
# Core Data Services (CDS)

Tutorial 04: Use the SQL dialect for Core Data Services to improve the data retrieval

Alexander Maetzing, 13.11.2020

# Step 01 - Create a new CDS or reuse an existing one


- Create a new Core Data Services ZI_FLIGHT_## as shown or reuse the already created one from tutorial 02 (with slight simplifications).

[TRL] ZI_FLIGHT_00

```
@AbapCatalog.sqlViewName: 'ZI_FLIGHT_00_A'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'CDS Composite Interface View'
define view ZI_FLIGHT_00
  as select from    /dmo/connection as c
    left outer join /dmo/flight     as f on c.carrier_id = f.carrier_id
{
  key c.carrier_id                        as CarrierId,
  key c.connection_id                     as ConnectionId,
  key f.flight_date                       as FlightDate,
      c.airport_from_id                   as AirportFromId,
      c.airport_to_id                     as AirportToId,
      c.departure_time                    as DepartureTime,
      c.arrival_time                      as ArrivalTime,
      c.distance                          as Distance,
      c.distance_unit                     as DistanceUnit,
      price                               as Price,
      currency_code                       as CurrencyCode,
      plane_type_id                       as PlaneTypeId
}
where
  c.carrier_id = 'LH'
```

# Step 02 - Arithmetical Operations and SQL functions

- Multiply, add and subtract values in the field list. Keep the information that the new column PriceBusiness is connected with the currency code in column CurrencyCode via an annotation.

```
19      c.distance_unit          as DistanceUnit,
20      price                    as Price,
21      @Semantics.amount.currencyCode: 'CurrencyCode'
22      2 * price + c.distance - 1234 as PriceBusiness,
23      currency_code            as CurrencyCode,
```

Documentation:
https://help.sap.com/doc/abapdocu_750_index_htm/7.50/de-de/abencds_f1_arithmetic_expression.htm

ZI_FLIGHT_00 — Raw Data — 32 rows retrieved - 1 ms

| CarrierId | ConnectionId | FlightDate | DistanceUnit | Price | PriceBusiness | CurrencyCode | PlaneTypeId |
|---|---|---|---|---|---|---|---|
| LH | 0400 | 2021-06-19 | KM | 5484.00 | 15896.00 | EUR | A340-600 |
| LH | 0400 | 2020-08-23 | KM | 2649.00 | 10226.00 | EUR | 767-200 |
| LH | 0400 | 2021-06-18 | KM | 3697.00 | 12322.00 | EUR | 747-400 |
| LH | 0400 | 2020-08-22 | KM | 4867.00 | 14662.00 | EUR | A380-800 |
| LH | 0400 | 2021-06-14 | KM | 4911.00 | 14750.00 | EUR | 767-200 |
| LH | 0400 | 2020-08-18 | KM | 3232.00 | 11392.00 | EUR | 747-400 |
| LH | 0400 | 2021-06-14 | KM | 2797.00 | 10522.00 | EUR | A340-600 |

- Round the distance to a value with two valid numbers (before the comma). Use the built-in function `round( )` with import parameters table column and the number of valid columns as negative digit.

```
16      c.arrival_time           as ArrivalTime,
17      c.distance               as Distance,
18      round( c.distance, -2 )  as DistanceRnd,
19      c.distance_unit          as DistanceUnit,
```

Documentation:
https://help.sap.com/doc/abapdocu_750_index_htm/7.50/de-de/abencds_f1_sql_functions_numeric.htm

ZI_FLIGHT_00 — Raw Data — 32 rows retrieved - 1 ms

| CarrierId | ConnectionId | FlightDate | ArrivalTime | Distance | DistanceRnd | DistanceUnit |
|---|---|---|---|---|---|---|
| LH | 0400 | 2021-06-19 | 11:34:00 AM | 6.162 | 6.200 | KM |
| LH | 0400 | 2020-08-23 | 11:34:00 AM | 6.162 | 6.200 | KM |
| LH | 0400 | 2021-06-18 | 11:34:00 AM | 6.162 | 6.200 | KM |
| LH | 0400 | 2020-08-22 | 11:34:00 AM | 6.162 | 6.200 | KM |
| LH | 0400 | 2021-06-14 | 11:34:00 AM | 6.162 | 6.200 | KM |
| LH | 0400 | 2020-08-18 | 11:34:00 AM | 6.162 | 6.200 | KM |
| LH | 0400 | 2021-06-14 | 11:34:00 AM | 6.162 | 6.200 | KM |

# Step 03 - String Operations and SQL functions

- Concatenate the values of two columns with the built-in function `concat( )`, using the columns to be concatenated as parameters.

```
10  key c.carrier_id                          as CarrierId,
11  key c.connection_id                       as ConnectionId,
12  key f.flight_date                         as FlightDate,
13      concat(c.carrier_id, c.connection_id) as Verbindung,
14      c.airport_from_id                     as AirportFromId,
```

Documentation:
https://help.sap.com/doc/abapdocu_750_index_htm/7.50/de-de/abencds_f1_sql_functions_character.htm

ZI_FLIGHT_00 ▸ Raw Data

Filter pattern | 32 rows retrieved - 1 ms

| CarrierId | ConnectionId | FlightDate | Verbindung | AirportFromId |
|---|---|---|---|---|
| LH | 0400 | 2021-06-19 | LH0400 | FRA |
| LH | 0400 | 2020-08-23 | LH0400 | FRA |
| LH | 0400 | 2021-06-18 | LH0400 | FRA |
| LH | 0400 | 2020-08-22 | LH0400 | FRA |
| LH | 0400 | 2021-06-14 | LH0400 | FRA |
| LH | 0400 | 2020-08-18 | LH0400 | FRA |
| LH | 0400 | 2021-06-14 | LH0400 | FRA |

- Use the built-in function `instr( )` to find the first occurence of a string in the value of the column. First parameter is the column to search, second parameter the string to look for.

```
14      c.airport_from_id                 as AirportFromId,
15      c.airport_to_id                   as AirportToId,
16      instr(c.airport_to_id, 'R')       as AirportToIdPos,
17      c.departure_time                  as DepartureTime,
```

ZI_FLIGHT_00 ▸ Raw Data

Filter pattern | 4 rows retrieved - 56 ms | SQL Console | Number of Entries | Select Columns | Add filter

| CarrierId | ConnectionId | FlightDate | AirportFromId | AirportToId | AirportToIdPos | DepartureTime |
|---|---|---|---|---|---|---|
| LH | 0400 | 2021-06-19 | FRA | JFK | 0 | 10:10:00 AM |
| LH | 0401 | 2021-06-19 | JFK | FRA | 2 | 06:30:00 PM |
| LH | 0402 | 2021-06-19 | FRA | EWR | 3 | 01:30:00 PM |
| LH | 0403 | 2021-06-19 | EWR | FRA | 2 | 06:09:00 PM |

# Step 04 - Date and Time Operations and SQL functions


- Create a new simple CDS-View ZI_BOOK_## based on table /DMO/BOOKING.

```
@AbapCatalog.sqlViewName: 'ZI_BOOK_00_A'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'CDS Basic Ifc View for /DMO/BOOKING'
define view ZI_BOOK_00
  as select from /dmo/booking
{
  key travel_id     as TravelId,
  key booking_id    as BookingId,
      booking_date  as BookingDate,
      customer_id   as CustomerId,
      carrier_id    as CarrierId,
      connection_id as ConnectionId,
      flight_date   as FlightDate,
      flight_price  as FlightPrice,
      currency_code as CurrencyCode
}
```

Documentation:
https://help.sap.com/doc/abapdocu_751_index_htm/7.51/en-US/abencds_f1_date_functions.htm

- Calculate the time-/date difference between the values of two columns with the built-in function `dats_days_between()`, using the columns to be evaluated as parameters.

```
      booking_date                                as BookingDate,
      customer_id                                 as CustomerId,
      carrier_id                                  as CarrierId,
      connection_id                               as ConnectionId,
      flight_date                                 as FlightDate,
      dats_days_between(booking_date, flight_date) as DateDelta,
```

Raw Data

Filter pattern — 101 rows retrieved - 1 ms (partial result)

| TravelId | BookingId | BookingDate | FlightDate | DateDelta |
|---|---|---|---|---|
| 00000001 | 0001 | 2020-08-05 | 2020-08-22 | 17 |
| 00000001 | 0002 | 2020-08-22 | 2020-08-24 | 2 |
| 00000001 | 0003 | 2021-05-30 | 2021-06-18 | 19 |
| 00000001 | 0004 | 2021-06-03 | 2021-06-20 | 17 |
| 00000002 | 0001 | 2020-08-20 | 2020-08-22 | 2 |

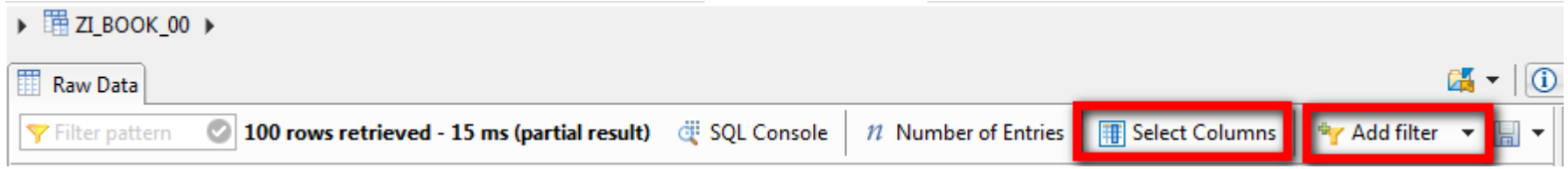
# Step 05 - Case Distinctions (1)

- Use the CDS SQL statement CASE to distinguish between simple conditions ("simple case").

```
16      dats_days_between(booking_date, flight_date) as DateDelta,
17      case dats_days_between(booking_date, flight_date)
18           when 0 then 'X'
19           when 1 then 'Y'
20           else ' '
21      end                                          as LastMinute,
```

Documentation:
https://help.sap.com/doc/abapdocu_750_index_htm/7.50/en-US/abencds_f1_case_expression.htm

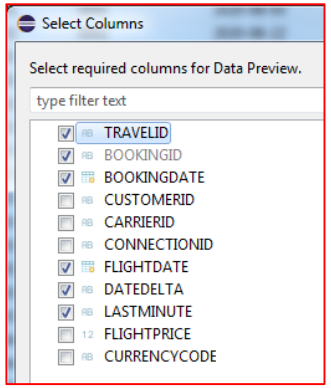
- Use filters and column selections in the data preview to reduce the resultset.

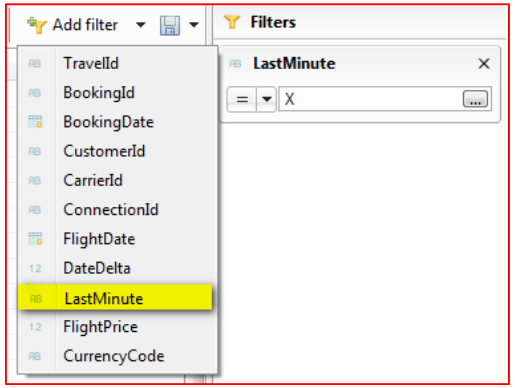



Raw Data

Filter pattern | 100 rows retrieved - 17 ms (partial result) | SQL Console | Number of Entries | Sele

| TravelId | BookingId | BookingDate | FlightDate | DateDelta | LastMinute |
|---|---|---|---|---|---|
| 00000012 | 0001 | 2020-08-22 | 2020-08-22 | 0 | X |
| 00000012 | 0002 | 2020-08-22 | 2020-08-22 | 0 | X |
| 00000012 | 0003 | 2020-08-22 | 2020-08-22 | 0 | X |
| 00000016 | 0007 | 2021-06-18 | 2021-06-18 | 0 | X |
| 00000016 | 0008 | 2021-06-18 | 2021-06-18 | 0 | X |
| 00000016 | 0009 | 2021-06-18 | 2021-06-18 | 0 | X |
| 00000022 | 0002 | 2020-08-24 | 2020-08-24 | 0 | X |



# Step 05 - Case Distinctions (2)

- Use the CDS SQL statement CASE to distinguish between complex conditions ("searched case").

```
21      end                                          as LastMinute,
22      case when flight_date > '2021-01-01' then 'next'
23           when flight_date < '2019-12-31' then 'last'
24           else 'this'
25      end                                          as FlightYear,
26      flight_price                                 as FlightPrice,
```

Raw Data

Filter pattern

101 rows retrieved - 1 ms (partial result)

| TravelId | BookingId | FlightDate | DateDelta | FlightYear |
|---|---|---|---|---|
| 00000001 | 0001 | 2020-08-22 | 17 | this |
| 00000001 | 0002 | 2020-08-24 | 2 | this |
| 00000001 | 0003 | 2021-06-18 | 19 | next |
| 00000001 | 0004 | 2021-06-20 | 17 | next |
| 00000002 | 0001 | 2020-08-22 | 2 | this |
| 00000002 | 0002 | 2020-08-22 | 2 | this |
| 00000003 | 0001 | 2020-08-22 | 20 | this |

# Step 06 - Casting of Data (1)

- Use the CDS SQL statement CAST to allow a type conversion inside the field list of a CDS. Use any DDIC type as a template for the casting.

```
 9      '20201113'                                    as StammtischBern,
10
11      //Use ABAP-DDIC types as template for a cast
12      cast('20201113' as abap.int8 ) as StammtischBernInt,
13      cast('20201113' as abap.dec(10,2) ) as StammtischBernDec,
14      cast('20201113' as abap.fltp ) as StammtischBernFltp,
15      cast('20201113' as abap.dats ) as StammtischBernDate,
```

Documentation:
https://help.sap.com/doc/abapdocu_750_index_htm/7.50/en-US/abencds_f1_cast_expression.htm

- Use any DDIC data element as a template for the casting. ATTENTION: In SAP Cloud only released Repository Objects can be applied!

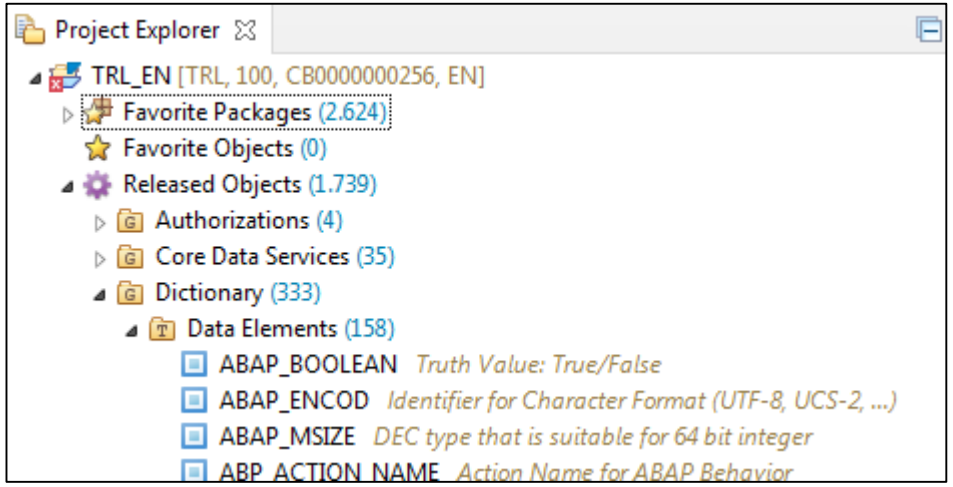
```
17      //Use any Data Element as a template for a cast
18      cast('20201113' as s_date preserving type ) as StammtischBernDateDE,
19      cast('X' as s_smoker) as StammtischBernBoolDE,
20      //SAP Cloud: Use any RELEASED Data Element as a template for a cast
21      cast('20201113' as xsddate_d) as StammtischBernDateDERel,
22      cast('1' as abap_boolean) as StammtischBernBoolDERel
```

Task Repositories | Task List | Problems | Properties | Templates | Feed Reader

2 errors, 0 warnings, 0 others

Description

- Errors (2 items)
  - Use of Data Element S_DATE is not permitted.
  - Use of Data Element S_SMOKER is not permitted.



# Step 06 - Casting of Data (2)

- Before arithmetical operatopns with "/", castings to a floating point type (e.g. *abap.fltp*) have to be done. But because of several casting restrictions, sources of a casting sometimes have to be casted several times to achieve the requested target type (*customer_id*: NUMC -> FLTP, *carrier_id*: CHAR -> FLTP).

```
24      //Before arithmetical operatopns with "/", castings to a floating point type have to be done
25      cast( cast(customer_id as abap.int4) as abap.fltp )
26        / cast( connection_id as abap.fltp) as StammtischBernRatio
```

- See the result of the different castings.

Raw Data | Show Log | Max. Rows: 100

Filter pattern | 100 rows retrieved - 14 ms (partial result) | SQL Console | Number of Entries | Select Columns | Add filter

| StammtischBern | StammtischBernInt | StammtischBe... | StammtischBernFltp | StammtischBernDate | Stammti... | | StammtischBernRatio |
|---|---|---|---|---|---|---|---|
| 20201113 | 20201113 | 20201113.00 | 2.0201113000000000E+07 | 2020-11-13 | 2020-11-13 | 1 | 3.8646714378659724E-01 |
| 20201113 | 20201113 | 20201113.00 | 2.0201113000000000E+07 | 2020-11-13 | 2020-11-13 | 1 | 1.8447204968944100E+00 |
| 20201113 | 20201113 | 20201113.00 | 2.0201113000000000E+07 | 2020-11-13 | 2020-11-13 | 1 | 3.8646714378659724E-01 |
| 20201113 | 20201113 | 20201113.00 | 2.0201113000000000E+07 | 2020-11-13 | 2020-11-13 | 1 | 1.8447204968944100E+00 |
| 20201113 | 20201113 | 20201113.00 | 2.0201113000000000E+07 | 2020-11-13 | 2020-11-13 | 1 | 6.4411190631099541E-02 |
| 20201113 | 20201113 | 20201113.00 | 2.0201113000000000E+07 | 2020-11-13 | 2020-11-13 | 1 | 4.2940793754066364E-01 |
| 20201113 | 20201113 | 20201113.00 | 2.0201113000000000E+07 | 2020-11-13 | 2020-11-13 | 1 | 6.0507482108002601E-02 |

# Appendix - Sourcecode (1)


```
@AbapCatalog.sqlViewName: 'ZI_FLIGHT_00_A'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'CDS Composite Interface View'
define view ZI_FLIGHT_00
  as select from    /dmo/connection as c
    left outer join /dmo/flight     as f on c.carrier_id =
f.carrier_id
{
  key c.carrier_id                          as CarrierId,
  key c.connection_id                       as ConnectionId,
  key f.flight_date                         as FlightDate,
      concat(c.carrier_id, c.connection_id) as Verbindung,
      c.airport_from_id                     as AirportFromId,
      c.airport_to_id                       as AirportToId,
      instr(c.airport_to_id, 'R')           as AirportToIdPos,
      c.departure_time                      as DepartureTime,
      c.arrival_time                        as ArrivalTime,
      c.distance                            as Distance,
      round( c.distance, -2 )               as DistanceRnd,
      c.distance_unit                       as DistanceUnit,
      price                                 as Price,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      2 * price + c.distance - 1234         as PriceBusiness,
      currency_code                         as CurrencyCode,
      plane_type_id                         as PlaneTypeId
}
where
  c.carrier_id = 'LH'
```

```
@AbapCatalog.sqlViewName: 'ZI_BOOK_00_A'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'CDS Basic Ifc View for /DMO/BOOKING'
define view ZI_BOOK_00
  as select from /dmo/booking
{
  key travel_id     as TravelId,
  key booking_id    as BookingId,
      booking_date  as BookingDate,
      customer_id   as CustomerId,
      carrier_id    as CarrierId,
      connection_id as ConnectionId,
      flight_date   as FlightDate,
      flight_price  as FlightPrice,
      currency_code as CurrencyCode
}
```

# Appendix - Sourcecode (2)


```
@AbapCatalog.sqlViewName: 'ZI_BOOK_00_A'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'CDS Basic Ifc View for /DMO/BOOKING'
define view ZI_BOOK_00
  as select from /dmo/booking
{
  key travel_id                                    as TravelId,
  key booking_id                                   as BookingId,
      booking_date                                 as BookingDate,
      customer_id                                  as CustomerId,
      carrier_id                                   as CarrierId,
      connection_id                                as ConnectionId,
      flight_date                                  as FlightDate,
      //Date and Time operations
      dats_days_between(booking_date, flight_date) as DateDelta,
      //Case distinctions
      case dats_days_between(booking_date, flight_date)
          when 0 then 'X'
          when 1 then 'Y'
          else ' '
      end                                          as LastMinute,
      case when flight_date > '2020-12-31' then 'next'
           when flight_date < '2020-01-01' then 'last'
           else 'this'
      end                                          as FlightYear,
      flight_price                                 as FlightPrice,
      currency_code                                as CurrencyCode,

      //Casting
      '20201113'                                   as StammtischBern,
      //Use ABAP-DDIC types as template for a cast
      cast('20201113' as abap.int8 )               as StammtischBernInt,
      cast('20201113' as abap.dec(10,2) )          as StammtischBernDec,
      cast('20201113' as abap.fltp )               as StammtischBernFltp,
      cast('20201113' as abap.dats )               as StammtischBernDate,

      //Use any Data Element as a template for a cast
      //cast('20201113' as s_date preserving type ) as StammtischBernDateDE,
      //cast('X' as s_smoker) as StammtischBernBoolDE,
      //SAP Cloud: Use any RELEASED Data Element as a template for a cast
      cast('20201113' as xsddate_d)                as StammtischBernDateDERel,
      cast('1' as abap_boolean)                    as StammtischBernBoolDERel,

      //Before arithmetical operatopns with "/", castings to a floating point
      //type have to be done
      cast( cast(customer_id as abap.int4) as abap.fltp )
        / cast( connection_id as abap.fltp)        as StammtischBernRatio

}
```