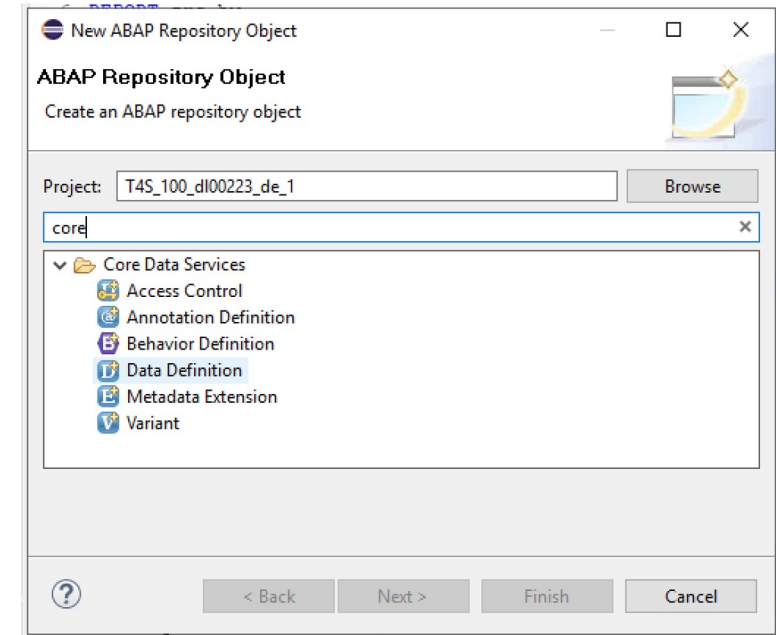
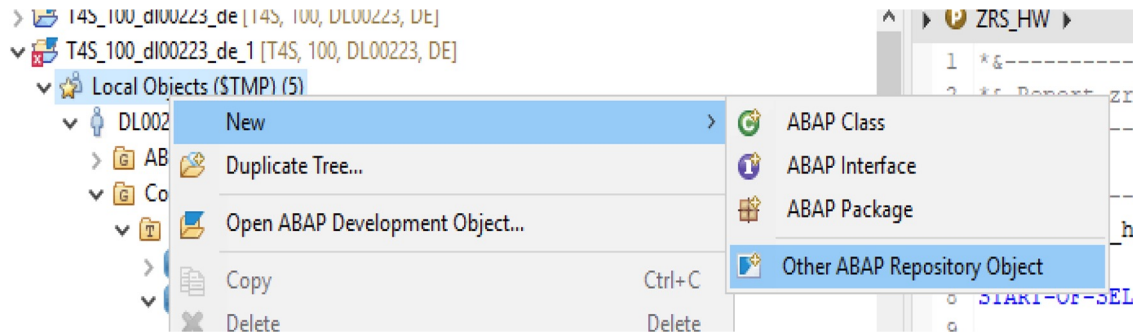
# Understanding CDS View and Using it in Open SQL

# 2



# CDS Data Definition

# Creating CDS Data Definition (classic)



# Create Data Definition

**New Data Definition**

**Data Definition**  
Create a data definition

Project: \* T4S\_100\_d100223\_de\_1

Package: \* \$TMP

Add to favorite packages

Name: \* ZI\_RS\_SBOOK

Description: \* Buchungen

Original Language: DE

Referenced Object: SBOOK

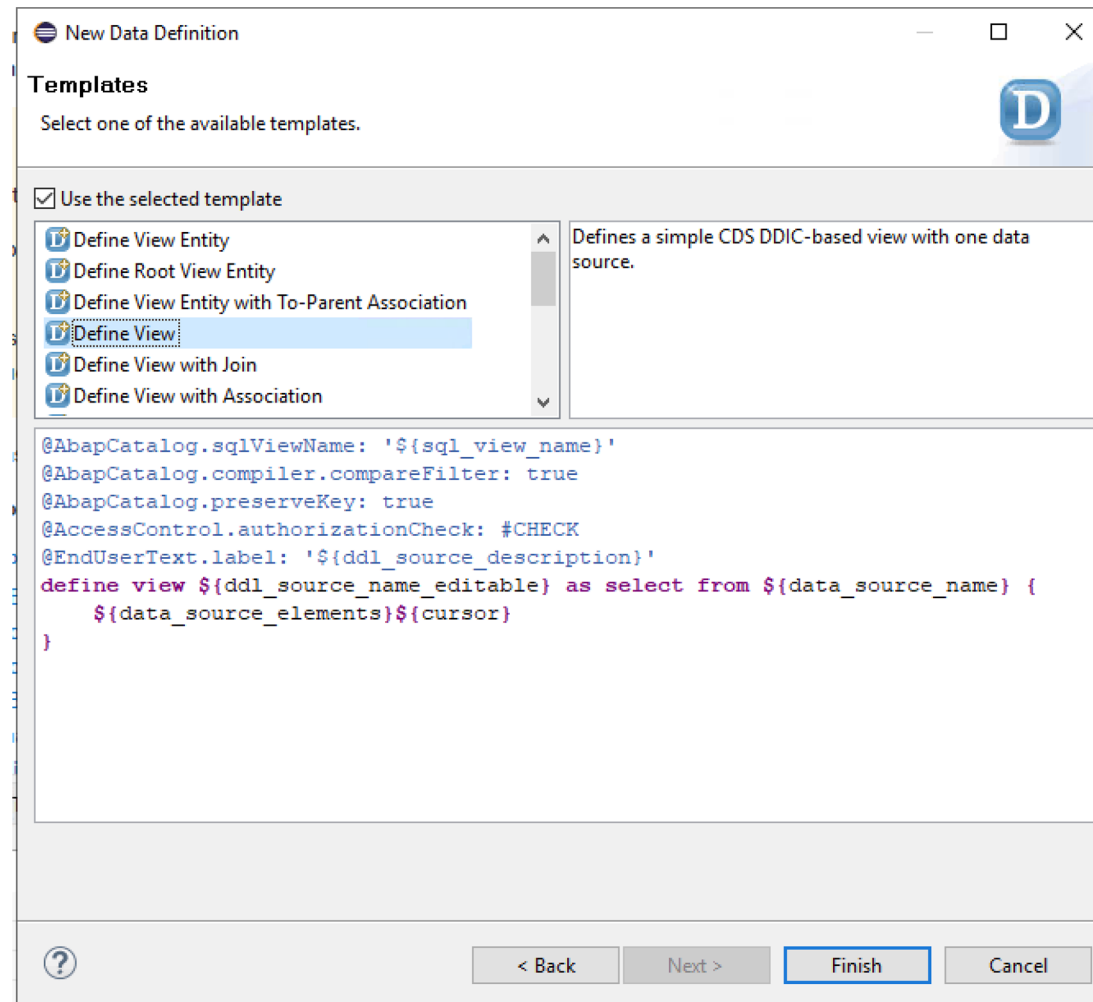
**Search Results:**

- SBOOK - table
- SBOOK\_T - structure

2 results

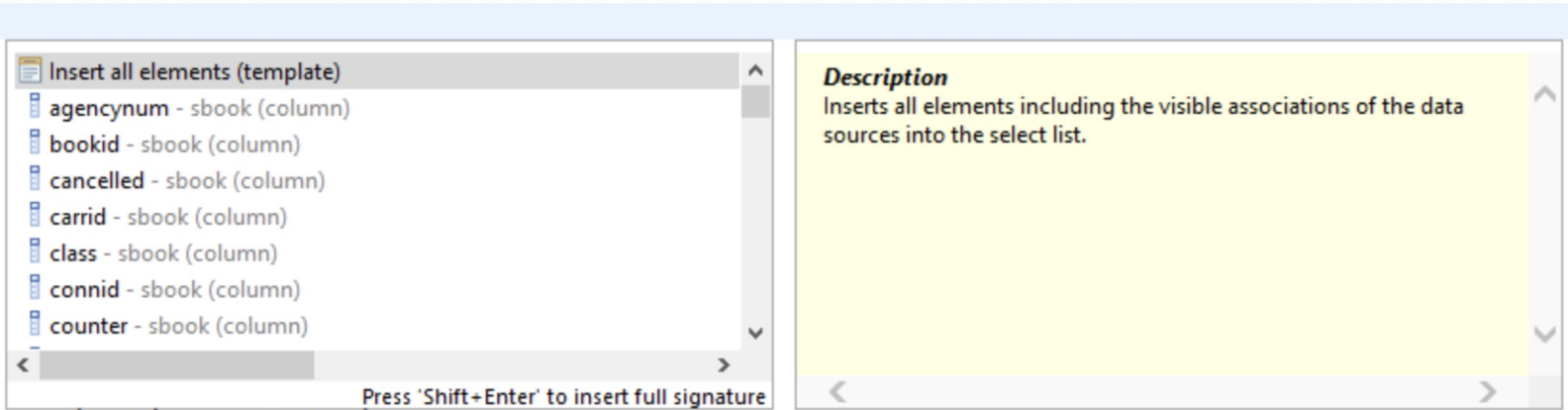
# Data Definition Template

## Define View (classic), Define View Entity



# Inserting Data Definition Elements

```
1 @AbapCatalog.sqlViewName: 'ZIRSSBOOK2'  
2 @AbapCatalog.compiler.compareFilter: true  
3 @AbapCatalog.preserveKey: true  
4 @AccessControl.authorizationCheck: #NOT_REQUIRED  
5 @EndUserText.label: 'CDS View für Buchungen'  
6 define view ZI_RS_SBOOK2 as select from sbook {  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19 invoice as Invoice,  
20 class as Class,  
21 forcuram as Forcuram,  
22 forcurkey as Forcurkey,  
23 loccuram as Loccuram,
```



**Insert all elements (template)**

- agencynum - sbook (column)
- bookid - sbook (column)
- cancelled - sbook (column)
- carrid - sbook (column)
- class - sbook (column)
- connid - sbook (column)
- counter - sbook (column)

**Description**  
Inserts all elements including the visible associations of the data sources into the select list.

Press 'Shift+Enter' to insert full signature

# SQL View Name, like in DDIC

```
*[GNX] ZI_RS_SBOOK ×  
1 Bitte für ZIRSSBOOKKKKKKKKKK einen kürzeren Namen wählen OOKkkkkkkkkk'  
2 @AbapCatalog.compiler.compareFilter: true  
3 @AbapCatalog.preserveKey: true  
4 @AccessControl.authorizationCheck: #NOT_REQUIRED  
5 @EndUserText.label: 'Basic View SBOOK'  
6  
7 @VDM.viewType: #BASIC  
8  
9 define view ZI_RS_SBOOK as select from sbook {  
10     key carrid as Carrid,  
11     key connid as Connid,
```

# Data Definition



```
1 @AbapCatalog.sqlViewName: 'ZIRSSBOOK2'  
2 @AbapCatalog.compiler.compareFilter: true  
3 @AbapCatalog.preserveKey: true  
4 @AccessControl.authorizationCheck: #NOT_REQUIRED  
5 @EndUserText.label: 'CDS View für Buchungen'  
6 define view ZI_RS_SBOOK2 as select from sbook {  
7     key carrid as Carrid,  
8     key connid as Connid,  
9     key fldate as Fldate,  
10    key bookid as Bookid,  
11    customid as Customid,  
12    custtype as Custtype,  
13    smoker as Smoker,  
14    luggweight as Luggweight,  
15    wunit as Wunit,  
16    invoice as Invoice,  
17    class as Class,  
18    forcuram as Forcuram,  
19    forcurkey as Forcurkey,  
20    loccuram as Loccuram,  
21    loccurkey as Loccurkey,  
22    order_date as OrderDate,  
23    counter as Counter,
```

# Create a legacy SQL-View based CDS

# 3



# DEFINE VIEW -> DEFINE VIEW ENTITY

## DEFINE VIEW

- Die *alte* Syntax für CDS-Views (bis ABAP 7.54 üblich).
- Erstellt im ABAP Dictionary ein **SQL View** (technischer Name über @AbapCatalog.sqlViewName).
- Es entsteht **zwei Objekte**:
  - CDS-View (Repository-Objekt im Paket)
  - SQL View im Dictionary (für DB und Open SQL)

## DEFINE VIEW ENTITY

- Die *neue* Syntax (ab ABAP 7.55).
- Erstellt ein einziges Objekt: die **View Entity** im ABAP Dictionary.
- Kein zusätzliches SQL View Objekt mehr nötig → **einfacher, moderner, performanter**.
- Kein @AbapCatalog.sqlViewName mehr nötig.
- **Nur noch ein Objekt** im Dictionary (keine Verdoppelung).
- Zukunftssicher: **neue CDS-Features** (z. B. **CDS Table Functions mit View Entities**, RAP, Annotations) funktionieren nur mit view entity.

# CDS View Create

4



# CDS View Inner Join, Projection and Selection

# View Inner Join, Projection and Selection

- An **INNER JOIN** combines rows from two (or more) data sources where the **join condition matches**.
  - If no match → row is excluded.
- **Projection** = selecting only the required fields instead of all columns.
- **Selection** = restricting data with WHERE clause.

# Inner Join, Projection and Selection - Example

define view entity ZC\_FlightOverview

as select from sflight as F

inner join scarr as C  
on F.carrid = C.carrid

{

key F.carrid,

key F.connid,

F.fldate,

F.price,

C.carrname

}

where

F.fldate between '20250101' and '20251231'

and F.price > 200

New Data Definition  
Data Definition  
Create a data definition

Project: \* T4S\_100\_dil00223\_de\_1 Browse...

Package: \* STMP Browse...

Add to favorite packages

Name: \* ZI\_RS\_JAS

Description: \* Join & Select

Original Language: DE

Referenced Object: \* SBOOK Browse...

< Back Next > Finish Cancel

New Data Definition  
Templates  
Select one of the available templates.

Use the selected template

- Define View Entity
- Define Root View Entity
- Define View Entity with To-Parent Association
- Define View
- Define View with Join
- Define View with Association
- Define View with To-Parent Association
- Define View with Parameters

Defines a CDS DDIC-based view which combines two data sources using a left outer join.  
The join conditions are specified in the on clause.

```
@AbapCatalog.sqlViewName: '${sql_view_name}'  
@AbapCatalog.compiler.compareFilter: true  
@AbapCatalog.preserveKey: true  
@AccessControl.authorizationCheck: #CHECK  
@EndUserText.label: '${ddl_source_description}'  
define view ${ddl_source_name_editable} as select from ${data_source_name}  
left outer join ${joined_data_source_name}  
on ${data_source_name}.${element_name} = ${joined_data_source_name}.${joined_  
${data_source_elements}${cursor}
```

< Back Next > Finish Cancel

# CDS View Definition Inner Join, Projection and Selection

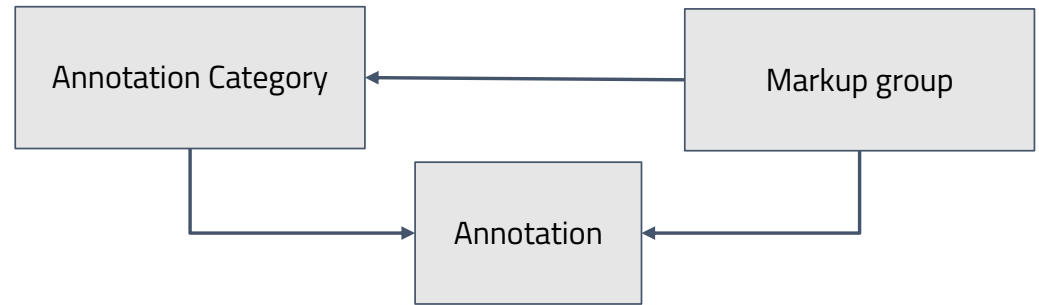
# 5



# DDL Source and Annotations

# Annotationen

- View-Annotationen
  - @AbapCatalog.sqlViewName
- Element-Annotationen
  - @Semantics.unitOfMeasure
- Parameter-Annotationen
  - @Environment.systemField
- Erweiterungs-Annotationen
  - @AbapCatalog.sqlViewAppendName
- Funktions-Annotationen
  - @ClientDependent
- FIORI
  - @UI
- OData
  - @OData



# Annotations

## Properties

- Enriching the definition with metadata
- Starts with the @ sign
- Scope
  - View
  - Element
  - Parameter
  - ...

Viewfeld	Tabelle	Feld	Key	Datenein.	M...	DTyp	Länge
MANDT	SCARR	MANDT	<input checked="" type="checkbox"/>	S_MANDT	<input type="checkbox"/>	CLNT	
CARRID	SCARR	CARRID	<input checked="" type="checkbox"/>	S_CARR_ID	<input type="checkbox"/>	CHAR	
CARRNAME	SCARR	CARRNAME	<input type="checkbox"/>	S_CARRNAME	<input type="checkbox"/>	CHAR	2
CURRCODE	SCARR	CURRCODE	<input type="checkbox"/>	S_CURRCODE	<input type="checkbox"/>	CUKY	
URL	SCARR	URL	<input type="checkbox"/>	S_CARRURL	<input type="checkbox"/>	CHAR	25

```
@AbapCatalog.sqlViewName: 'ZI_TRN_CAR'  
@AccessControl.authorizationCheck: #CHECK  
@EndUserText.label: 'Carrier'  
define view ZI_TRN_CARRIER as select from scarr  
{  
  key mandt as Mandt,  
  carrid as Carrid,  
  carrname as Carrname,  
  
  @Semantics.currencyCode: true  
  currcode as Currcode,  
  
  @Semantics.url: true  
  url as Url  
}
```

# Annotations

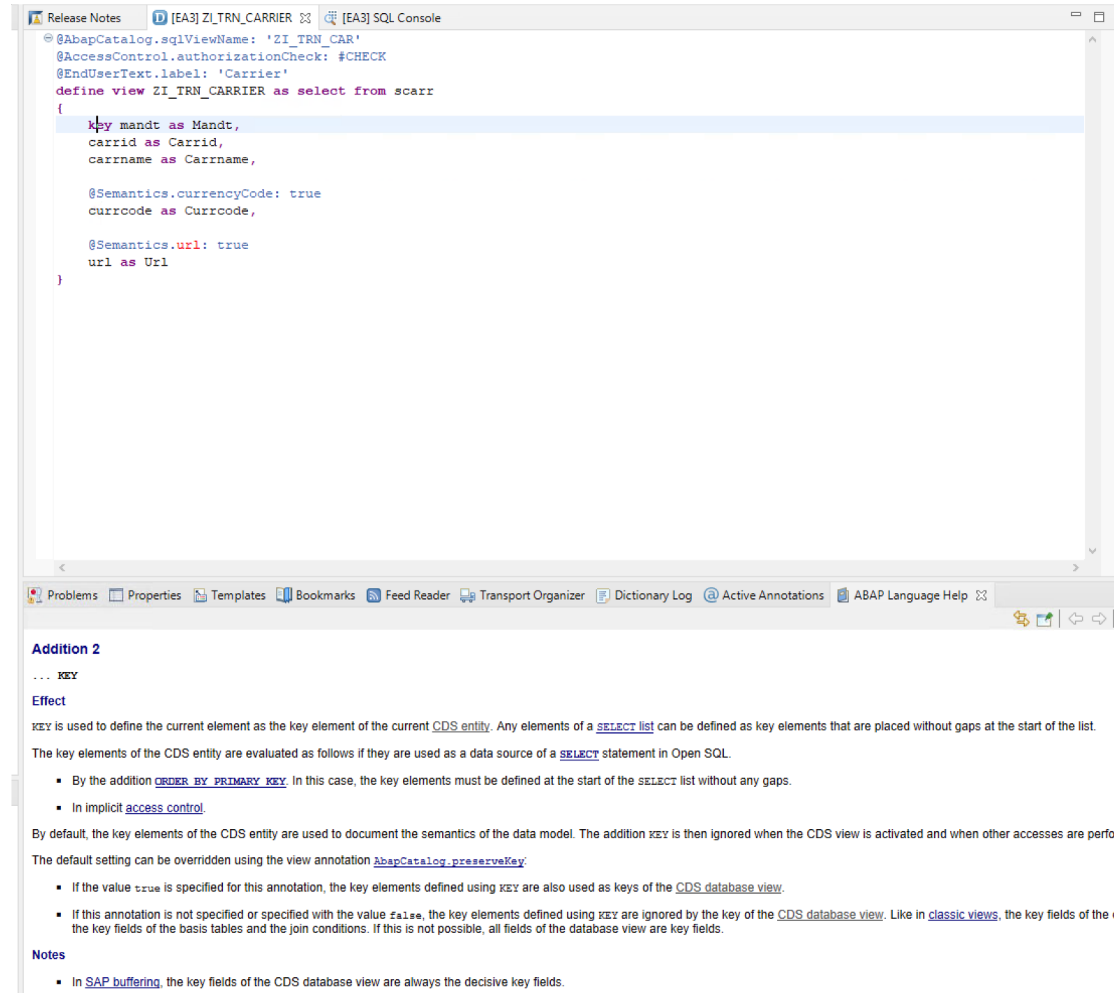
```
Release Notes [EA3] ZI_TRN_CARRIER [EA3]  
⊖ @AbapCatalog.sqlViewName: 'ZI_TRN_CAR'  
  @AbapCatalog.compiler.compareFilter: true  
  @AbapCatalog.preserveKey: true  
  @AccessControl.authorizationCheck: #CHECK  
  @EndUserText.label: 'Carrier'  
  define view ZI_TRN_CARRIER as select from scarr {  
    mandt as Mandt,  
    carrid as Carrid,  
    carrname as Carrname,  
    currcode as Currcode,  
    |  
    @Semantics.url: true  
    url as Url  
  }
```

View Annotationen  
(before View  
keyword)

Element Annotationen  
(usually before the element,  
can also be inserted after the  
element)

# Help - there is also as before ;-)

# F1



The screenshot shows the SAP SQL Console interface. The top pane displays the SQL code for a CDS view named 'ZI\_TRN\_CARRIER'. The code includes annotations for authorization, user text, and semantics, and defines a view with key elements 'mandt', 'carrid', and 'carrname'. The bottom pane shows the help text for the 'KEY' annotation, explaining its effect and usage in CDS views.

```
@AbapCatalog.sqlViewName: 'ZI_TRN_CAR'
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Carrier'
define view ZI_TRN_CARRIER as select from scarr
{
  key mandt as Mandt,
  carrid as Carrid,
  carrname as Carrname,

  @Semantics.currencyCode: true
  currcode as Currcode,

  @Semantics.url: true
  url as Url
}
```

### Addition 2

... **KEY**

#### Effect

**KEY** is used to define the current element as the key element of the current **CDS entity**. Any elements of a **SELECT list** can be defined as key elements that are placed without gaps at the start of the list.

The key elements of the CDS entity are evaluated as follows if they are used as a data source of a **SELECT** statement in Open SQL.

- By the addition **ORDER BY PRIMARY KEY**. In this case, the key elements must be defined at the start of the **SELECT** list without any gaps.
- In implicit **access control**.

By default, the key elements of the CDS entity are used to document the semantics of the data model. The addition **KEY** is then ignored when the CDS view is activated and when other accesses are performed.

The default setting can be overridden using the view annotation **AbapCatalog.preserveKey**.

- If the value **true** is specified for this annotation, the key elements defined using **KEY** are also used as keys of the **CDS database view**.
- If this annotation is not specified or specified with the value **false**, the key elements defined using **KEY** are ignored by the key of the **CDS database view**. Like in **classic views**, the key fields of the basis tables and the join conditions. If this is not possible, all fields of the database view are key fields.

#### Notes

- In **SAP buffering**, the key fields of the CDS database view are always the decisive key fields.

# Important annotations ABAP-View

Client handling controlled in Open SQL

`@ClientHandling.type: #INHERITED`

`@ClientHandling.algorithm: #AUTOMATED`

Access control

`@AccessControl.authorizationCheck: #CHECK`

Controlling the evaluation of filtered associations

`@AbapCatalog.compiler.CompareFilter: true`

Buffering SQL View in Open SQL

`@AbapCatalog.Buffering.type: #GENERIC`

`@AbapCatalog.Buffering.numberOfKeyFields: 1`

`@AbapCatalog.Buffering.status: #ACTIVE`

# ABAP Annotations for View Elements

## Amount

```
@Semantics.amount.currencyCode: 'Currency'
```

```
price,
```

```
Currency @<Semantics.currencyCode: true, //After element possible too
```

## Quantity

```
@Semantics.quantity.unitOfMeasure: 'Unit' //<-Use alias here
```

```
distance,
```

```
@Semantics.unitOfMeasure: true
```

```
unit as Unit,
```



# ABAP annotations for translatable texts (SE63, short text, A5 User Interface Texts, DDLS CDS view)

## View-Annotation

```
@EndUserText.label: 'My Text' //Max 60 Characters
```

## Element-Annotation

```
@EndUserText.label: 'Field Identifiers' //Max 60 Characters
```

```
@EndUserText.quickInfo: 'And another explanation'  
afield,
```

# Annotation texts with method in ABAP

To evaluate the semantic properties (annotations) of a CDS view (stored in system tables), a corresponding API (CL\_DD\_DDL\_ANNOTATION\_SERVICE, if available in your system) should be used.

## Method Call

```
CL_DD_DDL_ANNOTATION_SERVICE=>GET_LABEL_4_ELEMENT(PA_CDS,  
PA_ELEM, PA_LANG)
```

## Parameter

PA_CDS:	DDSTRUCOBJNAME
PA_ELEM:	DDFIELDNAME_L
PA_LANG:	SY-LANGU

# DDL Source and Annotation

5c



# Expressions

# SQL Expressions

- **Expressions** are operations that can be used in the **field list, WHERE clause, GROUP BY, or HAVING** of a CDS view.
- They allow you to **calculate, transform, and conditionally derive values** at the database level.
- Expressions are **evaluated in the database** (pushdown principle).

Types of expressions:

- **Arithmetic Expressions**
- **String Expressions**
- **Date & Time Expressions**
- **CASE Expressions (Conditional Logic)**

# SQL Expressions - Examples

- **Literals (characters, numbers)**

- 'Hello' as Character1
- 1 as Integer1

- **Type conversions (Target Type: Predefined Dictionary Types, ABAP Types: abap.int4, S\_CARR\_ID, ...)**

- CAST ( '20250916' AS /dmo/flight\_date) as Fldate

- **CASE Expressions (Conditional Logic)**

CASE operand // class

    WHEN operand1 THEN Value1 // WHEN 'Y' THEN 'Yes'

    WHEN operand2 THEN Value2

    ...

    ELSE Valuen

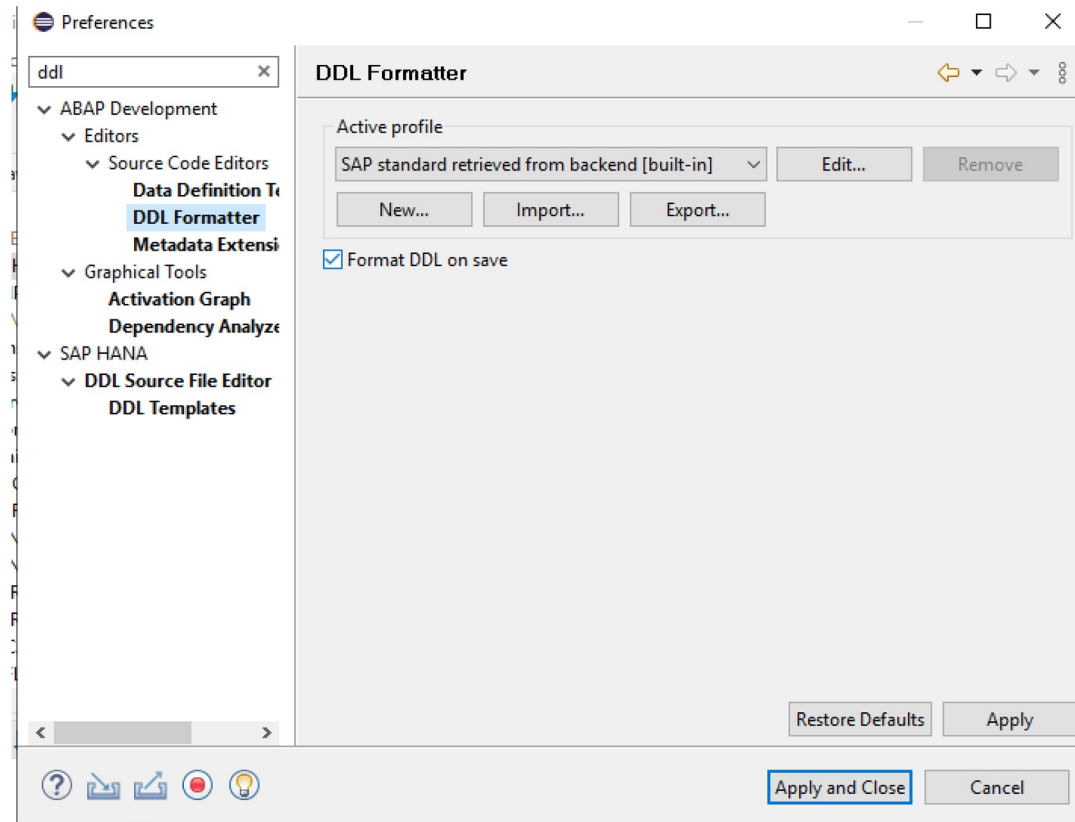
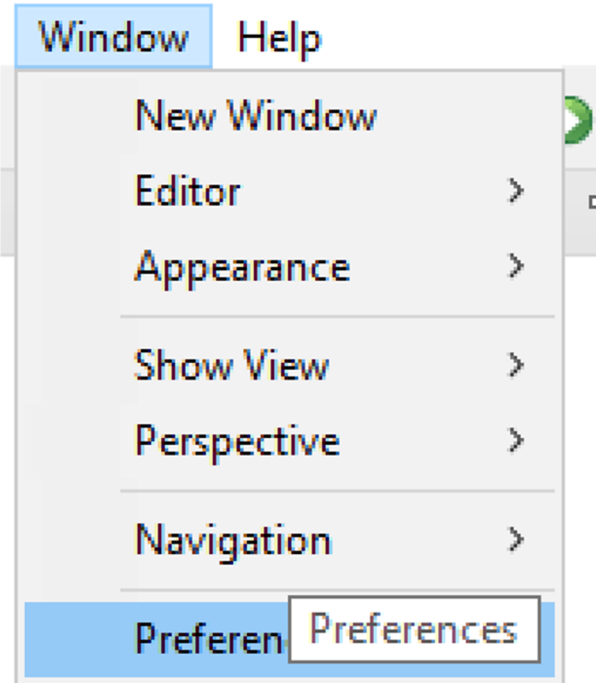
END AS Classtext //Goal

- **Arithmetic expressions (+,-,\*,/)**

- seatsmax - seatsocc as FreeSeats,
- price \* seatsocc as Revenue

Which types can be converted to which type( 7.52 ) ABAP CDS - cast\_expr - ABAP Keyword Documentation (sap.com)  
<[https://help.sap.com/doc/abapdocu\\_752\\_index\\_htm/7.52/de-DE/abencds\\_f1\\_cast\\_expression.htm](https://help.sap.com/doc/abapdocu_752_index_htm/7.52/de-DE/abencds_f1_cast_expression.htm)>

# Auto Format DDL on Save



# Using Expressions

6



# Built in Functions

# Built-in Functions

- CDS provides a large set of **built-in functions** that can be used directly in view definitions.
- They are evaluated on the **database level (pushdown)**, so they are highly efficient.
- We can group them into several categories:
  - **Arithmetic Functions**
  - **String Functions**
  - **Date & Time Functions**
  - **Conversion Functions**
  - **Boolean Functions**

# Built-in SQL functions

- Arithmetic Functions

`div(ar1,arg2), mod(arg1,arg2), division(arg1,arg2,dec)`

- Rounding Functions

`abs(arg), floor(abs), ceil(arg), round(arg,pos)`

- String functions

7.40: `concat, replace, substring, ..`

7.50: `concat_with_space, length, left, right, ..`

7.51: `lower, upper, ..`

- Currency and Unit Conversion

`Unit_Conversion (DBTab T006), Currency_Conversion (DBTab TCUR)`

- Calculations with dates

`date_is_valid, dats_days_between, dats_add_days, dats_add_months`

# Built-in Functions – Example CASE

```
define view entity ZC_Case
  as select from sflight
  {
    carrid,
    price,
    case
      when price < 200 then 'CHEAP'
      when price between 200 and 500 then 'MEDIUM'
      else 'EXPENSIVE'
    end as PriceCategory
  }
```

# Use built-in functions

7



# Nested Views

# Nested Views – What?

- In ABAP CDS, a **nested view** is simply a CDS view that is based on another CDS view.
- Instead of building one very complex view, you can **layer views**:
- **Base view(s)**: provide raw data selection.
- **Consumption view(s)**: enrich or aggregate data for reporting, OData exposure, or analytics.
- This approach promotes **reusability, readability, and separation of concerns**.

# Nested Views – Why?

- **Reusability:** A base view can be used in multiple higher-level views.
- **Simplification:** Split complex logic into smaller, understandable parts.
- **Consistency:** KPIs or calculated fields defined once in a base view are reused consistently everywhere.
- **Performance:** Push-down to DB still applies — the database optimizer handles the nested queries efficiently.

# Nested View - Example

## Interface View

```
@EndUserText.label: 'Flight Interface View'
define view entity I_Flight
  as select from sflight
{
  key carrid      as AirlineID,
  key connid     as ConnectionID,
  fldate        as FlightDate,
  price         as FlightPrice,
  currency      as CurrencyCode
}
```

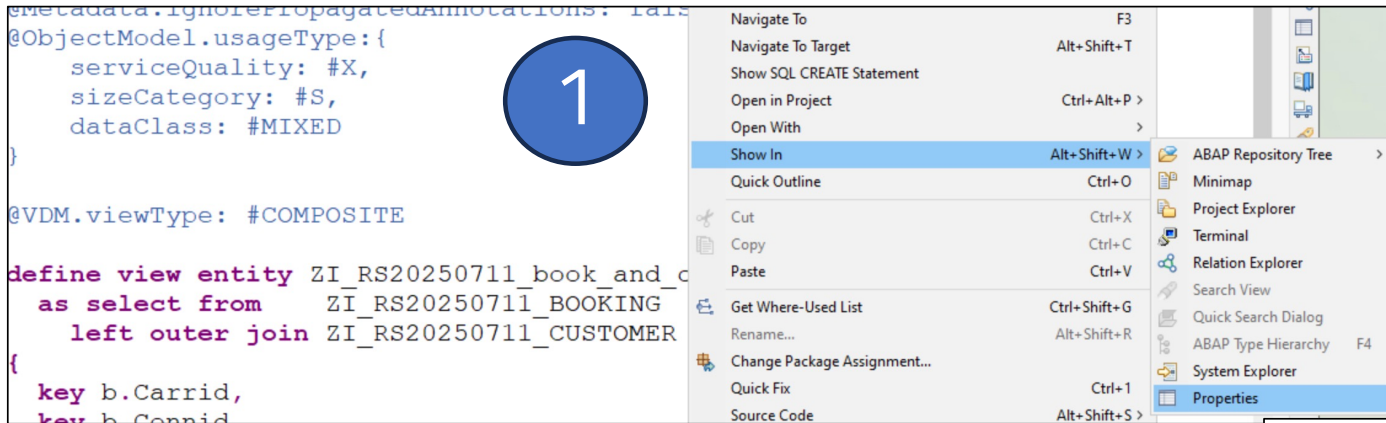
- Acts as a clean, reusable **semantic layer**.
- No aggregations, no UI annotations — pure data access

## Composite View

```
@EndUserText.label: 'Flight Aggregates by Airline'
define view entity C_FlightKPI
  as select from I_Flight
{
  key AirlineID,
  COUNT( * ) as NumFlights,
  AVG( FlightPrice ) as AvgPrice,
  SUM( FlightPrice ) as TotalRevenue,
  MIN( FlightPrice ) as MinPrice,
  MAX( FlightPrice ) as MaxPrice
}
group by AirlineID
```

- **Business logic layer.**
- Can be reused by multiple reports, apps, or analytics queries.

# Change Description of CDS DD



1

```
@metadata.ignorePropagatedAnnotations: false
@ObjectModel.usageType: {
  serviceQuality: #X,
  sizeCategory: #S,
  dataClass: #MIXED
}

@VDM.viewType: #COMPOSITE

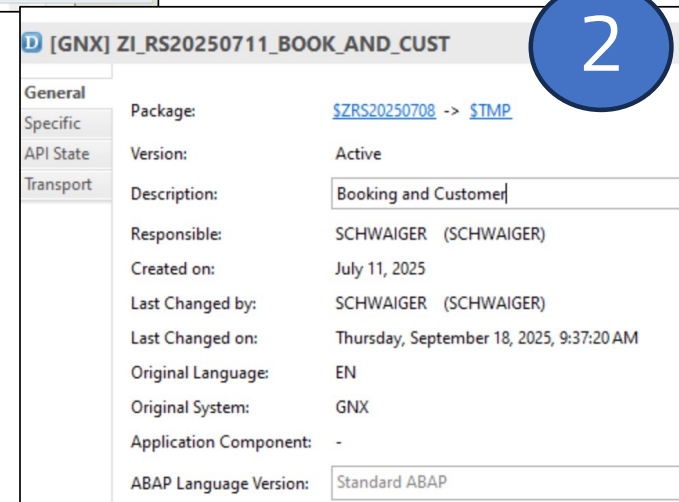
define view entity ZI_RS20250711_book_and_c
  as select from ZI_RS20250711_BOOKING
  left outer join ZI_RS20250711_CUSTOMER
  {
    key b.Carrid,
    key b.Connid
```

Context menu options:

- Navigate To (F3)
- Navigate To Target (Alt+Shift+T)
- Show SQL CREATE Statement
- Open in Project (Ctrl+Alt+P >)
- Open With (>)
- Show In (Alt+Shift+W >)
- Quick Outline (Ctrl+O)
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Get Where-Used List (Ctrl+Shift+G)
- Rename... (Alt+Shift+R)
- Change Package Assignment...
- Quick Fix (Ctrl+I)
- Source Code (Alt+Shift+S >)

Sub-menu options for 'Show In':

- ABAP Repository Tree >
- Minimap
- Project Explorer
- Terminal
- Relation Explorer
- Search View
- Quick Search Dialog
- ABAP Type Hierarchy (F4)
- System Explorer
- Properties



2

[GNX] ZI\_RS20250711\_BOOK\_AND\_CUST

General	Package:	<a href="#">SZRS20250708</a> -> <a href="#">\$TMP</a>
Specific	Version:	Active
API State	Description:	Booking and Customer
Transport	Responsible:	SCHWAIGER (SCHWAIGER)
	Created on:	July 11, 2025
	Last Changed by:	SCHWAIGER (SCHWAIGER)
	Last Changed on:	Thursday, September 18, 2025, 9:37:20 AM
	Original Language:	EN
	Original System:	GNX
	Application Component:	-
	ABAP Language Version:	Standard ABAP

3

# Activate!

# Nested Views

8



# Aggregations

# Aggregation – What?

- In ABAP CDS (Core Data Services), aggregations are used to **summarize data across multiple rows**. Instead of returning all detailed records, aggregate functions compute a single value (or a smaller set of values) per grouping. They are very similar to SQL aggregates but integrated into the CDS modeling approach.
- Aggregations are typically used in **analytical scenarios, reporting views**, or whenever you need aggregated KPIs.

# Aggregations Functions

## MIN, MAX

- Returns the **smallest/largest value** of an expression within the grouped set.
- Example: The earliest hire date in an organization.

## SUM

- Returns the **total sum** of numeric values.
- Example: Total sales amount per customer.

## AVG

- Returns the **average value** (arithmetic mean).
- Example: Average ticket price per airline.

## COUNT

- Returns the **number of rows** in a group.
- Example: Number of flights per carrier.

# Aggregations - Example

## Grouping and Aggregations

- Aggregations are only meaningful in combination with GROUP BY.
- Without GROUP BY, the entire dataset is treated as a single group.

## Example with multiple aggregates:

define view entity ZSalesSummary

as select from zsales

{

customer\_id,

COUNT(\*) as NumOrders,

SUM(amount) as TotalAmount,

AVG(amount) as AvgOrder,

MIN(amount) as MinOrder,

MAX(amount) as MaxOrder

}

group by customer\_id

# Aggregations

9



# Input Parameters

# Parameters – what for?

- Input for SQL expressions and functions
- Selection of data to be used in aggregations
- Mandatory selection criteria
- Filter on association

# Input parameter

Define view ...  
with parameters

par1: type, // abap.char(10), s\_carr\_id, ...

par2: type,

@Environment.systemField: #SYSTEM\_DATE

p\_dats: syst\_datum, // Optional Parameters

@EndUserText.label: 'Personell Number'

p\_pernr: persno

...

\$parameters.par1 //Access, Usage

```

Check Feature Support (Before Release 7.50)
DATA: gt_feature_set TYPE CL_ABAP_DBFEATURES=>features_set_t.
INSERT CL_ABAP_DBFEATURES=>view_with_parameters INTO TABLE
gt_feature_set.
IF CL_ABAP_DBFEATURES=>use_feature( gt_feature_set ) = abap_false.
...
ENDIF.
    
```

## Default Values

Value for the Annotation	Related ABAP system field
#CLIENT	client (sy-mandt)
#SYSTEM_DATE	system date (sy-datum)
#SYSTEM_TIME	system time (sy-zeit)
#SYSTEM_LANGUAGE	language (sy-langu)
#USER	user name (sy-uname)
#USER_DATE	user date (sy-datlo)
#USER_TIMEZONE	user time zone (sy-zonlo)

Session Variables = ABAP system fields

Session Variable	Related ABAP system field
\$session.client	sy-mandt
\$session.system_date	sy-datum
\$session.system_language	sy-langu
\$session.user	sy-uname
\$session.user_date	sy-datlo
\$session.user_timezone	sy-zonlo

To access ABAP system fields in ABAP CDS,  
prefer annotated input parameters (see next lesson)!

# Access to input parameters

Define view ...

with parameters

```
    par1: type, // abap.char(10), s_carr_id, ...
```

```
    par2: type,
```

```
as select from ...{
```

```
  $parameters.par1 as param1,
```

```
  :par2 as param2,
```

```
}
```

Where to use parameters in CDS?

- in the element list
- in expressions
- in where or having clause
- in on condition of joins

Where to use parameters in applications?

- in call of cds with parameters

```
    as select from <cds>( par1:
```

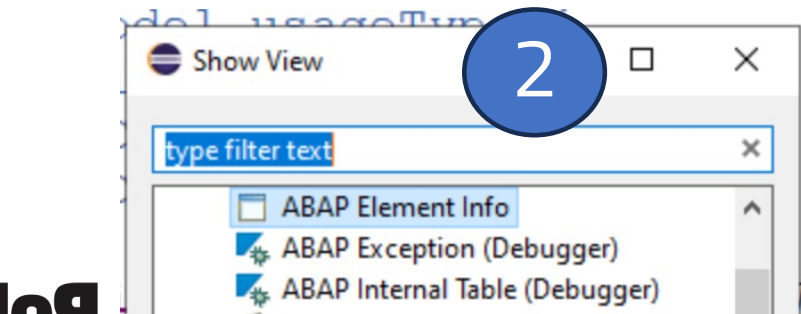
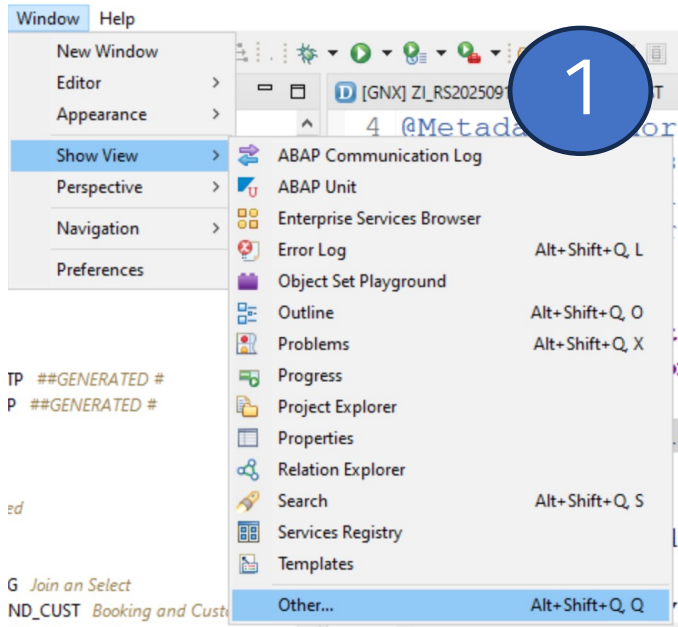
```
    <value>, par2: <value>, ...)
```

- in ABAP SQL

```
    SELECT FROM <cds>( par1 =
```

```
    <value>, par2 = <value>, ...) ...
```

# ABAP Element Info



```

12
13 key c.Customid,
14     c.Name,
15     c.Street,
16     c.Postcode,
17     c.City,
18     c.Country,

```

Problems Feed Reader Properties Templates ABAP Element Info X ABAP Langu

**Customid** in zi\_rs20250917\_book\_and\_cust  
Customer Number

Component Type	Data Type
s_customer	numc(8)

Field Label	Text	Length
Short:	Cust. No.	10
Medium:	Customer Number	15
Long:	Customer Number	20
Heading:	Cust. No.	10

Input parameter

10

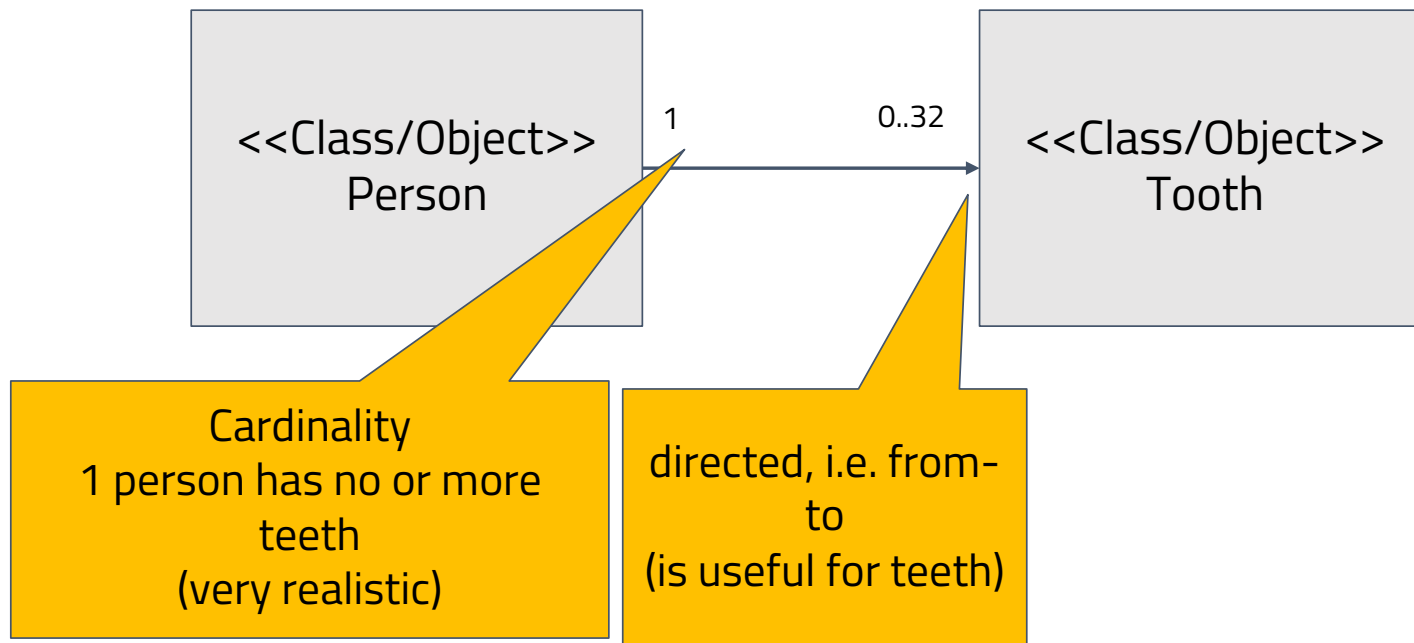


# Associations

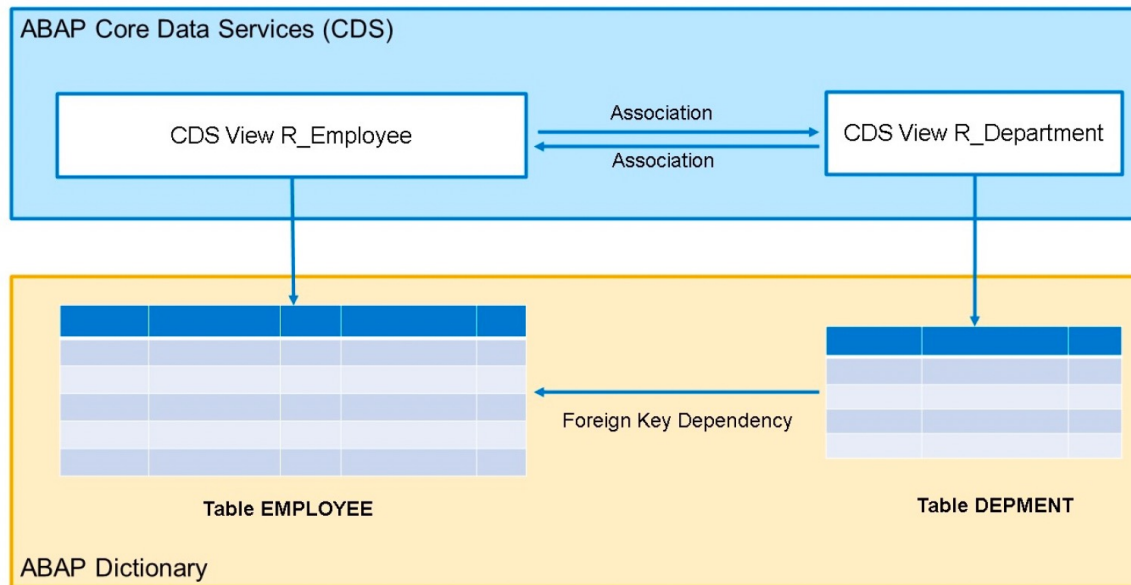
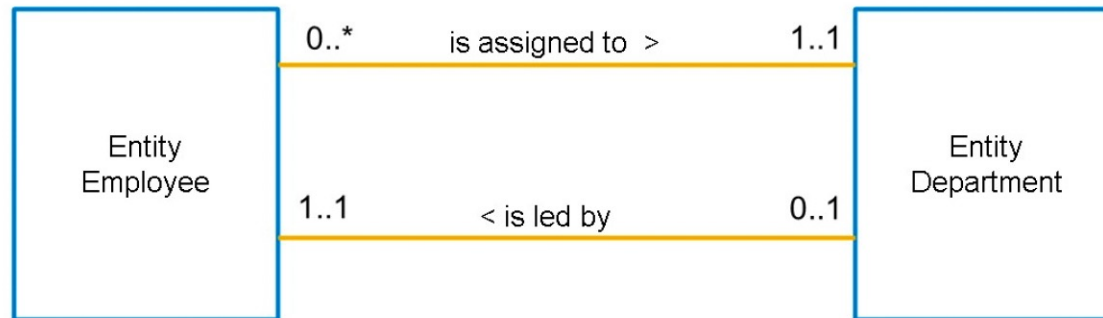
# Relationships

- How to model relations between entity types?
- Where have you seen relationships in technical systems before?

# Associations - UML / ERM



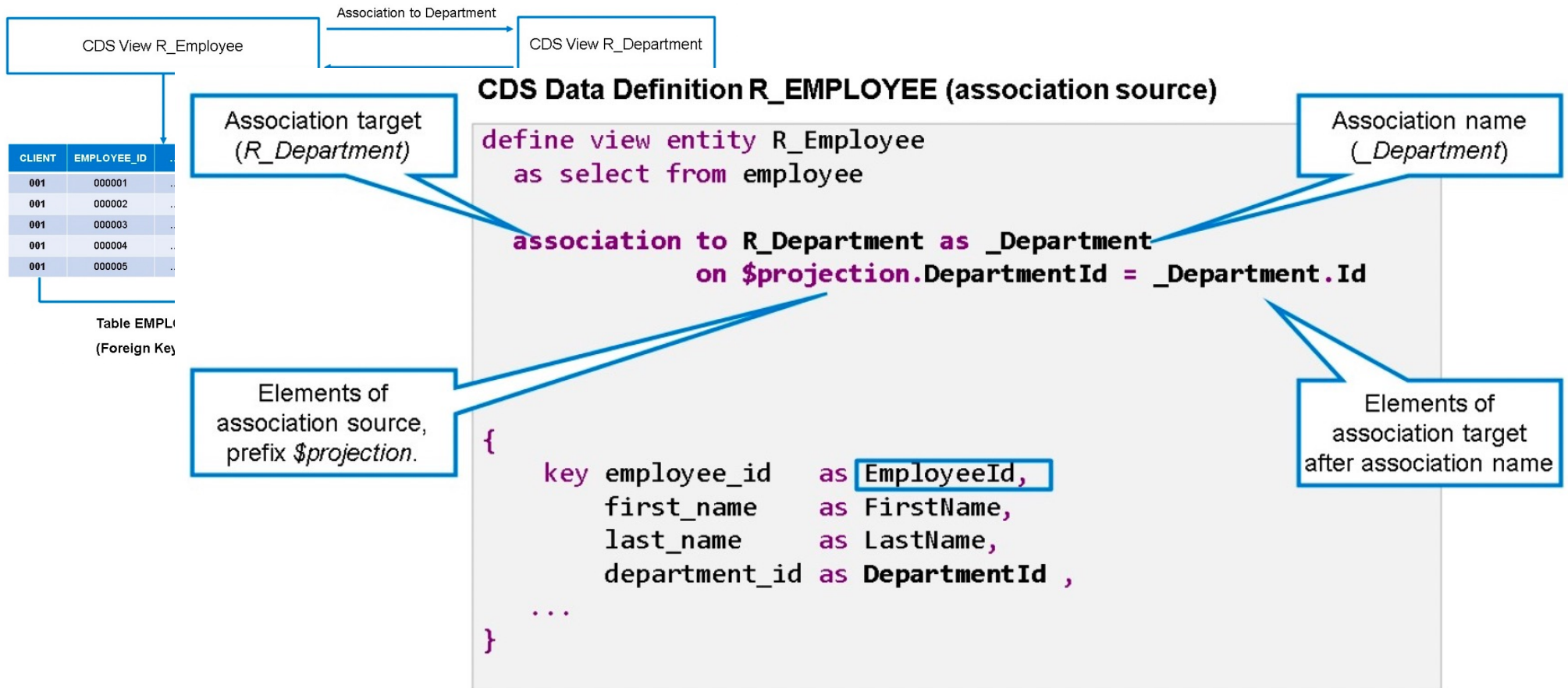
# Relationships between entity types



# Associations CDS

- Description of relationships
  - e.g. a join
  - Associations are technically realized as joins
- Associations are easier to read, just like before in the UML class diagram
  - Cardinalities
  - Paths as information about the data model and data origin
  - Filters as an alternative to WHERE or CASE
- Performance
  - Evaluation of on-demand ("JOIN on demand") of exposed associations

# Association Syntax



# Completion of the Association

Name: \* ZI\_TRN\_FLIGHPLAN  
Description: \* Flugplan  
Original Language: EN  
Content Assist Available (Ctrl+Space)  
Referenced Object: | Browse...

Efficient completion  
with CTRL + Space  
Bar

New Data Definition

Data Definition  
Create a data definition

Project: \* EA3\_010\_u910283\_en Browse...  
Package: \* STMP Browse...  
 Add to favorite packages

Name: \* ZI\_TRN\_FLIGHTPLAN  
Description: \* Flugplan  
Original Language: EN  
Referenced Object: ZI\_TRN\_CARRIER Browse...

ZI\_TRN\_CARRIER - Data Definition

one result

< Back Next > Finish Cancel

# Associations

Association

JOIN Condition

The cardinality [x..y] defines, how many target rows are possible per source row.

Association  
Naming convention: \_<Name>

```
1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Expression'
4 @Metadata.ignorePropagatedAnnotations: false
5 @ObjectModel.usageType: {
6   serviceQuality: #X,
7   sizeCategory: #S,
8   dataClass: #MIXED
9 }
10
11 @VDM.viewType: #COMPOSITE
12
13 define view entity ZI_RS20250711_BOOK ASSO
14 as select from ZI_RS20250711_BOOKING
15 association [0..1] to ZI_RS20250711_CUSTOMER as _Customer on $projection.Customerid = _Customer.Id
16 {
17   key Customerid,
18   key Connid,
19   key Fldate,
20   key Bookid,
21   Customerid,
22 }
23
24 // Associations
25 _Customer
26 }
```

# Link Criteria CDS View

The screenshot shows the SAP ABAP CDS View Editor interface. The main editor displays the following code:

```
@AbapCatalog.sqlViewName: 'ZI TRN FLIGHTPLAN'  
@AbapCatalog.compiler.compareFilter: true  
@AbapCatalog.preserveKey: true  
@AccessControl.authorizationCheck: #CHECK  
@EndUserText.label: 'Flugplan'  
define view ZI TRN FLIGHTPLAN as select from ZI TRN CARRIER  
association [1] to SPFLI as SPFLI  
on $projection.element name = ... SPFLI.target element name {  
  Carrid,  
  Carrname,  
  Currcode,  
  Url,  
  __SPFLI // Make  
}
```

Below the code, there are two panels. The left panel shows the selected element 'Carrid' and its associated columns: Carrid (column), Carrname (column), Currcode (column), and Url (column). The right panel shows the selected element 'Carrid in ZI\_TRN\_FLIGHTPLAN' and its associated column: Carrid.

Press 'Shift+Enter' to insert full signature

# Handling Associations

The screenshot shows the SAP ABAP development environment with the following SQL view definition:

```
@AbapCatalog.sqlViewName: 'ZI_TRN_FLIGHTPLAN'  
@AbapCatalog.compiler.compareFilter: true  
@AbapCatalog.preserveKey: true  
@AccessControl.authorizationCheck: #CHECK  
@EndUserText.label: 'Flugplan'  
define view ZI_TRN_FLIGHTPLAN as select from ZI_TRN_CARRIER  
association [1] to spfli as _spfli  
  on $projection.Carrid = _spfli.carrid {  
    key ZI_TRN_CARRIER.Carrid,  
    _spfli.  
  }  
  _spfli
```

A context menu is open over the `_spfli.` line, listing the following columns:

- Insert all elements (template)
- airpfrom - spfli as \_spfli (column)
- airpto - spfli as \_spfli (column)
- arrtime - spfli as \_spfli (column)
- carrid - spfli as \_spfli (column)
- cityfrom - spfli as \_spfli (column)
- cityto - spfli as \_spfli (column)
- connid - spfli as \_spfli (column)
- countryfr - spfli as \_spfli (column)

Press 'Shift+Enter' to insert full signature

**Description**  
Inserts all elements including the visible associations of the data sources into the select list.

# JOIN on Demand

- If association is just exposed with no usage in the field list then the actual join of the datasource is postponed until usage of a field.

# Association -> Navigation

Release Notes [EA3] ZI\_TRN\_CARRIER [EA3] ZI\_TRN\_FLIGHTPLAN [EA3] ZI\_TRN\_FLIGHTPLAN

ZI\_TRN\_FLIGHTPLAN

Raw Data

Filter pattern 18

**List of Associations**

\_spfli -> spfli [ 1 .. \* ]

To follow the association, choose an association from the list

AB	Carrid
AA	
AR	

Release Notes [EA3] ZI\_TRN\_CARRIER [EA3] ZI\_TRN\_FLIGHTPLAN [EA3] ZI\_TRN\_FLIGHTPLAN

ZI\_TRN\_FLIGHTPLAN > \_spfli -> spfli

Raw Data

Filter pattern 2 rows retrieved - 0 ms

Show Log Max. Rows: 100

SQL Console 11 Number of Entries Add filter

AB	mandt	AB	carrid	AB	connid	AB	countryfr	AB	cityfrom	AB	airpfrom	AB	countryto	AB	cityto	AB	airpto	12	fltime	deptime	
	010		AA		0017		US		NEW YORK		JFK		US		SAN FRAN...		SFO		361	11:00:00 am	02:0
	010		AA		0064		US		SAN FRANCIS...		SFO		US		NEW YORK		JFK		321	09:00:00 am	05:2

# CDS - Data Preview - SQL Console

The screenshot displays the SAP SQL Console interface. The top toolbar includes buttons for 'Check', 'Run', and 'Max. Rows: 100'. The main editor contains the following SQL query:

```
SELECT
  ZI_TRN_FLIGHTPLAN~CARRID,
  ZI_TRN_FLIGHTPLAN~CARRNAME,
  \_SPFLI-CONNID as CONNID
FROM
  ZI_TRN_FLIGHTPLAN
```

Below the editor, the execution log shows two successful queries:

```
2021-02-18 17:15:08 783 - Query successfully executed in 1.0000000 ms (server processor)
SELECT
  ZI_TRN_FLIGHTPLAN~CARRID,
  ZI_TRN_FLIGHTPLAN~CARRNAME,
  \_SPFLI-CONNID AS CONNID
FROM
  ZI_TRN_FLIGHTPLAN
INTO TABLE @DATA(LT_RESULT)
UP TO 100 ROWS .

2021-02-18 17:14:41 124 - Query successfully executed in 4.4410000 ms (server processor)
```

The 'Raw Data' tab on the right shows the results of the query, displaying 36 rows. The columns are 'CARRID', 'CARRNAME', and 'CONNID'. The data is as follows:

#	CARRID	CARRNAME	CONNID
1	AA	American Airlines	0017
2	AA	American Airlines	0064
3	AZ	Alitalia	0555
4	AZ	Alitalia	0788
5	AZ	Alitalia	0789
6	AZ	Alitalia	0790
7	DL	Delta Airlines	0106
8	DL	Delta Airlines	1699
9	DL	Delta Airlines	1984
10	JL	Japan Airlines	0407
11	JL	Japan Airlines	0408
12	LH	Lufthansa	0400
13	LH	Lufthansa	0401
14	LH	Lufthansa	0402
15	LH	Lufthansa	2402
16	LH	Lufthansa	2407
17	QF	Qantas Airways	0005
18	QF	Qantas Airways	0006
19	SQ	Singapore Airlines	0002
20	SQ	Singapore Airlines	0015
21	SQ	Singapore Airlines	0158
22	SQ	Singapore Airlines	0988
23	UA	United Airlines	0941
24	UA	United Airlines	3504
25	UA	United Airlines	3516
26	UA	United Airlines	3517
27	UA	United Airlines	0000

# Associations Association instead of Inner Join

1 1



# Using Associations

# ABAP and Pretty Printer

New ABAP Program

ABAP Program

Create an ABAP program

Project: \* EA3\_010\_u910283\_en Browse...

Package: \* STMP Browse...

Add to favorite packages

Name: \* ZR\_TRN\_FLIGHPLAN

Description: \* Flugplan

Original Language: EN

System Program

< Back Next > Finish Cancel

Preferences

abap edit

Source Code Editors

Settings for ABAP source code editors.

See 'Text Editors' for general text editor preferences.

See 'Colors and Fonts' for general color and font preferences.

See 'Annotations' to change the appearance of editor annotations.

See 'ABAP Formatter' for project specific properties.

Properties for EA3\_010\_u910283\_en (Filtered)

type filter text

ABAP Formatter

Indentation

Indent Lines

Upper/Lower Case Conversion

None

Custom

Keywords

Upper Case

Lower Case

Identifiers

Upper Case

Lower Case

Preview

```
LOOP AT cars INTO DATA(car).
  car->drive( ).
ENDLLOOP.
```

Apply

Apply and Close Cancel

# CDS - ABAP Usage Exposed Association

```

@AbapCatalog.sqlViewName: 'ZI TRN FLIGHTPLAN'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Flugplan'
define view ZI TRN FLIGHTPLAN as select from ZI TRN CARRIER
association [1] to SPFLI as SPFLI
on $projection.element name = SPFLI.target element name
Carrid,
Carrname,
Currcode,
Url,
__SPFLI // Make
    
```

```

*&-----*
*& Report zr_trn_flightplan
*&-----*
*& Verwendung von CDS Views im ABAP
*&-----*
REPORT zr_trn_flightplan.

DATA: BEGIN OF gs_flightplan,
       carrid TYPE s_carr_id,
       connid TYPE s_conn_id,
END OF gs_flightplan,
gt_flightplan2 LIKE TABLE OF gs_flightplan.

START-OF-SELECTION.

* Simple SELECT
SELECT * FROM zi_trn_flightplan INTO TABLE @DATA(gt_flightplan1).

* Projection and Selection
SELECT
  carrid,
  \_spfli-connid
FROM zi_trn_flightplan
where carrid = 'LH'
and \_spfli-connid = '0400'
INTO TABLE @gt_flightplan2.

* Ausgabe
cl_demo_output=>display( gt_flightplan2 ).
    
```

In ABAP "\" for all association names

If an alias is used: a~\\_spfli-connid

Im ABAP ist der Komponentenselektor "-" in der CDS-DDL ist es "."

# Filter in Associations

```

1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Currency'
4 @Metadata.ignorePropagatedAnnotations: true
5 @ObjectModel.usageType: {
6   serviceQuality: #X,
7   sizeCategory: #S,
8   dataClass: #MIXED
9 }
10 define view entity ZI_RS20250917_SCARR_CURR
11   with parameters
12     p_langu : syst_langu
13   as select from scarr
14   association [0..*] to tcurt as _Currency
15   on $projection.Currcode = _Currency.waers
16 {
17   key carrid
18     carrname
19     currcode
20     url
21     Currency[spras = $parameters.p_langu].ktext as CurrText
22 }

```

Data Preview

find pattern 18 rows retrieved - 46 ms

AB	Carrid	AB	Carrname	AB	Currcode	AB	Url	AB	CurrText
	AA		American Airlines		USD		http://w...		US Dollar
	AC		Air Canada		CAD		http://w...		Kan.Dollar
	AF		Air France		EUR		http://w...		Euro
	AZ		Alitalia		EUR		http://w...		Euro
	BA		British Airways		GBP		http://w...		Pfund

1 SELECT  
2 ZI\_RS20250917\_SCARR\_CURR~CARRID,  
3 ZI\_RS20250917\_SCARR\_CURR~CARRNAME,  
4 ZI\_RS20250917\_SCARR\_CURR~CURRCODE,  
5 ZI\_RS20250917\_SCARR\_CURR~URL,  
6 ZI\_RS20250917\_SCARR\_CURR~CURRTEXT  
7 FROM  
8 ZI\_RS20250917\_SCARR\_CURR ( P\_LANGU = 'D' )

2025-09-18 14:14:29 090 - Query successfully executed in 38.38900000 ms (server processing time)

SELECT  
ZI\_RS20250917\_SCARR\_CURR~CARRID,  
ZI\_RS20250917\_SCARR\_CURR~CARRNAME,  
ZI\_RS20250917\_SCARR\_CURR~CURRCODE,  
ZI\_RS20250917\_SCARR\_CURR~URL,  
ZI\_RS20250917\_SCARR\_CURR~CURRTEXT  
FROM  
ZI\_RS20250917\_SCARR\_CURR ( P\_LANGU = 'D' )  
INTO TABLE @DATA(LT\_RESULT)  
UP TO 100 ROWS.

Data Preview

find pattern 18 rows ... - 38 ms

AB	CARRID	AB	CARRNAME	AB	CURRCODE	AB	URL	AB	CURRTEXT
	AA		American Airlines		USD		http://w...		US Dollar
	AC		Air Canada		CAD		http://w...		Kan.Dollar
	AF		Air France		EUR		http://w...		Euro
	AZ		Alitalia		EUR		http://w...		Euro
	BA		British Airways		GBP		http://w...		Pfund
	FJ		Air Pacific		USD		http://w...		US Dollar
	CO		Continental Airlines		USD		http://w...		US Dollar
	DL		Delta Airlines		USD		http://w...		US Dollar
	AB		Air Berlin		EUR		http://w...		Euro
	LH		Lufthansa		EUR		http://w...		Euro
	NG		Lauda Air		EUR		http://w...		Euro
	JL		Japan Airlines		JPY		http://w...		Yen
	NW		Northwest Airlines		USD		http://w...		US Dollar
	QF		Qantas Airways		AUD		http://w...		Austr.Dollar
	SA		South African Air.		ZAR		http://w...		Rand
	SQ		Singapore Airlines		SGD		http://w...		Sing.Dollar
	SR		Swiss		CHF		http://w...		Schweiz.Franken
	UA		United Airlines		USD		http://w...		US Dollar

# ABAP

- Using DDL Source in ABAP (Link Postings, Customers)
- Write a small program
- Activate, execute



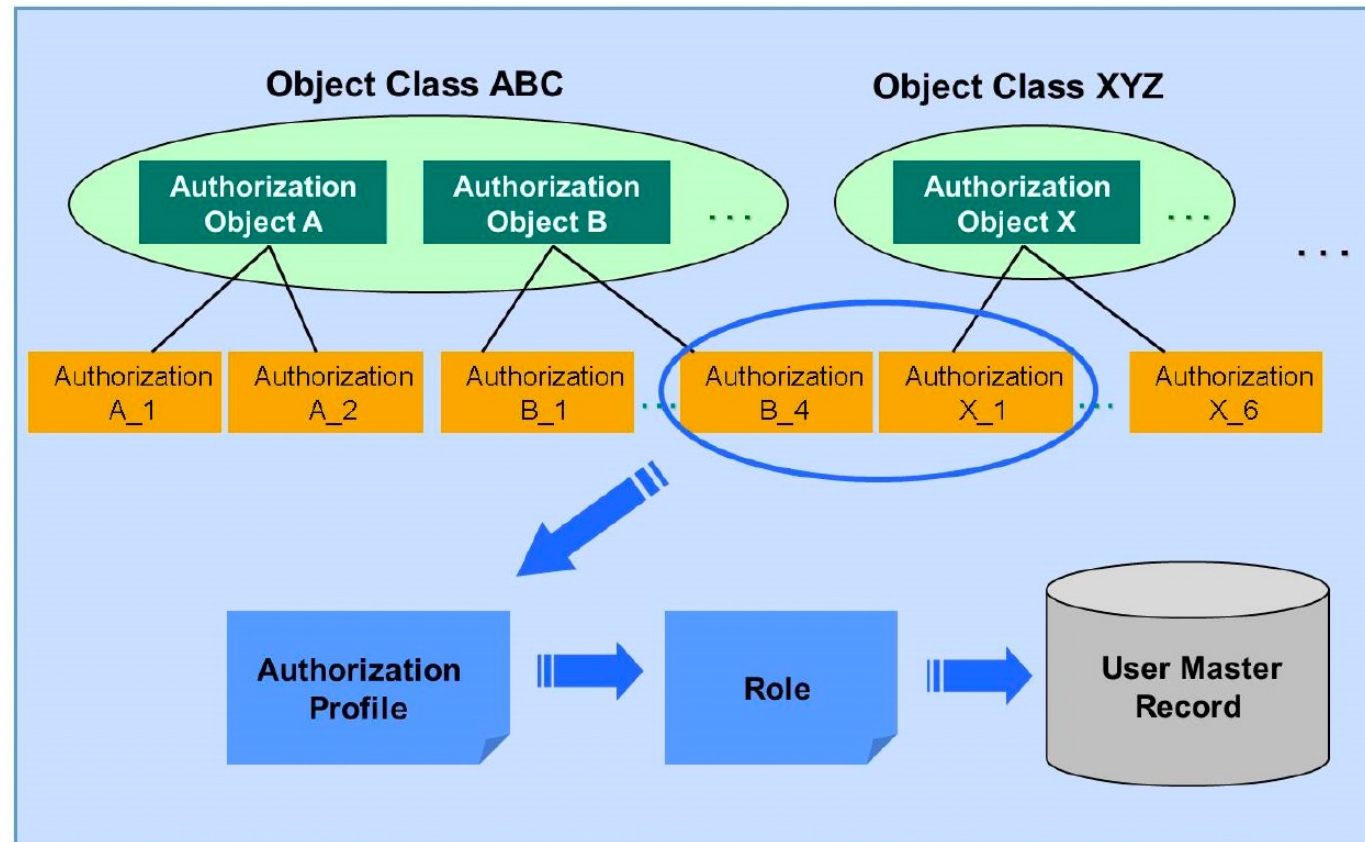
# 12

# Access Control

# Access control (Classical)

Authorization Object = SU21  
Roles: PFCG  
User: SU01

Check Auth User: SU53  
User (Fiori): STAUTHTRACE



# Auth Checks – Who is checking?

- Implicit Authorization Checks
  - Performed by the ABAP Framework
  - Start of a transaction
  - Start of a Web Dynpro application
  - Start of a report
  - Call of an RFC function module
  - Access to table content with data browser or maintenance dialogue
- Explicit Authorization Checks
  - Defined in the ABAP Coding (AUTHORITY-CHECK statement)
  - Access to specific functionality
  - Access to specific data

# Known Problems

- General Problems:
  - No visible link between data and authorization objects
  - Developer can forget (or ignore) necessary checks
  - Users with debugging rights can bypass authorization checks
- Problems related to Code-to-Data:
  - All authorization checks on application level
  - Some calculations cannot be done on the database because they require an authorization check between steps

# Access control (CDS)

```

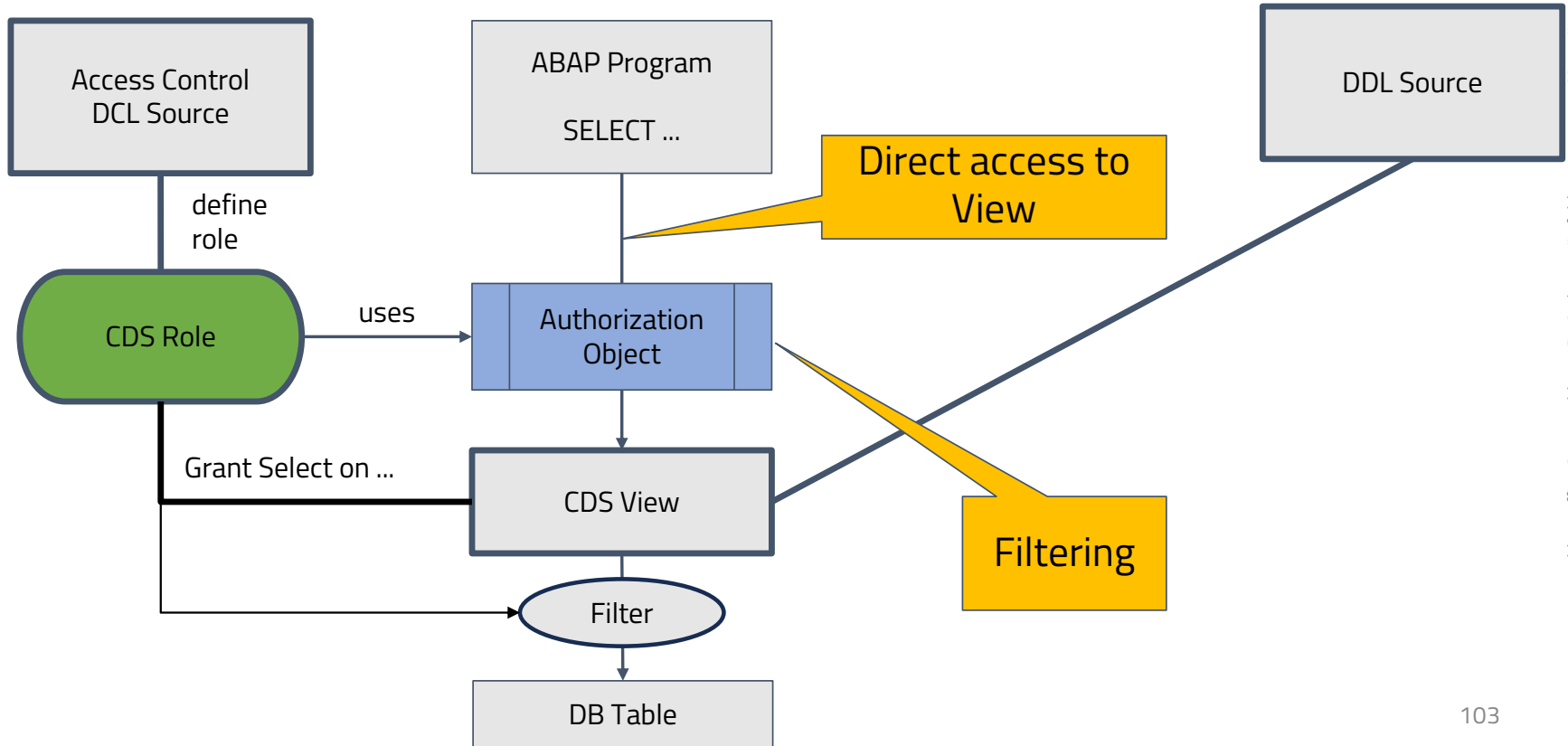
@EndUserText.label: 'Zugriffskontrolle Fluggesellschaft'
@MappingRole: true
define role ZI_TRN_CARRIER_AC {
  grant select on ZI_TRN_CARRIER

```

```

@AbapCatalog.sqlViewName: 'ZI_TRN_CAR'
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Carrier'
define view ZI_TRN_CARRIER as select from scarr
{

```



# Example Code – AUTHORITY-CHECK

```
DATA gt_connections TYPE TABLE OF ZI_CONN.  
DATA gs_connection LIKE LINE OF gt_connections.  
  
PARAMETERS pa_citfr TYPE s_from_cit DEFAULT 'FRANKFURT'.  
  
SELECT FROM ZI_CONN  
  FIELDS carrid, connid, cityfrom, cityto  
  WHERE cityfrom = @pa_citfr  
  INTO TABLE @gt_connections.  
  
LOOP AT gt_connections INTO gs_connection.  
  
  AUTHORITY-CHECK OBJECT 'S_CARRID'  
    ID 'CARRID' FIELD gs_connection-carrid  
    ID 'ACTVT' FIELD '03'.  
  IF sy-subrc <> 0.  
    DELETE gt_connections INDEX sy-tabix.  
  ENDIF.  
  
ENDLOOP.
```

# Create Access Control

The screenshot displays the SAP IDEAS interface. On the left, a project tree shows the following structure:

- Connectivity
- Core Data Services
  - D
  - Dicti
  - Database Tables

The 'New' option is selected under 'D'. A context menu is open, showing 'Access Control' and 'Data Definition' as options.

The 'New Access Control' dialog box is open, showing the following fields:

- Project: \* EA3\_010\_u910283\_en
- Package: \* \$TMP
- Add to favorite packages
- Name: \* ZI\_TRN\_CARRIER\_AC
- Description: \* Zugriffskontrolle Fluggesellschaft
- Original Language: EN
- Protected Entity: ZI\_TRN\_CARRIER

Buttons at the bottom of the dialog include: ? < Back Next > Finish Cancel

# Create Access Control

New Access Control

**Templates**  
Select one of the available templates.

Use the selected template

- Define Role with Simple Conditions
- Define Role with PFCG Aspect**
- Define Role with Inherited Conditions
- Define Role with Generic Aspect
- Define Generic Aspect
- Define Role with Unrestricted Access

Defines a role that grants instance-specific access to a CDS entity based on authorizations derived from PFCG roles assigned to the current user.

```
@EndUserText.label: '${dcl_source_description}'
@MappingRole: true
define role ${dcl_source_name} {
  grant
  select
  on
  ${cds_entity}
  where
    (${entity_element_1}, ${entity_element_2}) = aspect pfog
    ${cursor}
}
```

< Back Next > Finish Cancel

```
Release Notes [EA3] ZI_TRN... [EA3] ZI_TRN... [EA3] ZR_TRN... [EA3] EA3_01...
@EndUserText.label: 'Zugriffskontrolle Fluggesellschaft'
@MappingRole: true
define role ZI_TRN_CARRIER_AC {
  grant select on ZI_TRN_CARRIER
  where (carrid) = aspect pfog_auth(s_carrid, carrid, actvt = '03');
}
```

View Field

Authorization Object and Fields

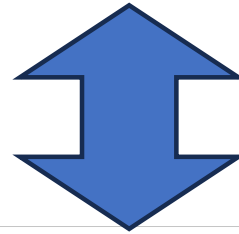
Infos (1 item)					
The entity ZI_TRN_CARRIER is used in DCL document ZI_TRN_CARRIER_AC [Access Control Management]		zi_trn_carrier....	/EA3_010_u910283_...	Unknown	ABAP Syntax ...

# Example

```
@EndUserText.label: 'Demo: Authorization Check With Link to User Profile'  
@MappingRole: true  
define role ZA_CONN  
{  
  grant select  
  on ZI_CONN  
  where (carrid) = aspect pfcg_auth( S_CARRID,  
    CARRID,  
    ACTVT = '03' )  
  
  and (carrid, counter) = aspect pfcg_auth( S_COUNTER,  
    CARRID,  
    COUNTNUM,  
    ACTVT = '03' );  
}
```

# Mapping between classical and CDS Access Control

```
where (carrid) = aspect pfcg_auth( S_CARRID,  
                                CARRID,  
                                ACTVT = '03'  
                                )
```



```
AUTHORITY-CHECK OBJECT 'S_CARRID'  
  ID 'CARRID' FIELD view-carrid  
  ID 'ACTVT' FIELD '03'.
```

# Annotation in Access Control

@EndUserText.label:

- Translatable short text for role (same as in Data Definition)

@MappingRole:

- Value true: Role is implicitly assigned to all users
- Value false: Currently not supported by ABAP CDS

# Annotation in Data Definition

@AccessControl.authorizationCheck:

#CHECK

Performs authorization check and provides warning if no role is assigned

#NOT\_REQUIRED

Performs permission check and does not provide a warning if no role is assigned

#NOT\_ALLOWED

Does not perform an authorization check and provides a warning if a role is assigned

# Access Control

- Provide DDL Source (Link Bookings, Customers)
  - ZZI\_xx\_SBOOK\_ASSO -> ZZI\_xx\_SBOOK\_AUTH
- Define CDS Role
  - Name DCL Source and CDS Role Same
  - Template DefineMappingRole
  - Check CARRID
- Activate, view data



# 13

# Buffering CDS View Entities

# Buffering CDS View Entities

- A CDS entity buffer temporarily caches the data retrieved from a CDS view entity into the shared memory of the current AS ABAP instance.

Starting with the following ABAP releases, **buffering** of CDS view entities is supported:

SAP BTP, ABAP Environment 2202

- ABAP platform cloud 2202

# Buffering CDS View Entities

## Preparation

- **Prepare CDS View Entity for Buffer Definitions**  
@AbapCatalog.entityBuffer.definitionAllowed: true
- The annotation itself does not lead to a buffer, but ensures the following restrictions apply:
  - Views with parameters are not supported
  - Views with unstable calculations are not supported (e.g. utcl\_current, tstmp\_current\_utctimestamp)
  - Session variables (other than client) can not be used
  - Base objects have to be tables
  - Key length > 900 bytes is not supported
  - Views without key fields can not be buffered
  - Data aging in base tables is not allowed

# Buffering CDS View Entities

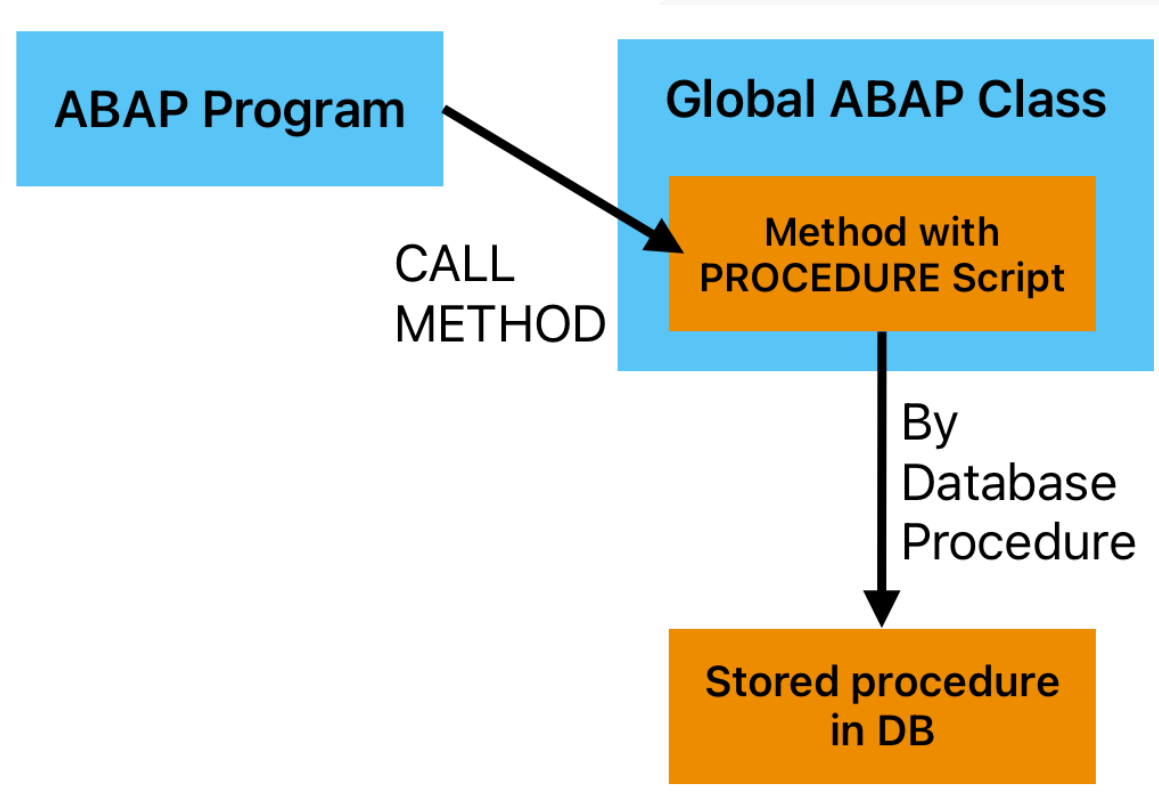
The image illustrates the steps to create a CDS view entity buffer in SAP Studio:

- 1**: Right-click on the CDS view in the project tree and select **Entity Buffer** from the context menu.
- 2**: In the **New Entity Buffer** dialog, specify the **Project**, **Package**, and **Name** of the new entity buffer.
- 3**: In the **New Entity Buffer** dialog, select a template from the **Templates** section.
- 4**: In the code editor, define the view entity buffer with the following code:

```
1 define view entity buffer on ZI_RS20250917_SCARR_2  
2   layer customer  
3   type full  
4
```

# AMDP Procedures & Table Functions

# AMDP



# AMDP Code

## **Class Definition**

- Implement the interface IF\_AMDP\_MARKER\_HDB

## **Method Definition**

- Add VALUE( ) for all parameters
- Only scalar types and table types allowed for parameters (no structures)

## **Supported Databases and Languages**

- Only SAP HANA (currently)
- Only SAP HANA SQLScript (currently)

## **Use of Dictionary Objects**

- Transparent tables and dictionary views
- Till 7.50: CDS views only via their CDS-SQL views
- Must be listed using the USING clause

# AMDP Example – Fuzzy Search

```
1 CLASS zcl_rs20250917_fuzzy DEFINITION
2 PUBLIC
3 FINAL
4 CREATE PUBLIC .
5
6 PUBLIC SECTION.
7     types: td_score type p length 3 DECIMALS 2.
8     types: begin of mts_carrier,
9             score type td_score,
10            carrid type scarr-carrid,
11            carrname type scarr-carrname,
12            end of mts_carrier,
13     mtt_carriers type standard table of mts_carrier.
14
15 INTERFACES if_amdp_marker_hdb.
16
17 class-methods: carrier_fuzzy
18                 importing value(iv_name) type scarr-carrname
19                 exporting value(et_carriers) type mtt_carriers.
20
21 PROTECTED SECTION.
22 PRIVATE SECTION.
23 ENDClass.
24
25 CLASS zcl_rs20250917_fuzzy IMPLEMENTATION.
26 METHOD carrier_fuzzy BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT using Scarr.
27     et_carriers = select score( ) as score,carrid, carrname from scarr
28                   where CONTAINS (carrname, :iv_name, fuzzy( 0.8 ))
29                   and mandt = session_context( 'CLIENT' ) ORDER BY score( );
30 ENDMETHOD.
31
32 ENDClass.
```

TestObject->CARRIER\_FUZZY()

Case-Sensitive

Runtime: 201.514 Microseconds

CARRIER\_FUZZY

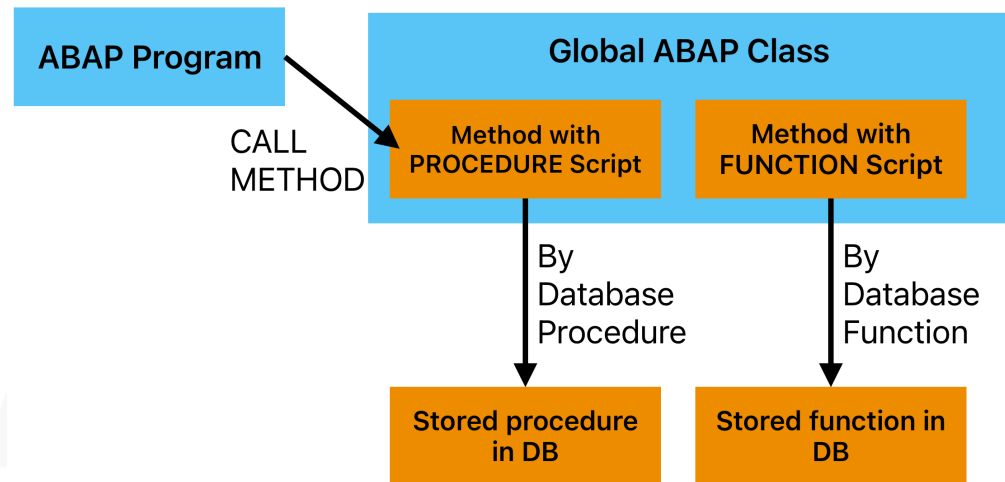
- Import Parameter
  - IM\_CARRNAME LUFTANSA
- Export Parameter
  - EX\_DETAILS 1 Entry

1 Entry

SCORE	CAR	CARRNAME
0,91	LH	Lufthansa

order by score()

# AMDP Function



Unlike the implementation of **AMDP procedures**, the implementations of **AMDP functions** do not have export or changing parameters. They have exactly one table-like return value.

AMDP function implementations cannot be called directly from an ABAP program. However, they can be called from other AMDP functions or procedures.

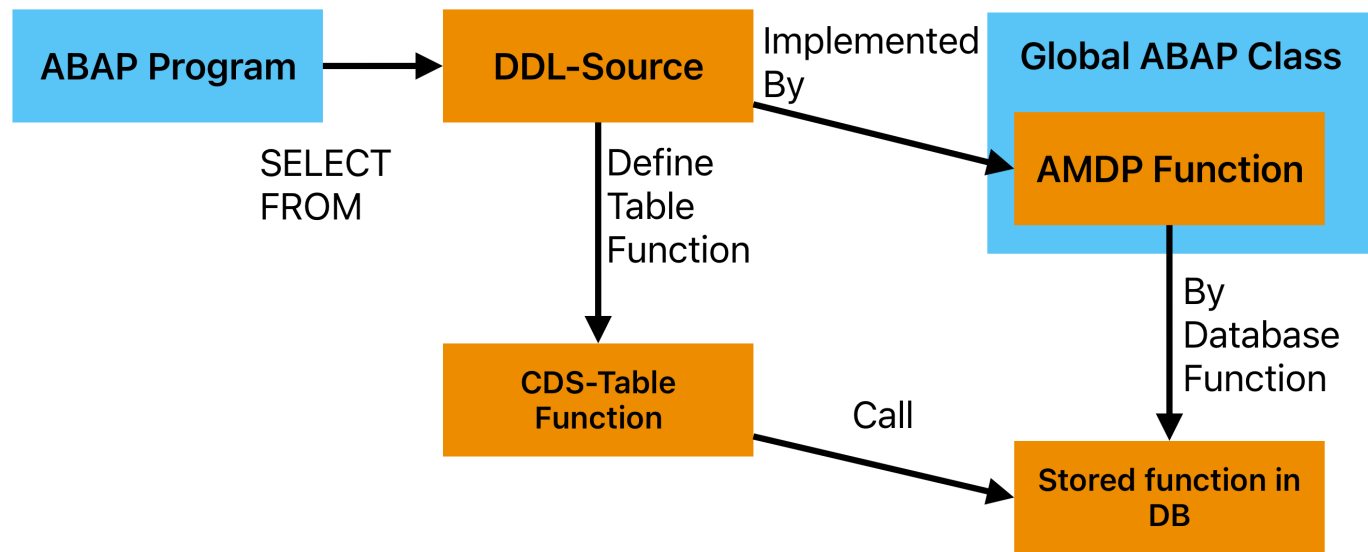
# AMDP Function

```
CLASS zcl_rs20250716_tbl_fun_fuzzy DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC.

  PUBLIC SECTION.
    INTERFACES if_amdp_marker_hdb.
    CLASS-METHODS: customer_fuzzy FOR TABLE FUNCTION
zi_rs20250716_tbl_fun_fuzzy.
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS zcl_rs20250716_tbl_fun_fuzzy IMPLEMENTATION.
  METHOD customer_fuzzy BY DATABASE FUNCTION FOR HDB LANGUAGE SQLSCRIPT
  OPTIONS READ-ONLY USING scustom.
    return select mandt, id, name, city, country from scustom where
contains( name, :name_in, fuzzy( 0.8 ) )| order by score( ) desc;
  ENDMETHOD.
ENDCLASS.
```

# CDS-Table Function



# CDS-Table Functions

New Data Definition

Templates  
Select one of the available templates.

Use the selected template

Name	Description
Table Function (creation)	
<b>defineTableFunctionWithParameters</b>	<b>Define Table Function with Parameters</b>
Abstract Entity (creation)	
Abstract Entity Extend (creation)	
Hierarchy (creation)	
Custom Entity (creation)	
Custom Entity Extend (creation)	
External Entity (creation)	

```
@EndUserText.label: '${ddl_source_description}'
define table function ${ddl_source_name_editable}
with parameters ${parameter_name} : ${parameter_type}
returns {
  ${client_element_name} : abap.clnt;
  ${element_name} : ${element_type};
  ${cursor}
}
implemented by method ${class name}=>${method name};
```

```
1 @EndUserText.label: 'Fuzzy'
2 define table function ZI_RS20250716_TBL_FUN_FUZZY
3   with parameters
4     name_in : s_custname
5 returns
6 {
7   mandt    : abap.clnt;
8   id       : s_customer;
9   name     : s_custname;
10  city     : s_city;
11  country  : s_country;
12 }
13 }
14 implemented by method
15   ZCL_RS20250716_TBL_FUN_FUZZY=>customer_fuzzy;
```

```
1=CLASS zcl_rs20250716_tbl_fun_fuzzy DEFINITION
2 PUBLIC
3 FINAL
4 CREATE PUBLIC .
5
6 PUBLIC SECTION.
7   INTERFACES if_amdp_marker_hdb .
8   CLASS-METHODS: customer_fuzzy FOR TABLE FUNCTION zi_rs20250716_tbl_fun_fuzzy.
9 PROTECTED SECTION.
10 PRIVATE SECTION.
11 ENDClass.
12
13=CLASS zcl_rs20250716_tbl_fun_fuzzy IMPLEMENTATION.
14= METHOD customer_fuzzy
15 BY DATABASE FUNCTION
16 FOR HDB
17 LANGUAGE SQLSCRIPT
18 OPTIONS READ-ONLY
19 USING scustom.
20   RETURN SELECT mandt, id, name, city, country
21   FROM scustom
22   WHERE contains( name,:name_in,fuzzy( 0.8 ) )
23   order by score( ) desc;
24 ENDMETHOD.
25 ENDClass.
```

# CDS-Table Functions in use

## Use CDS-Table Function

```
* Data Retrieval
*****
SELECT FROM zI_rs20250716_tbl_fun_fuzzy( name_in = @pa_nam )
FIELDS id, name, city, country ORDER BY country, city, name
INTO TABLE @gt_customers.
```

## Check feature availability

```
-----
INSERT cl_abap_dbfeatures=>amdp_table_function INTO TABLE gt_feature_set.
⊖ IF cl_abap_dbfeatures=>use_features( gt_feature_set ) = abap_false.
    MESSAGE 'Fehler' TYPE 'E'.
    ENDIF.
⊖ * Alternative with VALUE expression (Rel. 7.40)
*IF cl_abap_dbfeatures=>use_features(
* VALUE #( ( cl_abap_dbfeatures=>amdp_table_function ) )
* ) = abap_false.
* MESSAGE e001(Z<Initials><ANSI Date>).
*ENDIF.
```

# Eclipse Debugger for DB Procedures / Functions

```
13 CLASS zcl_rs20250716_tbl_fun_fuzzy IMPLEMENTATION.  
14 METHOD customer_fuzzy  
15 BY DATABASE FUNCTION  
16 FOR HDB  
17 LANGUAGE SQLSCRIPT  
18 OPTIONS READ-ONLY  
19 USING scustom.  
20 RETURN SELECT mandt, id, name, city, country  
21 FROM scustom  
22 WHERE contains( name, :name_in, fuzzy(  
23 order by score( ) desc;  
24 ENDMETHOD.  
25 ENDClass.
```

The screenshot shows the Eclipse IDE interface with the following components:

- Project Explorer:** Shows the project structure with a red box highlighting the class `Class ZCL_RS20250716_TBL_FUN_FUZZY (Line 20)` and the procedure `Procedure CUSTOMER_FUZZY (Class ZCL_RS20250716_TBL...`.
- Main Editor:** Displays the SQL script with line 20 highlighted in green. A red box highlights the `RETURN SELECT` statement.
- Variables View:** Shows the current state of variables:

Name	Value	Type
::CURRENT_OBJECT_NAME	ZCL_RS20250...	NVARCHAR(256)
::CURRENT_OBJECT_SCHEMA	SAPHANADB	NVARCHAR(256)
::ROWCOUNT	0	BIGINT
NAME_IN	King	NVARCHAR(25)
_SYS_SS2_RETURN_VAR_	TABLE[0]	TABLE

# AMDP and CDS - Fuzzy, Linguistic Search

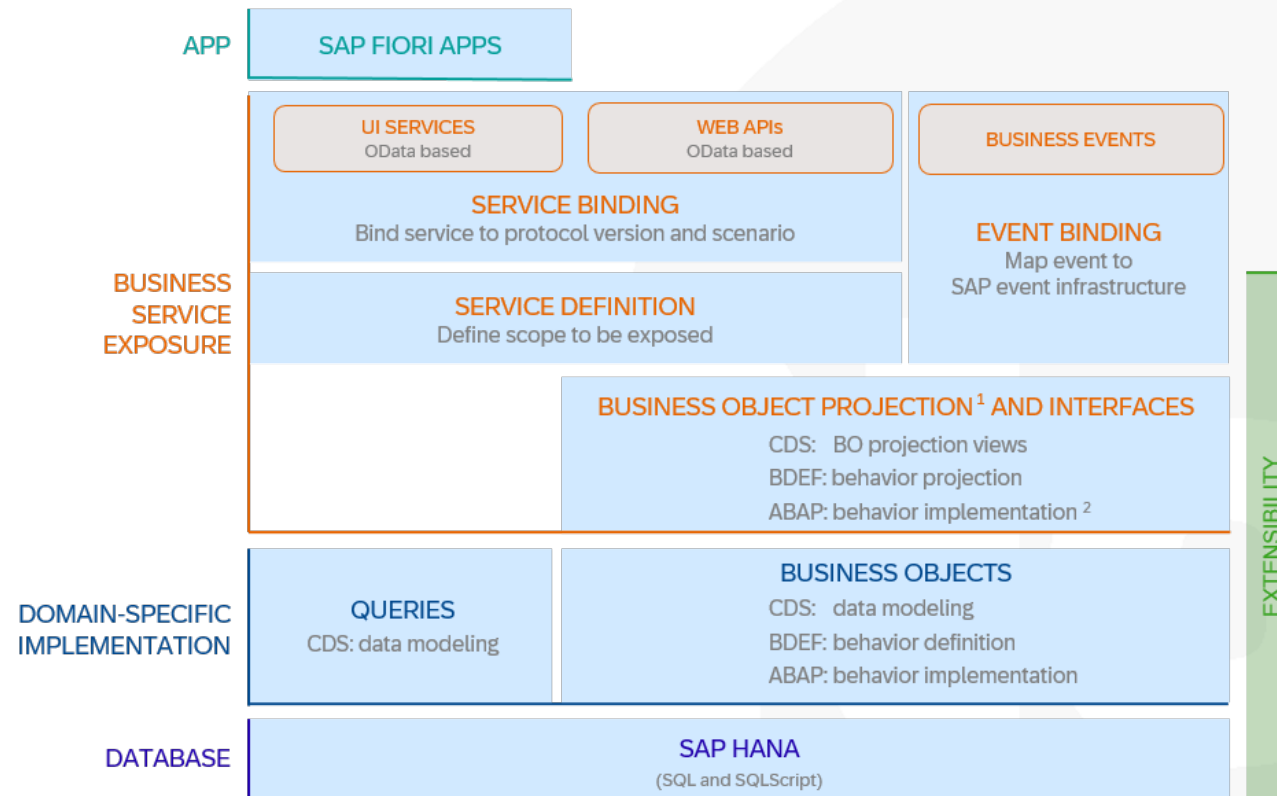


**1014**



RAP in 20 mins

# RAP Big Picture

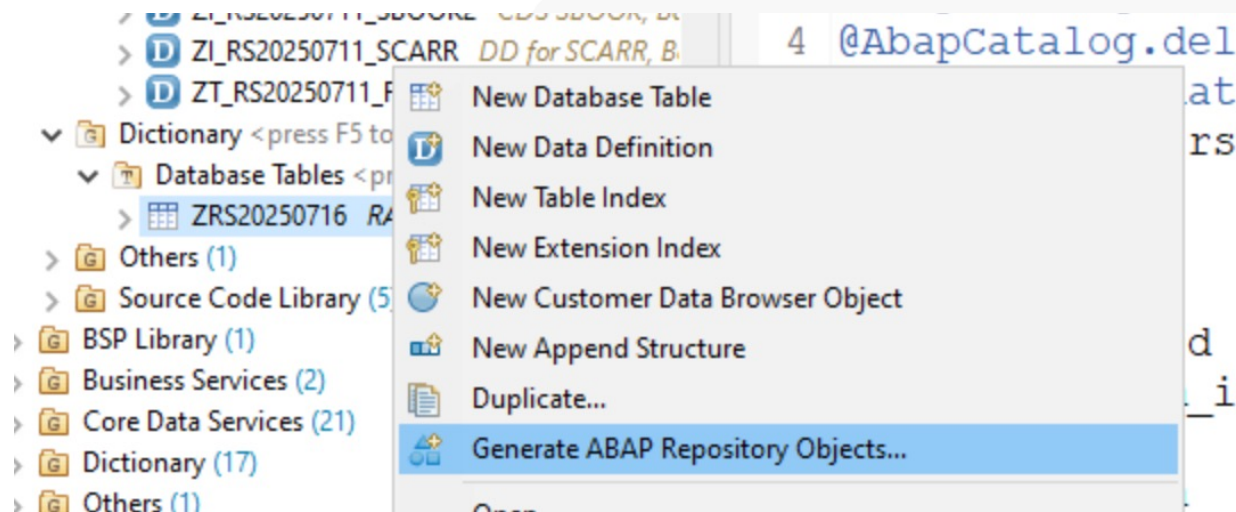


<sup>1</sup> Also called Service Projection    <sup>2</sup> Not applicable for RAP BO interfaces

# Database Table

```
1 @EndUserText.label : 'RAP Table'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 @AbapCatalog.tableCategory : #TRANSPARENT
4 @AbapCatalog.deliveryClass : #A
5 @AbapCatalog.dataMaintenance : #RESTRICTED
6 define table zrs20250716 {
7   key client           : abap.clnt not null;
8   key uuid             : sysuuid_x16 not null;
9   carrier_id           : /dmo/carrier_id;
10  connection_id        : /dmo/connection_id;
11  airport_from_id      : /dmo/airport_from_id;
12  city_from            : city;
13  country_from         : land1;
14  airport_to_id        : /dmo/airport_to_id;
15  city_to              : city;
16  country_to           : land1;
17  local_created_by     : abp_creation_user;
18  local_created_at     : abp_creation_tstmpl;
19  local_last_changed_by : abp_locinst_lastchange_user;
20  local_last_changed_at : abp_locinst_lastchange_tstmpl;
21  last_changed_at      : abp_lastchange_tstmpl;
22
23 }
```

# Generate ABAP Repository Objects



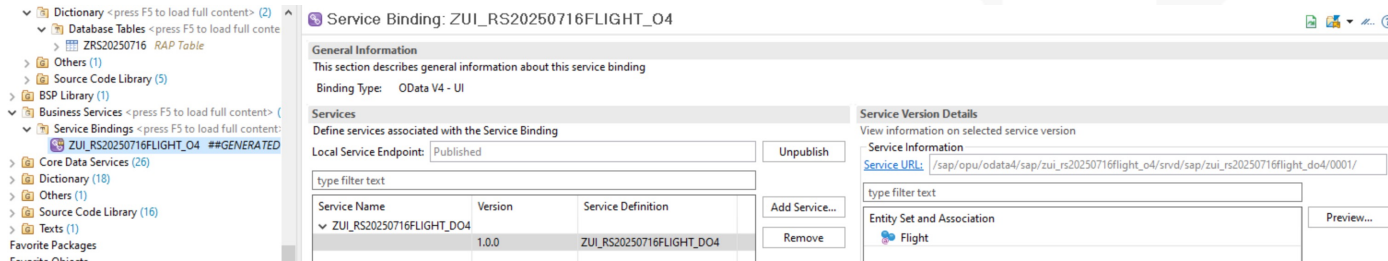
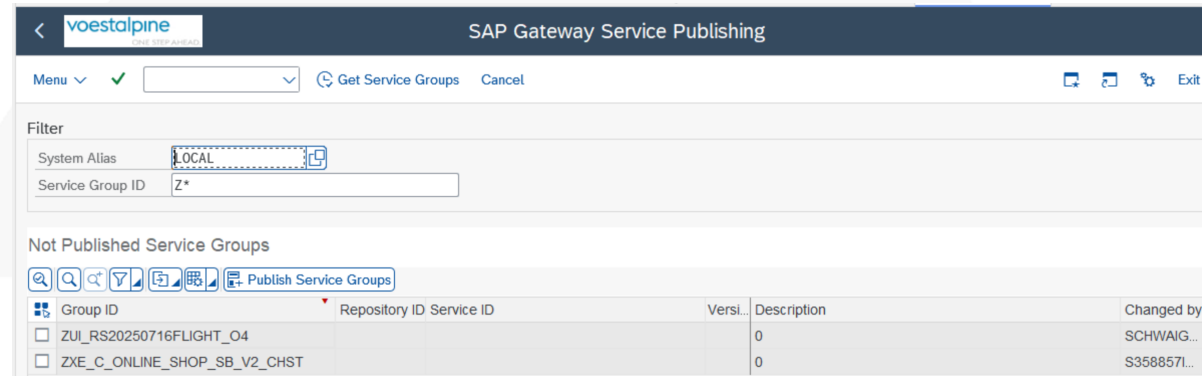
# Data Generator

## Naming suggestions

Field	Value
<i>General → Referenced Object</i>	Z##FLIGHT
<i>General → Project Name</i>	<empty>
<i>General → Artifacts Prefix</i>	<empty>
<i>General → Artifacts Suffix</i>	<empty>
<i>Business Object → Data Model → CDS Entity Name</i>	ZR_##Flight
<i>Business Object → Data Model → CDS Entity Alias</i>	Flight
<i>Business Object → Behavior → Behavior Implementation Class</i>	ZBP_R_##FLIGHT
<i>Business Object → Behavior → Draft Table Name</i>	Z##FLIGHT_D
<i>Service Projection → Service Projection Entity → CDS Entity Name</i>	ZC_##Flight
<i>Service Projection → Service Projection Behavior → Behavior Implementation Class</i>	ZBP_C_##FLIGHT
<i>Business Service → Service Definition → Service Definition Name</i>	ZUI_##FLIGHT_04
<i>Business Service → Service Binding → Service Binding Name</i>	ZUI_##FLIGHT_04
<i>Business Service → Service Binding → Binding Type</i>	0Data V4 - UI

# Register O4 Service

- /IWFND/V4\_ADMIN (Transaction)
- Publish Service Groups
- Search Service Group: LOCAL, Z\*
- Publish service group
- Refresh



# Preview App

Standard ▼ 🔗

Editing Status:  
All ▼

Go Adapt Filters (1)  
Enter

Flights Create Delete ⚙️ 📄 ▼

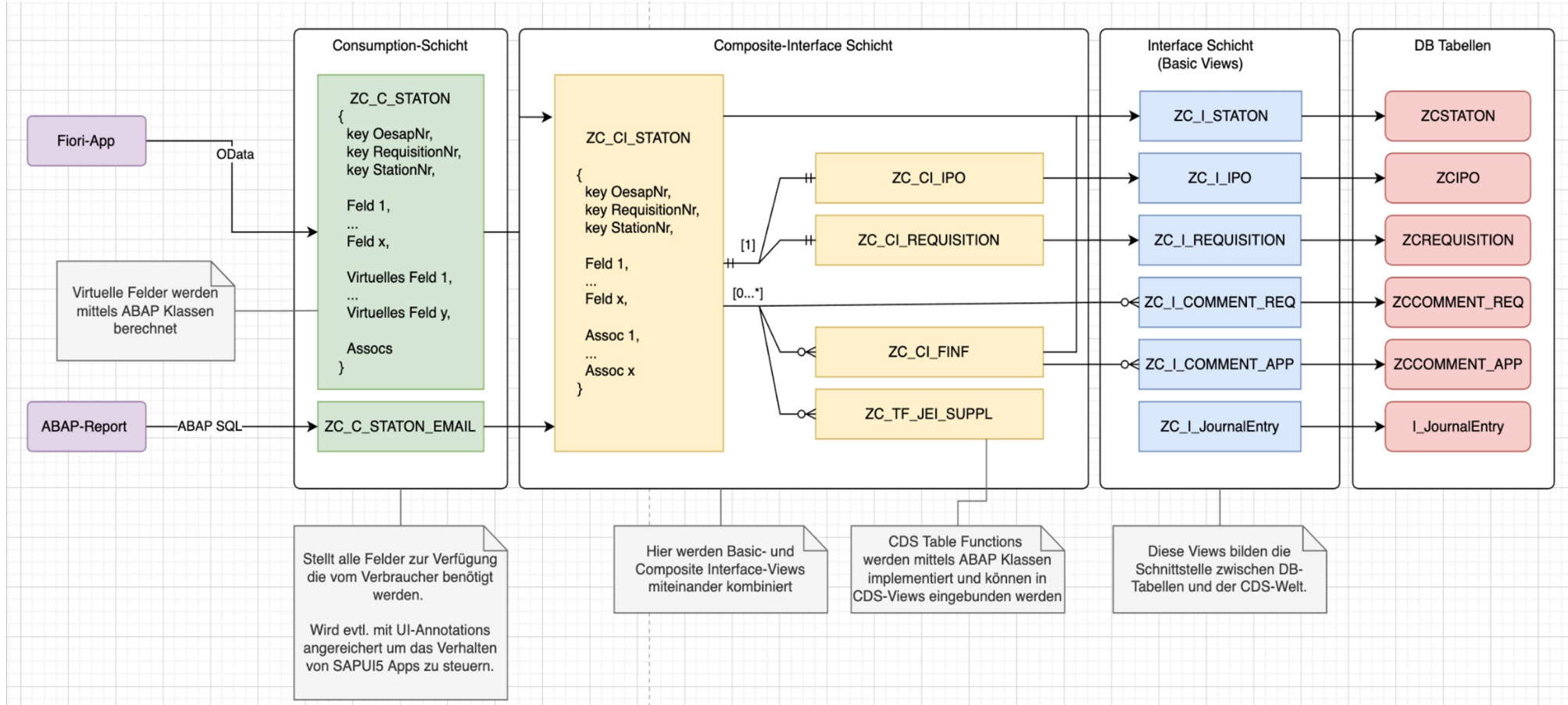
🔗	Flight Referen...	Flight Num...	Departure ...	City	Country/Re...	Destination ...	City	Country/Re...
No data found. Try adjusting the search or filter parameters.								

# RAP

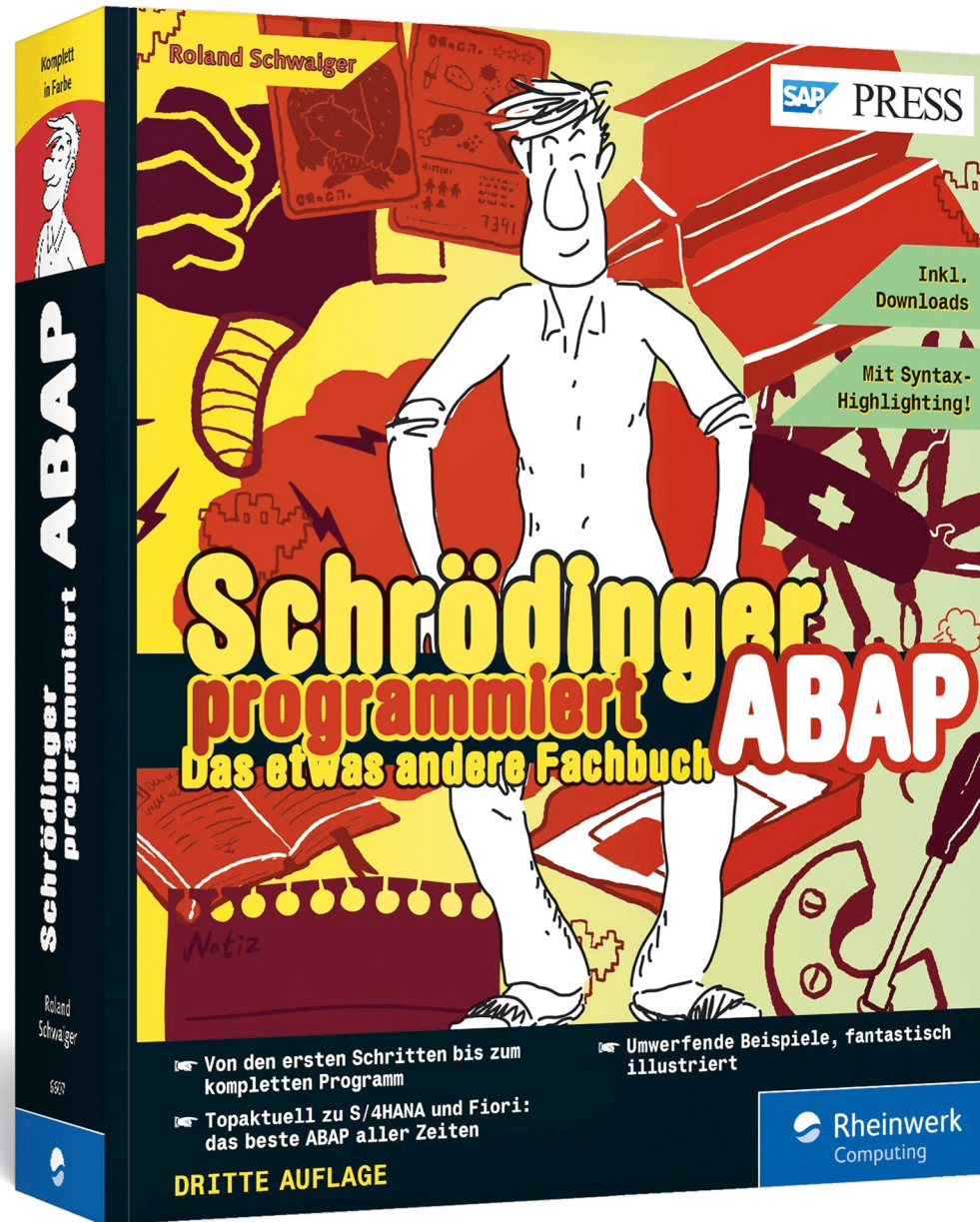
- Create RAP



# Real World Example



DONE!



# Anhang