ZS4HCDS
S/4HANA Development Fundamentals

SAP Development Learning

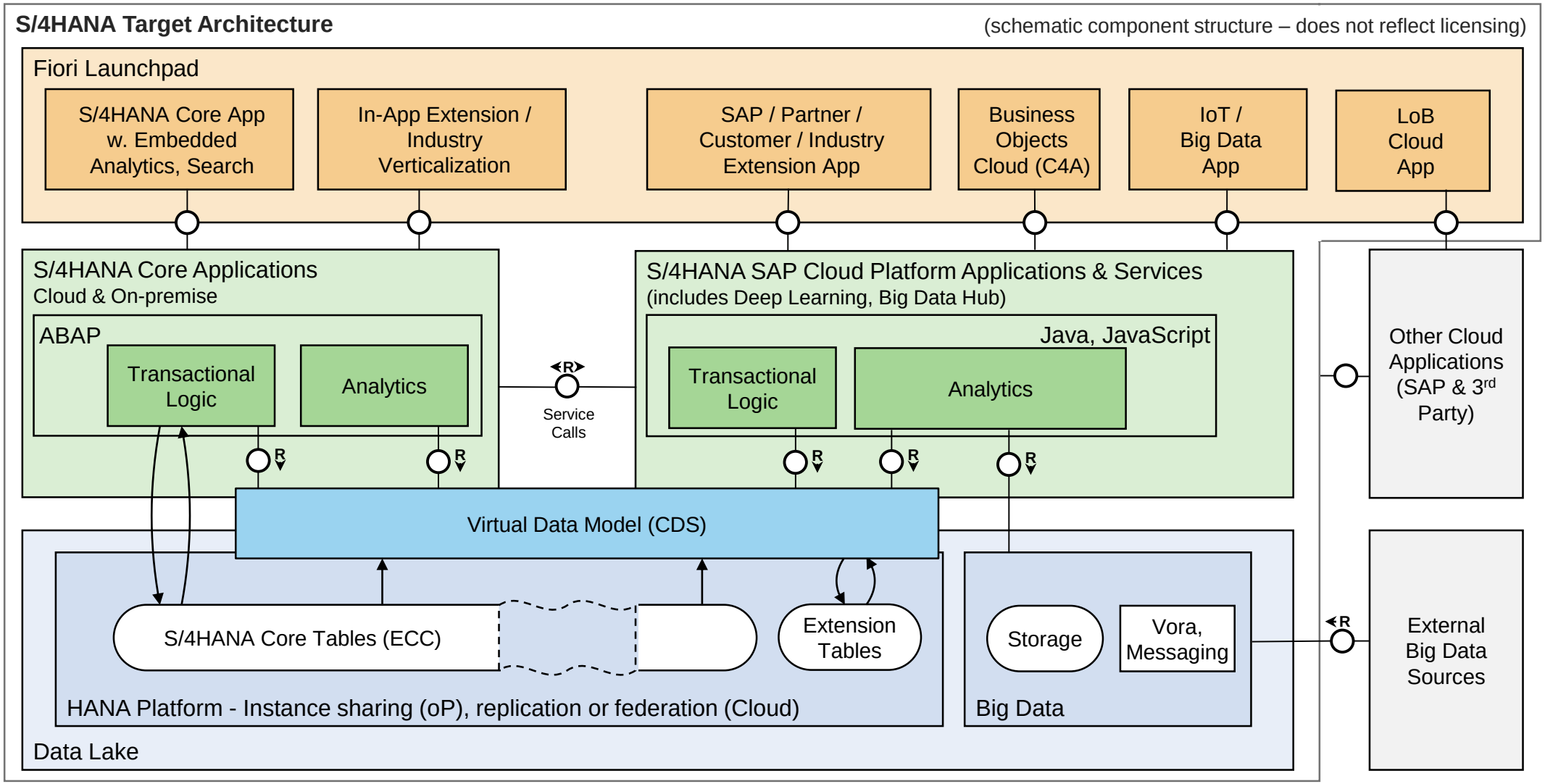
Unit 1 – Introduction to Core Data Services

Unit 1 – Introduction to Core Data Services

Lesson 1 – Motivation and Definition

SAP S/4HANA Target Architecture

SAP Development Learning
Simply Brilliant



Develop SAP HANA optimized SAP Fiori apps

The new programming model

SAP Development Learning
Simply Brilliant

Freestyle or templates based UI development
SAPUI5 / SAP Fiori Elements

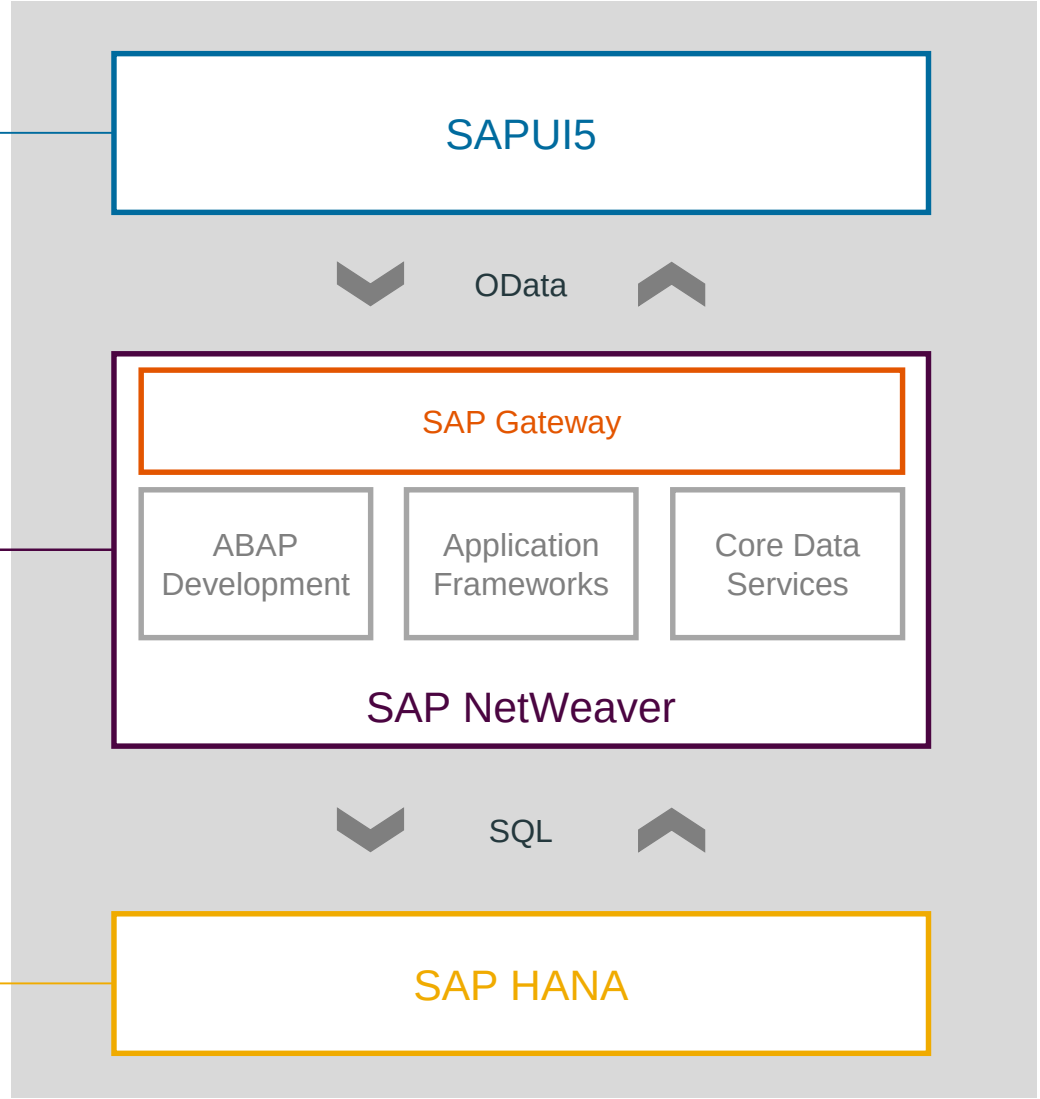
OData protocol infrastructure
SAP Gateway

Effective and efficient application development
Modern ABAP language and development tools

Transaction, error and event handling services
Business object framework (BOPF)

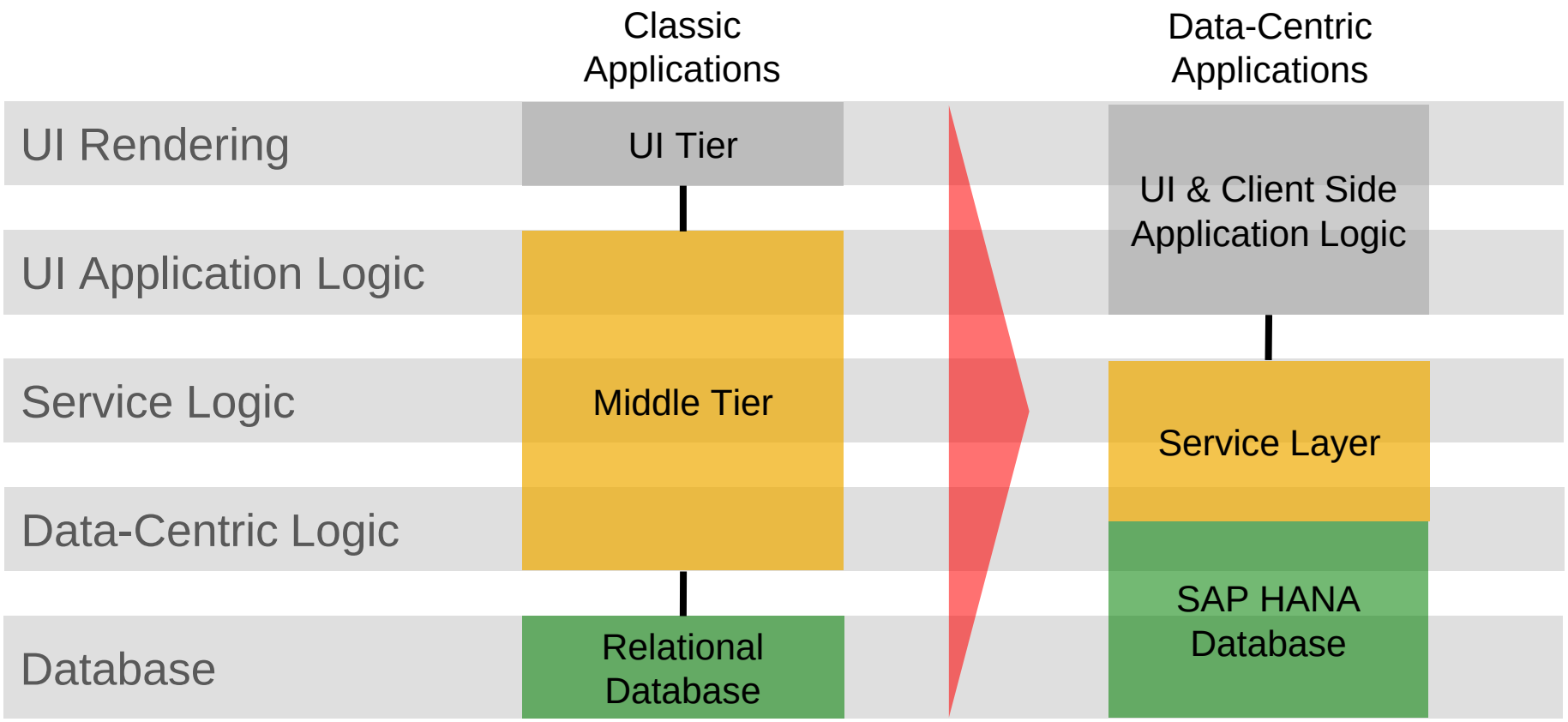
One common data modelling technology for all scenarios
Core Data Services (CDS)

Exploit SAP HANA features and performance
AMDP and CDS Table Functions



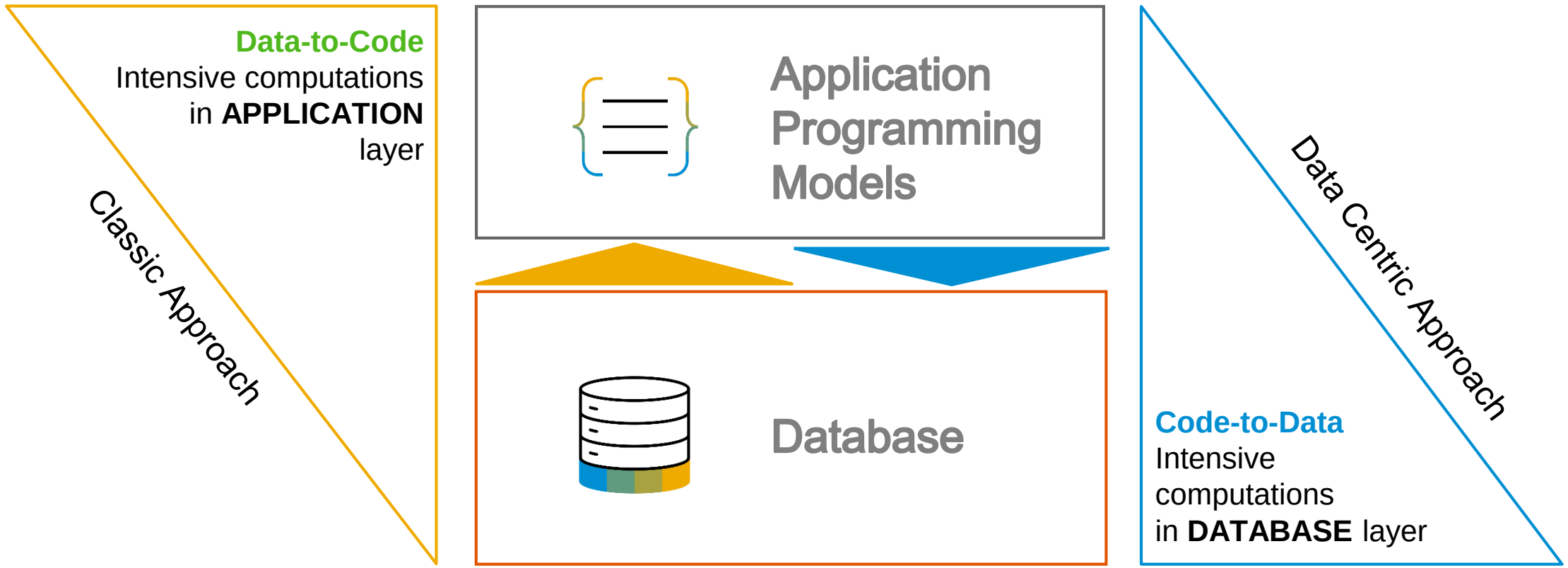
Transformation of Application Design with SAP HANA

SAP Development Learning
Simply Brilliant



Paradigm changes in application programming

SAP Development Learning
Simply Brilliant



Core Data Services at a Glance

SAP Development Learning
Simply Brilliant

1 **Semantically Rich Data-Models**

Domain specific languages (DDL, QL, DCL)

Declarative, close to conceptual thinking

2 **CDS is completely based on SQL**

Any ‘Standard SQL’ features directly available like joins, unions, build-in functions, …

3 **Fully Compatible with Any DB**

Generated and managed SQL Views

Modern Open SQL

4 **Common Basis for Domain-Specific Frameworks** e.g. UI, Analytics, Odata, BW,…

@AnalyticsDetails.aggregationBehaviour: SUM

5 **Built-in Functions and Code Pushdown**

Table Functions for Breakout Scenarios

Rich Set of Built-in SQL Functions

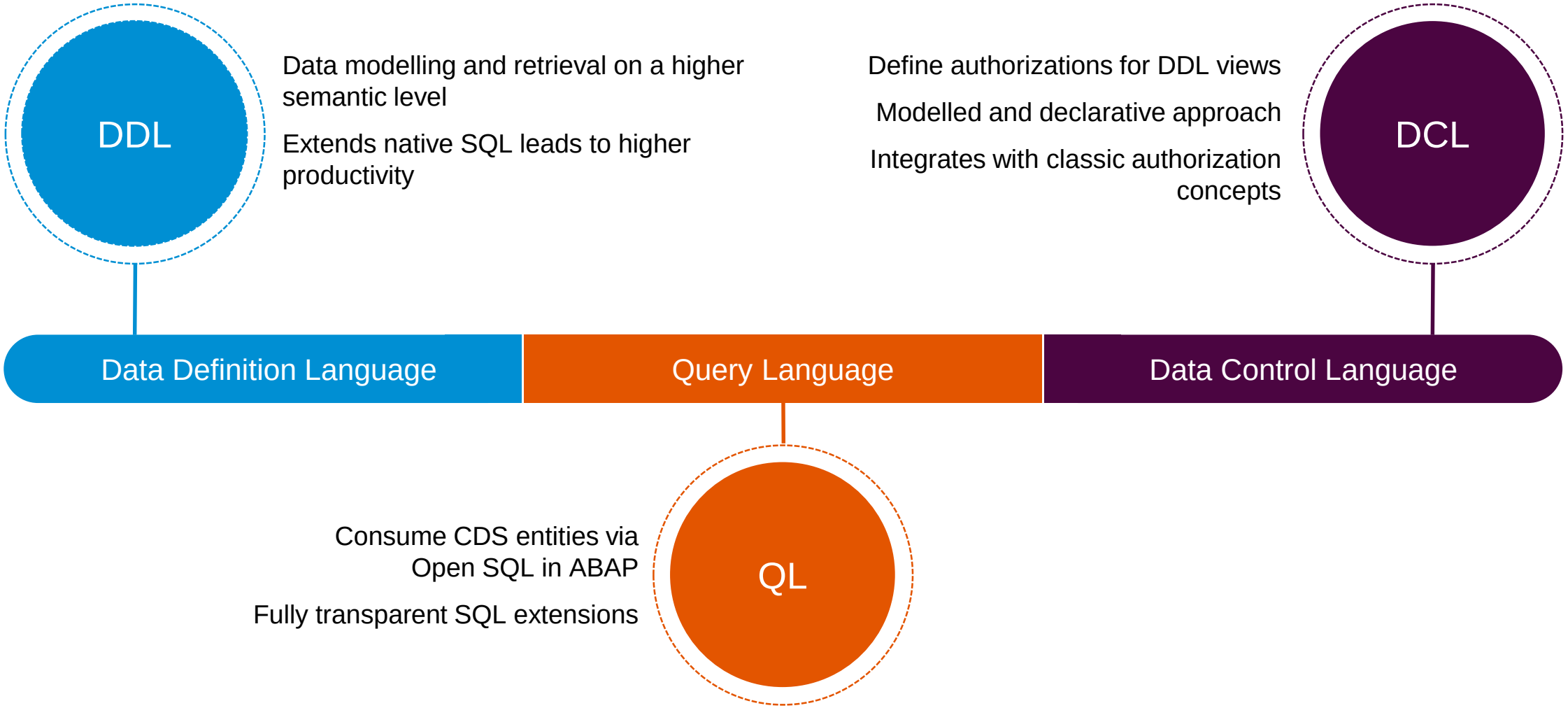
6 **Extensible**

On model level thru extensions

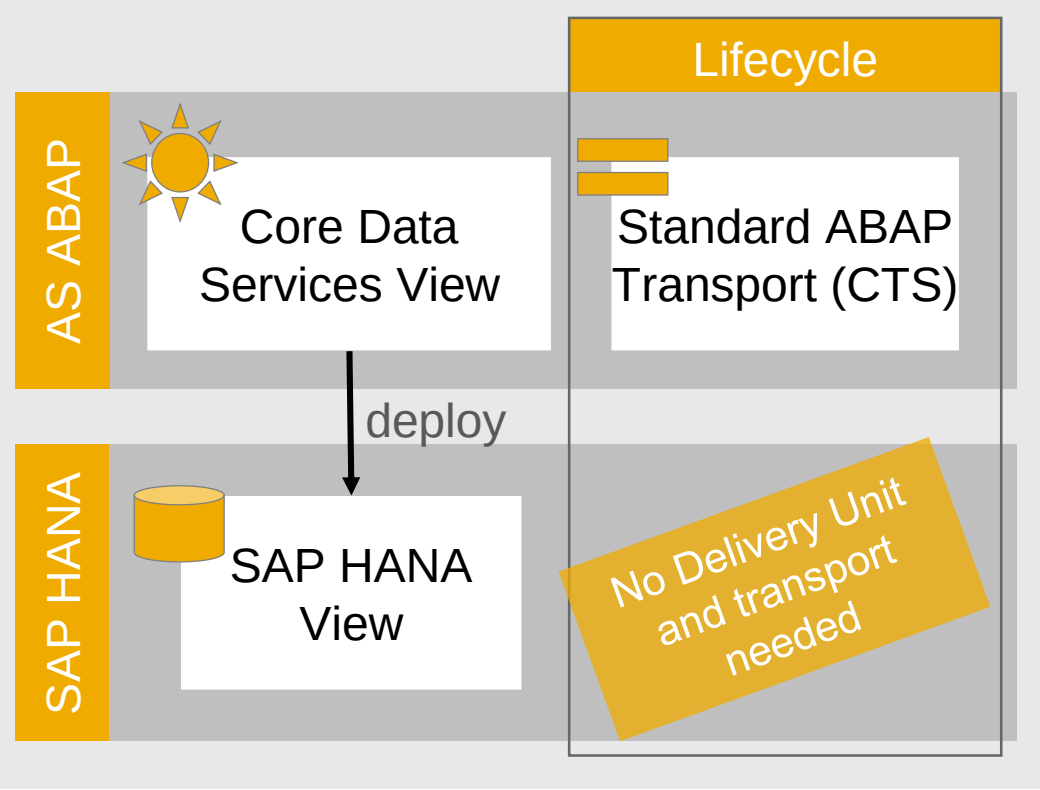
On meta-model level thru annotations

A family of domain specific languages

SAP Development Learning
Simply Brilliant



CDS Views in ABAP



ABAP stack as *Master* for editing, activating and transporting Core Data Services (CDS) View

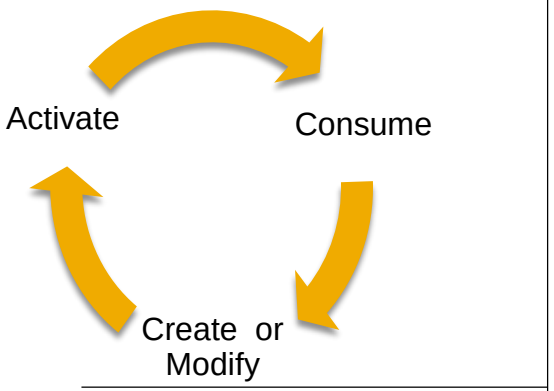
HANA view deployed during CDS view activation

Only CDS View definition is transported

Only ABAP Development Tools for SAP NetWeaver required

ABAP CDS provides read access to data only

CDS View Lifecycle – Create



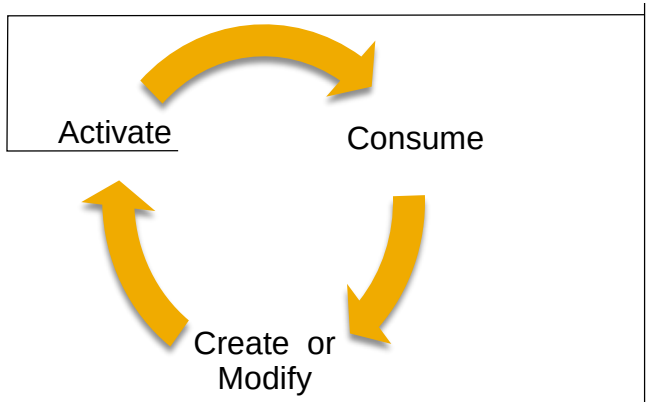
CDS is source based Tools:

ABAP Development Tools (eclipse based)

- Textual CDS source code editor
- Graphical CDS modelling view
- Dependency Analyzer
- Annotation viewer

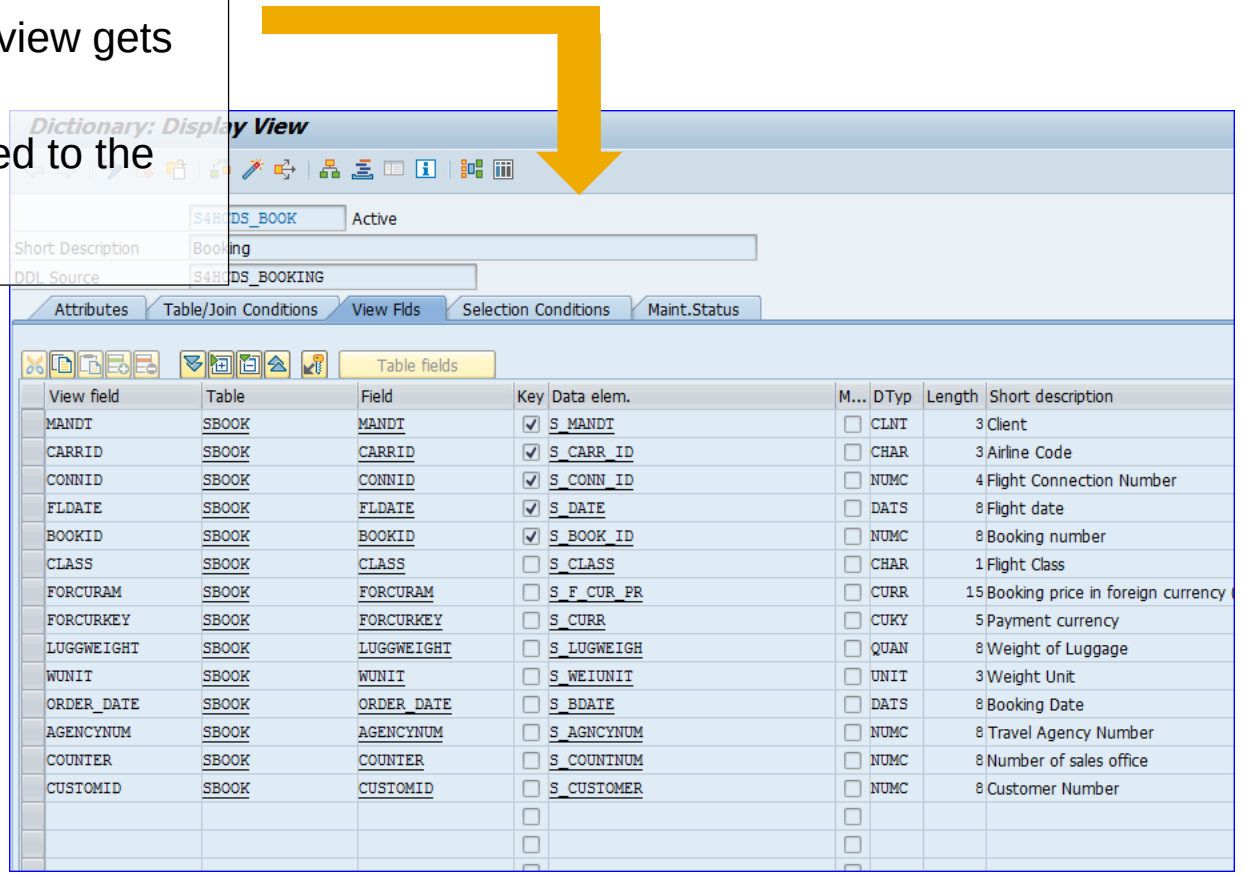
```
@AbapCatalog.sqlViewName:'S4HCDS_BOOK'
define view s4hcds_Booking as select from sbook
{
   key carrid,
   key connid,
   fldate,
   bookid,
   class
}
```

CDS View Lifecycle – Activate



During activation of a DDL source:

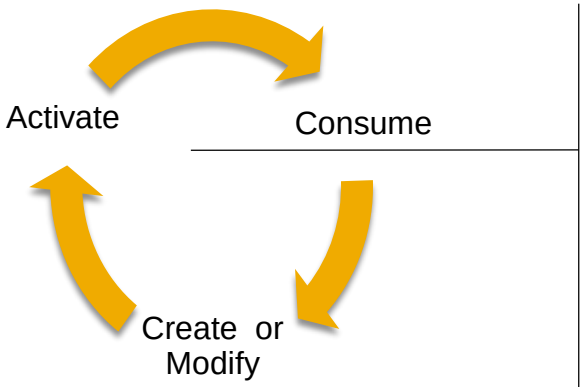
1. Corresponding DDIC view gets generated
2. The view gets deployed to the database



View field	Table	Field	Key	Data elem.	M...	DTyp	Length	Short description
MANDT	SBOOK	MANDT	☑	S_MANDT	☐	CLNT	3	Client
CARRID	SBOOK	CARRID	☑	S_CARR_ID	☐	CHAR	3	Airline Code
CONNID	SBOOK	CONNID	☑	S_CONN_ID	☐	NUMC	4	Flight Connection Number
FLDATE	SBOOK	FLDATE	☑	S_DATE	☐	DATS	8	Flight date
BOOKID	SBOOK	BOOKID	☑	S_BOOK_ID	☐	NUMC	8	Booking number
CLASS	SBOOK	CLASS	☐	S_CLASS	☐	CHAR	1	Flight Class
FORCURAM	SBOOK	FORCURAM	☐	S_F_CUR_PR	☐	CURR	15	Booking price in foreign currency
FORCURKEY	SBOOK	FORCURKEY	☐	S_CURR	☐	CUKY	5	Payment currency
LUGGWEIGHT	SBOOK	LUGGWEIGHT	☐	S_LUGWEIGH	☐	QUAN	8	Weight of Luggage
WUNIT	SBOOK	WUNIT	☐	S_WEIUNIT	☐	UNIT	3	Weight Unit
ORDER_DATE	SBOOK	ORDER_DATE	☐	S_BDATE	☐	DATS	8	Booking Date
AGENCYNUM	SBOOK	AGENCYNUM	☐	S_AGNCYNUM	☐	NUMC	8	Travel Agency Number
COUNTER	SBOOK	COUNTER	☐	S_COUNTNUM	☐	NUMC	8	Number of sales office
CUSTOMID	SBOOK	CUSTOMID	☐	S_CUSTOMER	☐	NUMC	8	Customer Number

CDS View Lifecycle – Consume

SAP Development Learning
Simply Brilliant



Open SQL

Application frameworks consuming CDS annotations

Annotation API

Tools:
ABAP source editors (SE38, SE24, ADT)
Trace Tools (SQLM, ST05)

```
REPORT Y_CDS_CONSUMPTION_EXAMPLE.

DATA lt_result TYPE STANDARD TABLE OF s4hcds_Booking.
SELECT * UP TO 10 ROWS
     FROM s4hcds_Booking
     INTO TABLE @lt_result
     WHERE bookid BETWEEN '25' and '30'.

   cl_demo_output=>display_data( lt_result ).
```

Output

LT_RESULT

CARRID	CONNID	FLDATE	BOOKID	CLASS	FORCURAM	FORCURKEY	LUGGWEIGHT	WUNIT	ORDER_DATE	AGENCYNUM	COU
AZ	0555	2015-10-22	00000025	Y	157.25	EUR	0.0	KG	2015-08-24	00000122	0000
AZ	0555	2015-10-22	00000027	Y	157.25	EUR	0.0	KG	2015-08-24	00000093	0000
AA	0017	2015-10-22	00000026	C	829.82	EUR	0.0	KG	2015-07-10	00000105	0000
AA	0017	2015-10-22	00000027	C	783.71	EUR	0.0	KG	2015-09-26	00000061	0000
AA	0017	2015-10-22	00000029	C	922.01	EUR	0.0	KG	2015-05-30	00000093	0000
JL	0407	2015-10-24	00000027	C	1825.0	EUR	0.0	KG	2015-08-14	00000120	0000
DL	0106	2015-10-20	00000030	C	1222.02	USD	0.0	KG	2015-07-13	00000304	0000
JL	0407	2015-10-24	00000025	C	2712.0	CHF	0.0	KG	2015-06-12	00000108	0000
AZ	0555	2015-10-22	00000030	Y	118.11	AUD	0.0	KG	2015-06-08	00000102	0000
DL	0106	2015-10-20	00000025	C	1160.92	USD	0.0	KG	2015-07-10	00000323	0000

Generated SQL Statement for CDS View

SAP Development Learning
Simply Brilliant

Two ways to get the generated CREATE SQL statement in the database for the CDS view

1. Right Click on the CDS source editor in Eclipse and select “Show SQL CREATE Statement”
2. Open the generated database view in SE11 and select *Menu -> Extras -> CREATE Statement*

Example:

```
@AbapCatalog.sqlViewName:'S4HCDS_SEL_V09'
define view s4hcds_sel_v9 as select from
scarr as c
{
   c.carrname as Carrier,
   c.carrid as ID
}
```

```
CREATE VIEW "S4HCDS_SEL_V09" AS SELECT
  "C"."MANDT" AS "MANDT",
  "C"."CARRNAME" AS "CARRIER",
  "C"."CARRID" AS "ID"
FROM "SCARR" "C"
```

Data Preview

SAP Development Learning
Simply Brilliant

- Eclipse Tool to view the data of the CDS view
- Right click on the CDS source editor and select “Open With” -> “Data Preview” or simply F8

```
@AbapCatalog.sqlViewName:'S4HCDS_SEL_V09'
define view s4hcds_sel_v9 as select from scarr as c
{
   c.carrname as Carrier,
   c.carrid as ID
}
```

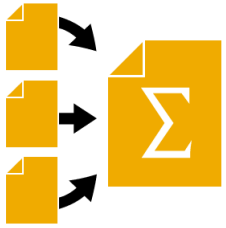
S4HCDS_SEL_V9

Raw Data

Filter pattern | 18 rows retrieved - 0 ms

Carrier	ID
American Airlines	AA
Air Canada	AC
Air France	AF
Alitalia	AZ
British Airways	BA
Air Pacific	FJ
Continental Airlines	CO
Delta Airlines	DL
Air Berlin	AB
Lufthansa	LH
Lauda Air	NG
Japan Airlines	JL
Northwest Airlines	NW
Qantas Airways	QF
South African Air.	SA
Singapore Airlines	SQ
Swiss	SR
United Airlines	UA

Lesson 1 – Motivation and Definition
Summary



You should now be able to:

- Describe the motivation for Core Data Services
- Define Core Data Services

Unit 1 – Introduction to Core Data Services

Lesson 2 – Basic Syntax of CDS Views

CDS View - Define View

SAP Development Learning
Simply Brilliant

Syntax: DEFINE VIEW <cds_entity> AS

CDS View Entity (s4hcds_Booking)

- Carries more semantics than SQL View
- Can be consumed by Open SQL

SQL View (S4HCDS_BOOK)

- Defined using annotation (`@AbapCatalog.sqlViewName`)
- Generated on activation of CDS view
- Representation on Database

Select *

- Selects all columns from the underlying data source

```
@AbapCatalog.sqlViewName:'S4HCDS_BOOK'
define view s4hcds_Booking
    as select from sbook
    {
         carrid,
         connid,
         fldate,
         bookid,
         …
    }
```

```
@AbapCatalog.sqlViewName: 'S4HCDS_SEL_ALL'
define view S4hcds_Select_All as select
from sflight {
    *
}
```

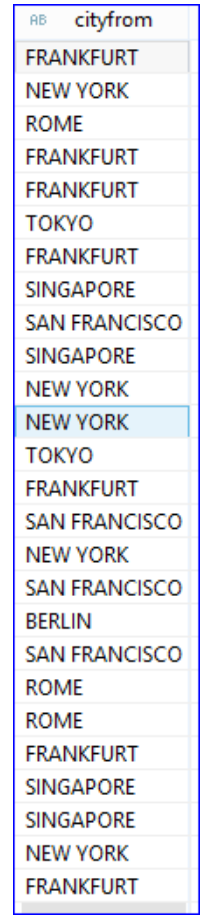
CDS Views – Select distinct

SAP Development Learning
Simply Brilliant

Duplicates can occur when a key column is not included in the projection list

```
@AbapCatalog.sqlViewName:'S4HCDS_SEL_V06A'
define view s4hcds_sel_v6a as select from spfli
{
    cityfrom
}
```

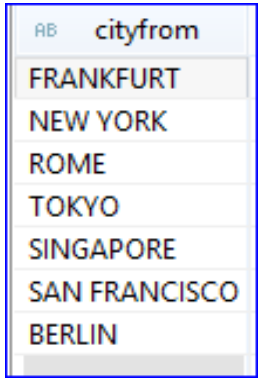
Output:



The keyword **DISTINCT** ensures that the result table contains no duplicates:

```
@AbapCatalog.sqlViewName:'S4HCDS_SEL_V06B'
define view s4hcds_sel_v6b as select distinct from spfli
{
    cityfrom
}
```

Output:



CDS Views - WHERE

SAP Development Learning
Simply Brilliant

- You can use compound WHERE clauses.
- You can reference the same column multiple times.

- Table aliases are defined in the **FROM** clause.
- You can use table aliases in the projection list.

```
@AbapCatalog.sqlViewName:'S4HCDS_SEL_V07'
define view s4hcds_sel_v7  as select from sflight as f
{
    f.carrid as ID,
    f.connid as Connection,
    f.planetype as PlaneType,
    f.seatsmax as MaxSeats
}
where
seatsmax <= 330 and seatsmax > 100 or planetype ='A340-600'
```

CAST Expression

SAP Development Learning
Simply Brilliant

- Supported Operands: Literal, column, path expression, build-in function, arithmetic expression
- Various data types in ABAP namespace supported
- No nesting of CAST expressions
- Alias names required for resulting columns

```
@AbapCatalog.sqlViewName: 'S4HCDS_SEL_VD12'
define view s4hcds_sel_v12 as select from sbook
{
    forcurkey,
    forcuram as Amount,
    cast( forcuram as abap.fltp )
    + ( cast ( -forcuram  as abap.fltp ) * 0.03 )
              as reduced_amount,
    cast( forcuram  as abap.fltp )* 0.03
              as savings
}
```

Example:

Cast foreign currency amount from sbook as reduced_amount and Savings.

Supported types in ABAP namespace:

char(len), clnt, cuky(len), curr(len, decimals), dats, dec(len, decimals), fltp, int1, int2, int4, lang, numc(len), quan(len, decimals), tims, unit(len)

Supported joins in CDS views

SAP Development Learning
Simply Brilliant

Supported Join Types:

1. Inner Join
2. Left Outer Join
3. Right Outer Join

- **Complex Join operations using (…) are supported**
- **Arbitrary On-Conditions**: >, >=, <, <=, Like, between, and, or, not

Inner Join: Output list of customer and booking details for booking id = 2406

```
@AbapCatalog.sqlViewName:'S4HCDS_JOIN_V01'
define view s4hcds_join_v1 as
select from sbook as b
     inner join spfli as p on p.carrid = b.carrid
                          and p.connid = b.connid
     inner join scustom as c on c.id = b.customid
{
    b.customid, c.name,
    b.fldate, p.cityfrom, p.cityto
}
where b.bookid = '00002406'
```

Left Outer Join: Output list of all customer details with booking id = 2406.

```
@AbapCatalog.sqlViewName: 'S4HCDS_LJOIN_01'
define view s4hcds_ljoin_demo1 as
select from scustom as c
   left outer join sbook as b on c.id = b.customid
{
   c.name as Customer_Name,
   c.city as Customer_city
}
where b.bookid = '00002406'
```

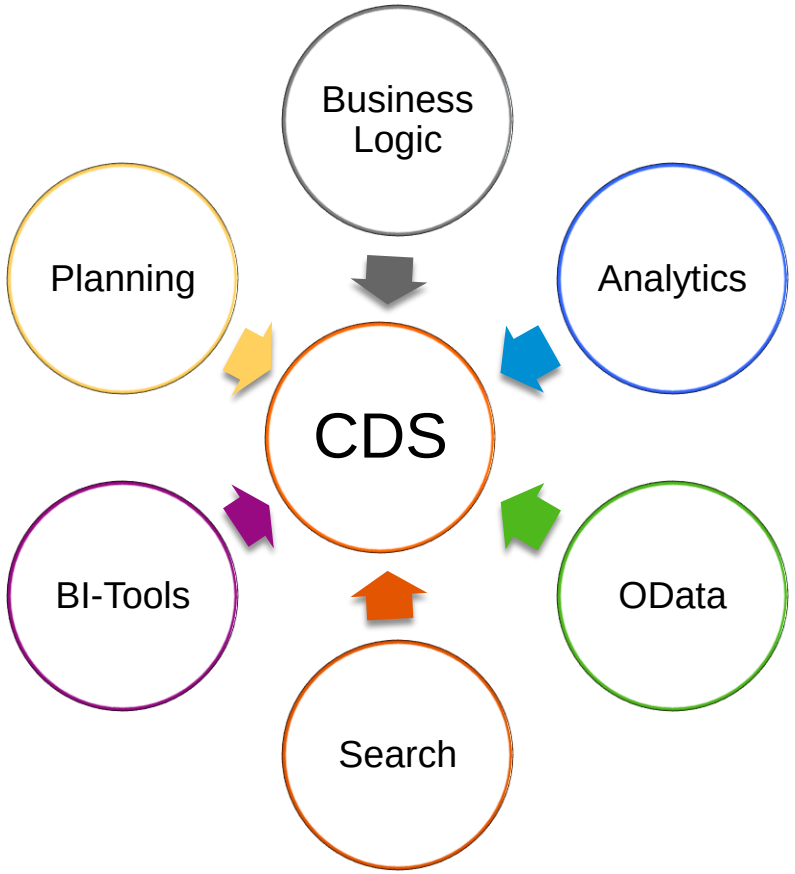
Unit 1 – Introduction to Core Data Services

Lesson 3 – Annotations

CDS: Common Basis for Domain-Specific Frameworks

SAP Development Learning
Simply Brilliant

- Reusable and unified view model for all use cases
- Annotations enabling flexible usage in different contexts
- Efficient development



Annotations

SAP Development Learning
Simply Brilliant

- Enrich CDS data models with additional metadata.
- Annotations begin with @
- ABAP Annotations:
  - ➢ Evaluated by the ABAP Core Data Services framework, namely the ABAP runtime environment itself
- Component Annotations:
  - ➢ Evaluated by frameworks of other SAP software components (e.g., ODATA, UI and Analytics)

Kinds of Annotations

SAP Development Learning
Simply Brilliant

- **AbapCatalog Annotations**
- **AccessControl Annotations**
- Analytics Annotations
- AnalyticsDetails Annotations
- **ClientDependent Annotations**
- **ClientHandling Annotations**
- Consumption Annotations
- **DataAging Annotations**
- DefaultAggregation Annotations
- **EndUserText Annotations**
- EnterpriseSearch Annotations
- **Environment Annotations**
- Hierarchy Annotations
- MappingRole Annotations
- ObjectModel Annotations
- OData Annotations
- Search Annotations
- SearchIndex Annotations
- Semantics Annotations
- UI Annotations
- VDM Annotations

Example

SAP Development Learning
Simply Brilliant

```
@AbapCatalog.sqlViewName: ‘S4HCDS_CONN3ANN‘

@AbapCatalog.compiler.compareFilter: true

@AbapCatalog.Buffering.type: #GENERIC

@AbapCatalog.Buffering.numberOfKeyFields: 1

@AbapCatalog.Buffering.status: #ACTIVE

@EndUserText.label: 'Flight Connection with ABAP Annotations‘

@ClientHandline.type: #CLIENT_DEPENDENT
@ClientHandline.algorithm: #AUTOMATED

define view s4hcds_Connection3_Annotations as select from spfli
{
// Projection list
}
```

Unit 2 – Advanced Concepts in CDS

Lesson 1 – Associations

CDS Associations – The Concept

SAP Development Learning
Simply Brilliant

Associations define relationships between entities in the data model

Association definition contains

- Target entity with optional alias (Recommended to start with _)
- Cardinality[min .. max] (optional)
- ON condition represents JOIN condition
  → easy to refactor

Consumption of Association

- From
- Projection list
- Aggregations
- WHERE, GROUP BY and HAVING clauses

Path Expressions Support

- Simplified consumption both in CDS view and Open SQL

Filter Expressions Support

```
@AbapCatalog.sqlViewName: 'S4HCDS_ASSOC_D1'
define view s4hcds_assoc_demo1
  as select from spfli
  association [1..1] to scarr as _scarr
     on spfli.carrid = _scarr.carrid
     {
       key spfli.carrid as id,

// Path expression
       key _scarr.carrname as carrier,

       key spfli.connid as flight,
       spfli.cityfrom as departure,
       spfli.cityto as destination
      }
```

Association Types

SAP Development Learning
Simply Brilliant

Ad Hoc Associations
- Association defined and used in the same CDS view → Association consumption constitutes a JOIN (Left Outer Join)

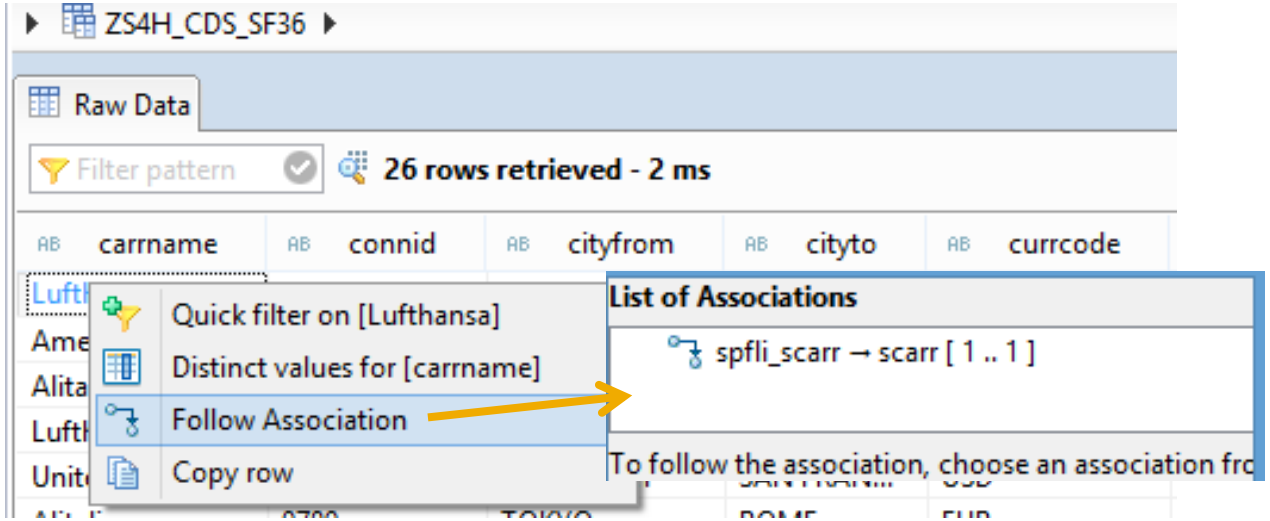
Exposed Association
- Association can be defined and exposed as part of the public signature of the view.
- Prerequisite: All columns of the view used in the Join-Condition have to be exposed as well, in this example carrid
- Consumption: From another CDS view or from Open SQL
  → Exposure does not automatically lead to a JOIN - "JOINs on demand"
- **$projection** – Alias name can be referred

```
@AbapCatalog.sqlViewName: 'S4HCDS_ASSOC_D2'
define view S4hcds_Assoc_Demo2 as
 select from spfli
     association [1..1] to scarr as _scarr
      on $projection.CarrierID = _scarr.carrid
{
       key _scarr.carrname,
       key spfli.connid,
       spfli.cityfrom,
       spfli.cityto,

// ad hoc association
       _scarr.currcode,

// field used in the ON condition
// Prerequisite for exposed association:
       spfli.carrid as CarrierID,
// exposed association
       _scarr
}
```

Using Exposed Associations In CDS Views

SAP Development Learning
Simply Brilliant

Use exposed associations In data preview



In other CDS views

- Use fields of associated entity as needed using path expression
- Leads to “JOIN on demand”
- CDS views can be defined for re-use

```
define view S4hcds_Assoc_Demo2b as
  select from S4hcds_Assoc_Demo2
{
      _scarr.carrid
}
```

Consuming Associations In Open SQL

SAP Development Learning
Simply Brilliant

Support for path expressions introduced in Open SQL with ABAP 7.5

- To allow consuming associations in CDS entities
- Simple path expressions supported, without filter expressions
- Association name prefixed with backslash \
- Supported in all clauses, e.g. field list, where, order by, grouping, having …

```
define view S4hcds_Assoc_Demo2 as select from spfli
    association[1..1] to scarr as _scarr
        on $projection.carrid = _scarr.carrid
{
  cityfrom, cityto,
  spfli.carrid,
// Exposed association:
  _scarr
}
```

```
SELECT FROM S4hcds_Assoc_Demo2
     FIELDS cityfrom, cityto,
             \_scarr-carrid as carrid,
             \_scarr-carrname as carrname,
             \_scarr-currcode as currcode
      WHERE \_scarr-currcode IN ( 'USD', 'EUR' )
      ORDER BY \_scarr-currcode INTO TABLE
@data(lt_customer).
```

Filtered Associations

SAP Development Learning
Simply Brilliant

Filtered Association

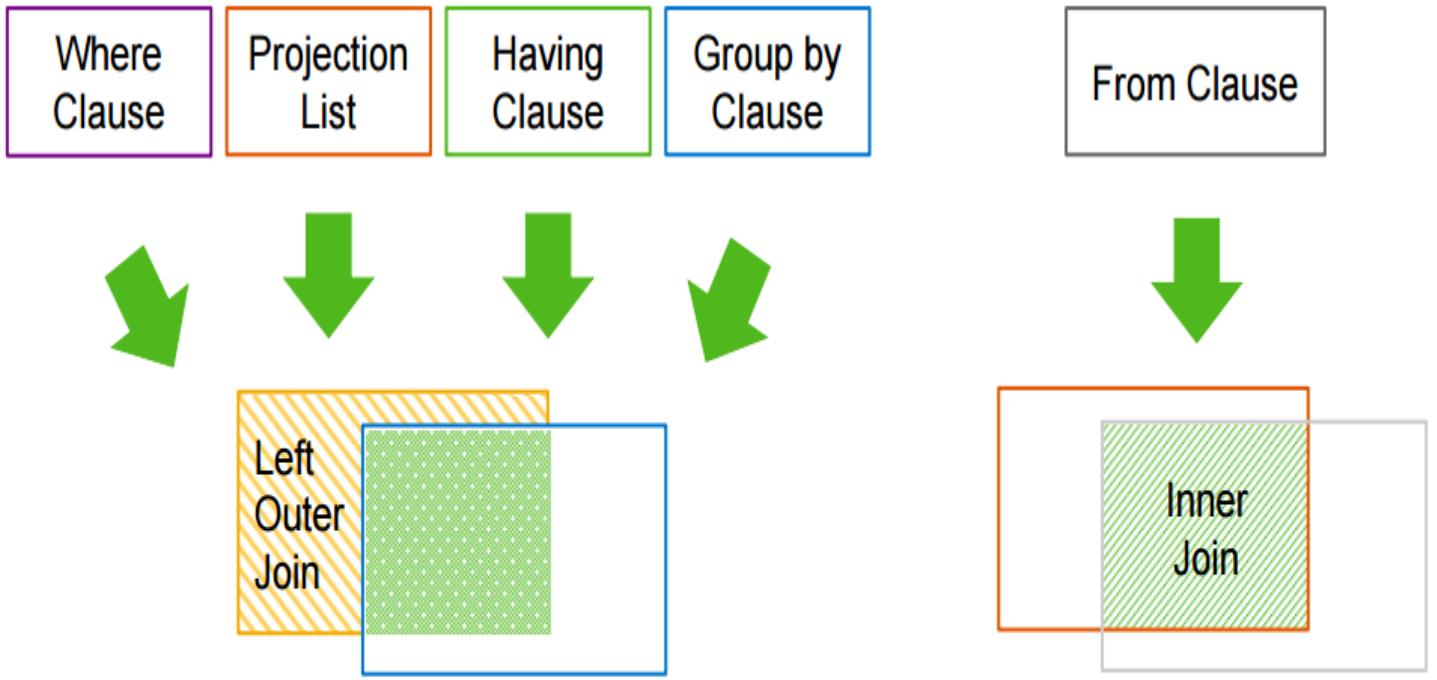
Associations can be enhanced by adding a supplementary filter

- Define association of cardinality “to-N”
- Use filter expressions in square brackets
- Usage of attribute **1:**
  (Declaration of monovalency)

```
define view S4hcds_Para_Demo2 as
  select from scustom as customer
    association[0..*] to sbook as _bookings
                    on customer.id = _bookings.customid
{ customer.id,
  customer.name,
// Exposed Association with filter
  _bookings[ class = 'C' ] as _business_flights,
// Exposed Association without filter
  _bookings }
```

```
define view S4hcds_Para_Demo3 as
  select from S4hcds_Para_Demo2
{ id,
  name,
// Path expressions with filter condition and 1:
  _bookings[1: fldate = '20150101'].carrid
       as ny_booking_carrid,
  _bookings[1: fldate = '20150101'].connid
       as ny_booking_connid
 }
```

CDS Associations – Translation into Joins



Used (!) associations are implicitly translated into SQL joins

Reuse of generated joins when semantically identical

CDS Associations: Advantages

SAP Development Learning
Simply Brilliant

Why would you use associations?

- Easy model consumption
  - Path expressions
  - Filter expressions
- Small(er) re-use views
- “JOINs on demand”:
  JOINs are only generated if the corresponding association is consumed