Core Data Services (CDS)

Tutorial 07: Learn about Associations to create "Joins on Demand"

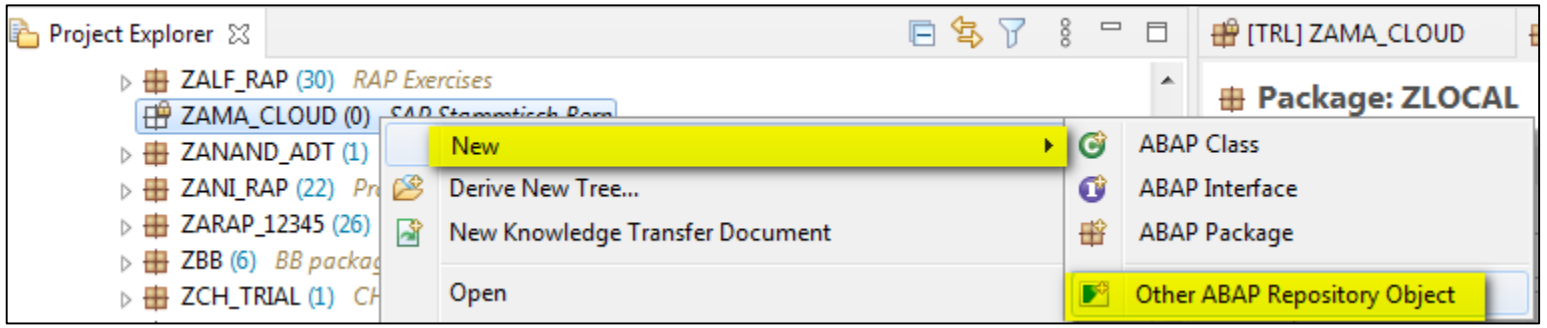
Alexander Maetzing, 13.11.2020

Associations

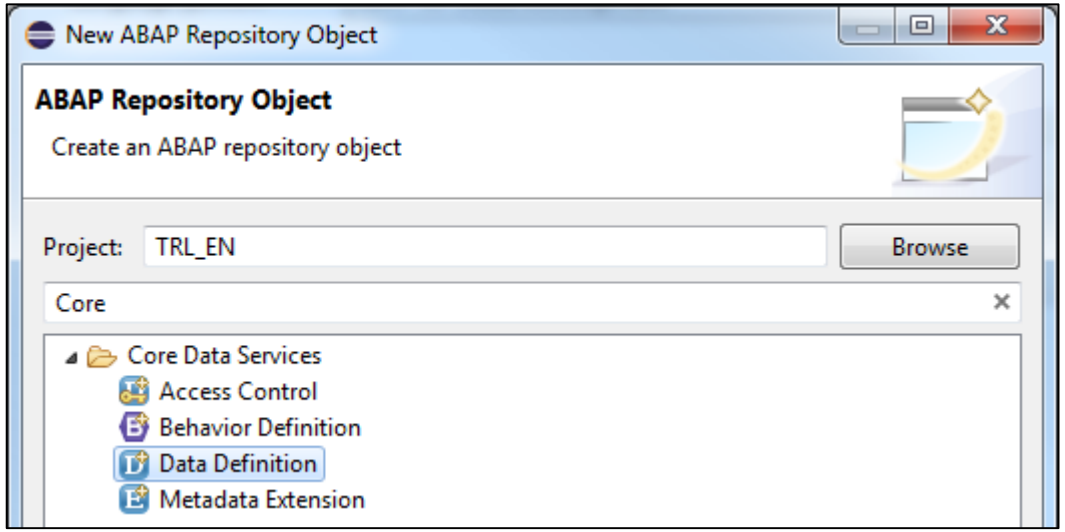
- A classical Join and an **Association** are basically very similar: They combine the accesses to two or more tables into one SQL (SELECT-) statement in a CDS.
- A Join inside a CDS is created statically as one combined access to the tables at designtime with an SQL CREATE statement. This so defined combined table access is always used at runtime of the CDS.
- An Association defines also a Join between two tables at designtime. During activation for each available Association a Join is created via an own SQL CREATE statement (plus one SELECT for the access to the basis table). This leads to a set of SELECTs for the processing of the CDS-View.
- At runtime however, the SELECTs to the Joins are only used if they are necessary (means the CDS selects data of both tables or the external access to the CDS requests data from both tables of the CDS). If the request is asking only for columns of the basis table, the SELECT only aims for that table at runtime.

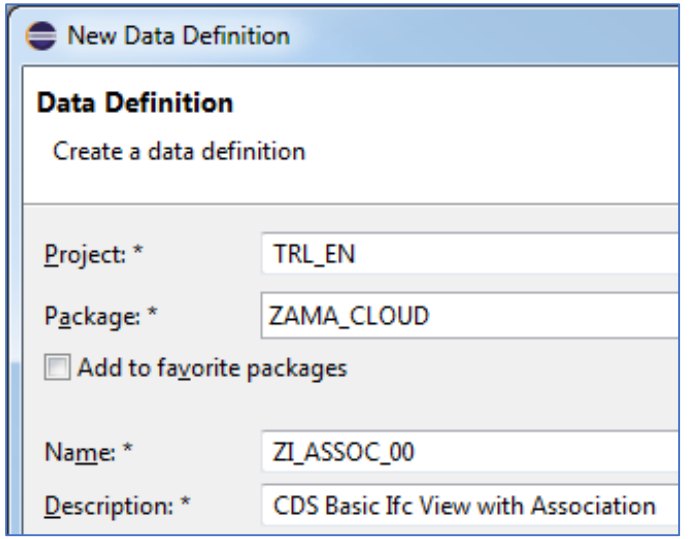
Step 01 - Create a new CDS-View (1)

- Create a new ABAP Repository Object.



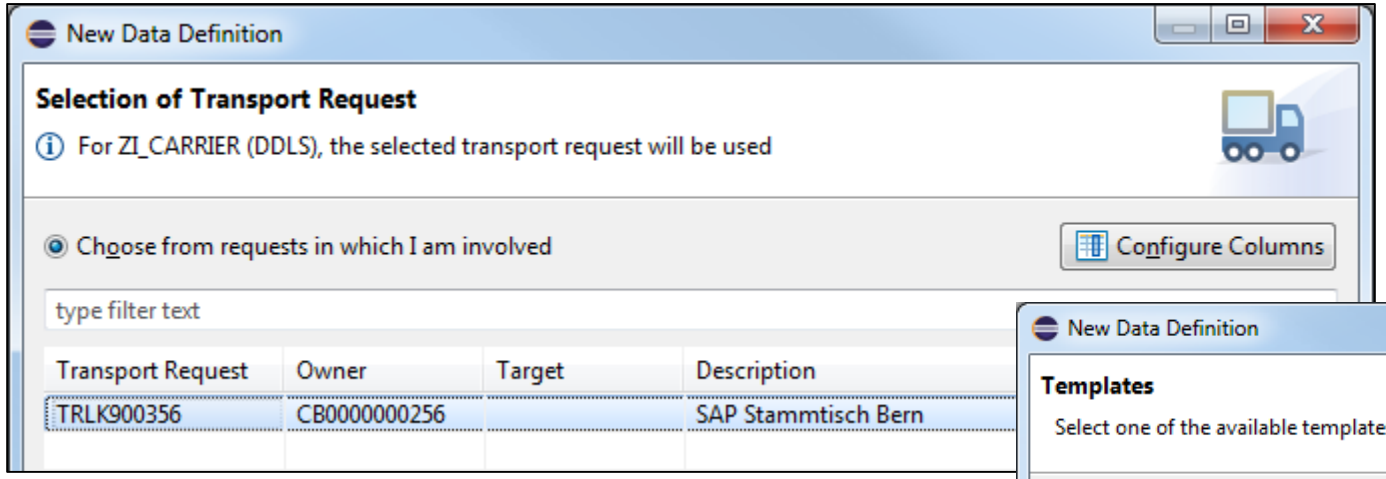
- Choose *Core Data Services -> Data Definition* and create a CDS-View ZI_ASSOC_##.



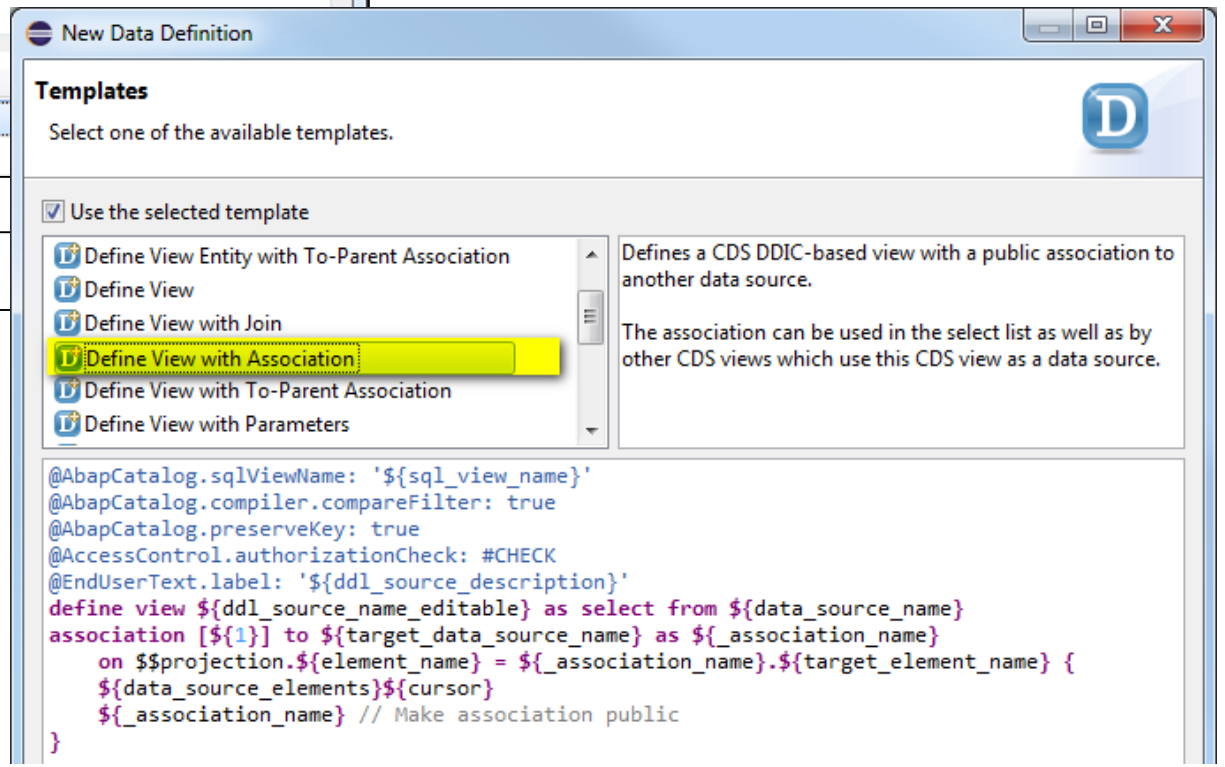


Step 01 - Create a new CDS-View (2)

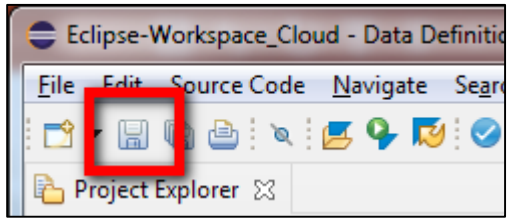
- Confirm the transport - and choose *Next*, **not** *Finish*!



- Select template *Define View with Association.*



- Don't forget to *Save* afterwards.



Step 02 - Fill header and field list

- Name the SQL-View and use the in an earlier Tutorial defined CDS ZI_FLIGHT_00 as basic data source of the new CDS. Use an alias for that data source.

[TRL] ZI_ASSOC_00

```
1 @AbapCatalog.sqlViewName: 'ZI_ASSOC_00_A'
2 @AbapCatalog.compiler.compareFilter: true
3 @AbapCatalog.preserveKey: true
4 @AccessControl.authorizationCheck: #CHECK
5 @EndUserText.label: 'CDS Basic Ifc View with Association'
6 define view ZI_ASSOC_00 as select from ZI_FLIGHT_00 as f
7 association [1] to target_data_source_name as _association_name
8     on $projection.element_name = _association_name.target_element_name {
9
10
11     _association_name // Make association public
12 }
```

- Choose only some of the columns of the basic data source for the field list.

```
6 define view ZI_ASSOC_00 as select from ZI_FLIGHT_00 as f
7 association [1] to target_data_source_name as _association_name
8     on $projection.element_name = _association_name.target_element_name {
9
10     key f.CarrierId,
11     key f.ConnectionId,
12     key f.FlightDate,
13         f.Verbindung,
14         f.AirportFromId,
15         f.AirportToId,
16
17     _association_name // Make association public
18
19 }
```

Step 03 - Define the Association

- Define the Association in the header of the CDS. Choose the also already created CDS ZI_CARRIER_00 as association target and name the Association *_carrier*.

```
7 association [1] to ZI_CARRIER_00 as _carrier
8     on $projection.element_name = _association_name.target_element_name {
```

  - Please see that the name of the Association should start with the character "_".
  - The character in cornered brackets after the key word `association` represents a cardinality between source and target data source. This is more or less documentation but may raise syntax warnings if the cardinality "to n" is used.

- Define the Join condition for the Association. While after the keyword *$projection* columns of the basic data source are expected, on the right side of the equal sign columns of the Association (means: the target data source) must be written.

```
7  association [1] to ZI_CARRIER_00 as _carrier
8      on $projection.CarrierId = _carrier. {
9
10     key f.CarrierId,
11     key f.ConnectionId,
12     key f.FlightDate,
13         f.Verbindung,
```

CarrierId - zi_carrier_00 as _carrier (column)
CurrencyCode - zi_carrier_00 as _carrier (column)
Name - zi_carrier_00 as _carrier (column)

Step 04 - Expose the Association

- Exposing an Association means to write down the theoretical possibility of the usage of the Join into the field list. For this write either the name of the association or selected fields of the association into the list.

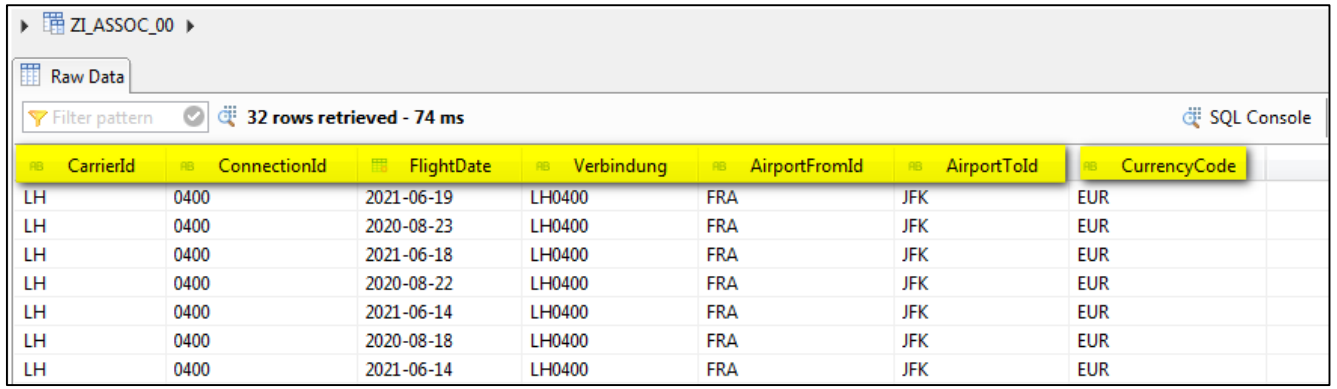
```
 6 define view ZI_ASSOC_00 as select from ZI_FLIGHT_00 as f
 7 association [1] to ZI_CARRIER_00 as _carrier
 8     on $projection.CarrierId = _carrier.CarrierId {
 9
10     key f.CarrierId,
11     key f.ConnectionId,
12     key f.FlightDate,
13         f.Verbindung,
14         f.AirportFromId,
15         f.AirportToId,
16
17     _carrier,
18     _carrier.CurrencyCode
19
20 }
```

  - Take care that each column used in the on-condition of the Association's definition is also in the field list (mandatory requirement!).
  - After the expose the target of the Association is visible for all consumers of the CDS View.

- Don't forget to save and activate the CDS.

Step 05 - See the data preview

- Use the data preview of the newly created CDS to have an access to the data. Note that only the data from the basic data source and the explicitely requested column from the target data source are shown.



CarrierId	ConnectionId	FlightDate	Verbindung	AirportFromId	AirportToId	CurrencyCode
LH	0400	2021-06-19	LH0400	FRA	JFK	EUR
LH	0400	2020-08-23	LH0400	FRA	JFK	EUR
LH	0400	2021-06-18	LH0400	FRA	JFK	EUR
LH	0400	2020-08-22	LH0400	FRA	JFK	EUR
LH	0400	2021-06-14	LH0400	FRA	JFK	EUR
LH	0400	2020-08-18	LH0400	FRA	JFK	EUR
LH	0400	2021-06-14	LH0400	FRA	JFK	EUR

- Examine the SQL Create Statement (CDS editor -> Context Menue) for the CDS. Also here the usage of the Association is not shown.
  Only the access to *CurrencyCode* is visible.

```
CREATE OR REPLACE VIEW "ZI_ASSOC_00_A" AS SELECT
  "F"."MANDT" AS "MANDT",
  "F"."CARRIERID",
  "F"."CONNECTIONID",
  "F"."FLIGHTDATE",
  "F"."VERBINDUNG",
  "F"."AIRPORTFROMID",
  "F"."AIRPORTTOID",
  "=A0"."CURRENCYCODE"
FROM "ZI_FLIGHT_00_A" "F" LEFT OUTER MANY TO ONE JOIN "ZI_CARRIER_00_A" "=A0" ON (
  "F"."MANDT" = "=A0"."MANDT" AND
  "F"."CARRIERID" = "=A0"."CARRIERID"
)
```

Step 06 - Change the exposition of the Association

- Remove the explicitely mentioned field *_carrier.CurrencyCode* from the field list and see the data preview. Only the data from the basic data source is visible.

```
15          f.AirportToId,
16
17      _carrier
18
19 }
```

ZI_ASSOC_00

Raw Data

Filter pattern | 32 rows retrieved - 44 ms

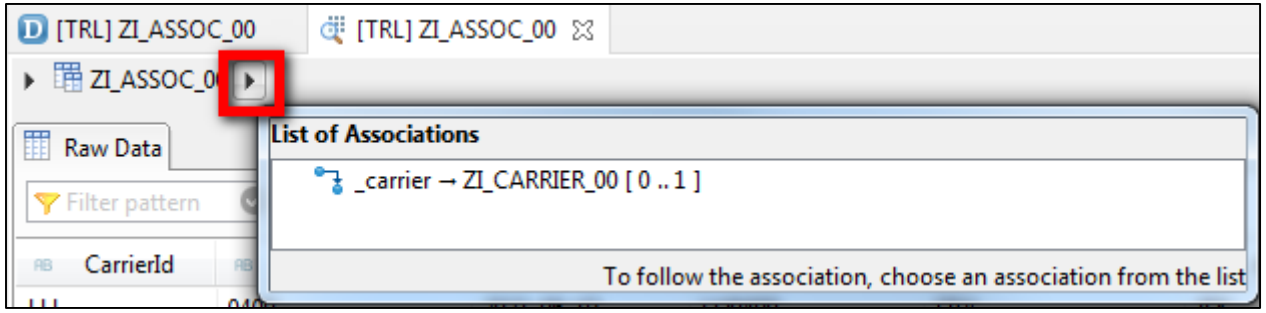
CarrierId	ConnectionId	FlightDate	Verbindung	AirportFromId	AirportToId
LH	0400	2021-06-19	LH0400	FRA	JFK
LH	0400	2020-08-23	LH0400	FRA	JFK
LH	0400	2021-06-18	LH0400	FRA	JFK

- The SQL Create Statement (CDS editor -> Context Menue) doesn't even has a Join.

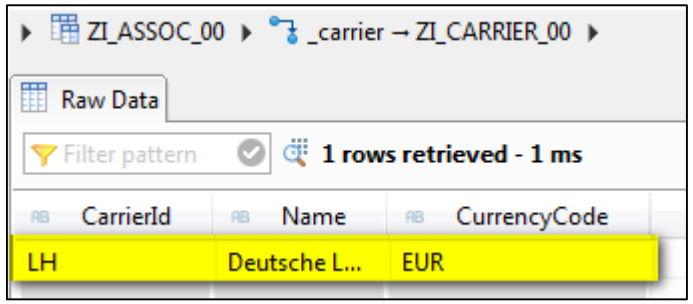
```
CREATE OR REPLACE VIEW "ZI_ASSOC_00_A" AS SELECT
  "F"."MANDT" AS "MANDT",
  "F"."CARRIERID",
  "F"."CONNECTIONID",
  "F"."FLIGHTDATE",
  "F"."VERBINDUNG",
  "F"."AIRPORTFROMID",
  "F"."AIRPORTTOID"
FROM "ZI_FLIGHT_00_A" "F"
```

Step 07 - Consume the Association in the data preview

- Call the data preview again and follow the small triangle after the name of the CDS-View ZI_ASSOC_##. A *List of Association* appears, showing *_carrier*.



- Clicking *_carrier* the data preview follows the association and shows the entry of the target data source for a selected record of the source table. In the current case no record was selected and the first one was therefore chosen implicitely.



- Note that before calling the association exposure (means: before clicking the small triangle) no Join was used. Only after that click the Join was processed by the runtime environment to achieve the result.

Step 08 - Access the basis table in ABAP (1)

- Use the ABAP class already created in an earlier tutorial and reuse the interface method `if_oo_adt_classrun~main()`.

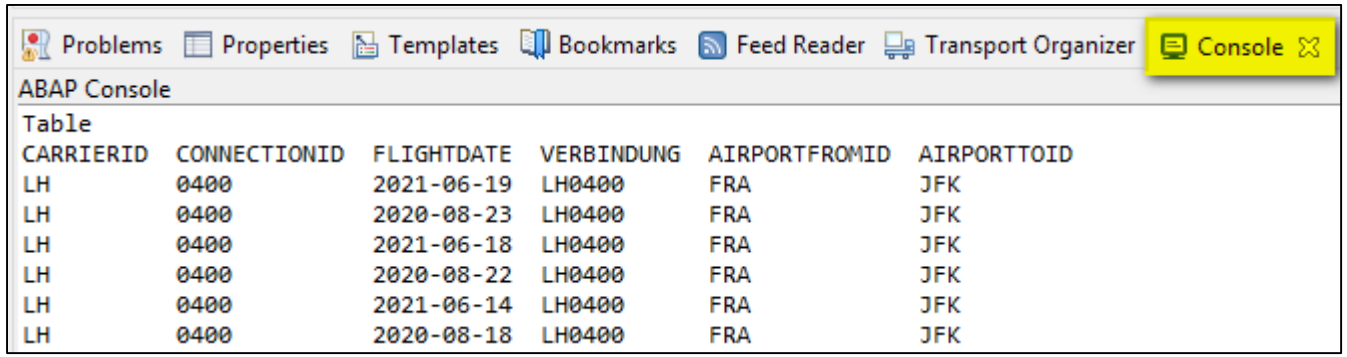
```
CLASS zama_cds DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.
    INTERFACES if_oo_adt_classrun.
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.
*********************************************
CLASS zama_cds IMPLEMENTATION.
  METHOD if_oo_adt_classrun~main.

  ENDMETHOD.
ENDCLASS.
```

- Write a normal SELECT statement to the CDS-View ZI_ASSOC_##. Please note also that although this is a consumation of a CDS-View and should be done only to a Consumption View ZC_* because of simplicity reasons the Composite Interface View ZI_* is chosen as data source.

```
    SELECT FROM zi_assoc_00
      FIELDS *
      INTO TABLE @DATA(gt_zi_assoc_00).
    out->write( gt_zi_assoc_00 ).
```

Step 08 - Access the basis table in ABAP (2)

- The console output shows only data from the basis table ZI_FLIGHT_00. No access to the records of the associated table ZI_CARRIER_00 is shown.



Step 09 - Consume the Association in ABAP (2)

- Enhance the SELECT statement by an access to the exposed Association.

```
15    SELECT FROM zi_assoc_00
16      FIELDS
17        CarrierId,
18        ConnectionId,
19        \_carrier-Name
20      INTO TABLE @DATA(gt_zi_assoc_00).
```

  - Note the syntax: the access to an Association must always start with the *PathPrefix* "\".
  - The similarity with the syntax of a modern *ABAP Mesh* access is not coincidentally.

- Using OPEN SQL or ABAP SQL brings a resultset with a fully processed Join following the association (not only one resultset for a given row as in the ADT preview).

Problems | Properties | Templates | Bookmarks | Feed Reader | Transport Organizer | Console

ABAP Console

Table

CARRIERID	CONNECTIONID	NAME
LH	0400	Deutsche Lufthansa AG
LH	0400	Deutsche Lufthansa AG
LH	0400	Deutsche Lufthansa AG
LH	0400	Deutsche Lufthansa AG
LH	0400	Deutsche Lufthansa AG
LH	0400	Deutsche Lufthansa AG

Appendix - Sourcecode


```
@AbapCatalog.sqlViewName: 'ZI_ASSOC_00_A'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'CDS Basic Ifc View with Association'
define view ZI_ASSOC_00 as select from ZI_FLIGHT_00 as f
association [1] to ZI_CARRIER_00 as _carrier
    on $projection.CarrierId = _carrier.CarrierId {

    key f.CarrierId,
    key f.ConnectionId,
    key f.FlightDate,
        f.Verbindung,
        f.AirportFromId,
        f.AirportToId,

    _carrier

}
```

```
CLASS zama_cds DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.
    INTERFACES if_oo_adt_classrun.
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.
*****************************************
CLASS zama_cds IMPLEMENTATION.
  METHOD if_oo_adt_classrun~main.

    SELECT FROM zi_assoc_00
      FIELDS
        CarrierId,
        ConnectionId,
        \_carrier-Name
      INTO TABLE @DATA(gt_zi_assoc_00).
    out->write( gt_zi_assoc_00 ).

  ENDMETHOD.
ENDCLASS.
```