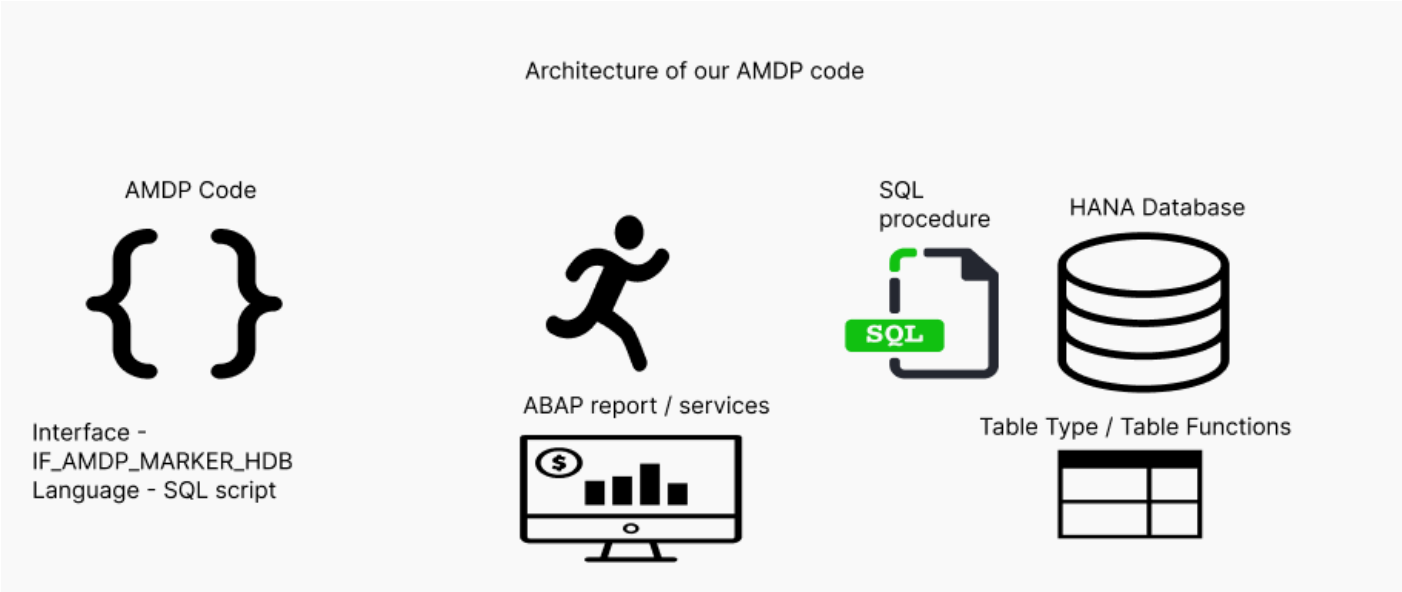
AMDP AND CDS framework

Note that < > are placeholders for the names we will be defining
What is AMDP - It is abap managed database procedure.

Architecture -



Implementation -
First thing to remember is that AMDP can only be implemented by making a class from eclipse or SE24 in SAP GUI . This is an OOPS concept.
Once your class has been made. We have to add a few things by which system will be able to differentiate that the defined class is an AMDP class.

Boiler Plate in Eclipse ADT class -

```
ZCL_AMDP_TEST
1  CLASS zcl_amdp_test DEFINITION
2    PUBLIC
3    FINAL
4    CREATE PUBLIC .
5  PUBLIC SECTION.
6
7  PROTECTED SECTION.
8  PRIVATE SECTION.
9  ENDCLASS.
10
11
12 CLASS zcl_amdp_test IMPLEMENTATION.
13 ENDCLASS.
```

Here after that PUBLIC Section we will be adding an interface which is given by SAP to tell the system that this following class is an AMDP class or a class where we will be writing the AMDP methods.
This will be written in the public section of our definition
`INTERFACES: if_amdp_marker_hdb.`

```
CLASS zcl_amdp_test DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .
PUBLIC SECTION.
INTERFACES: if_amdp_marker_hdb.

PROTECTED SECTION.
PRIVATE SECTION.
ENDCLASS.


CLASS zcl_amdp_test IMPLEMENTATION.
ENDCLASS.
```

Now since we have mentioned this interface we are definitely going to mention few other clauses which are given by SAP to be used with our AMDP classes as an AMDP class can contain normal ABAP methods as well as AMDP methods and the fact that the method is going to be an AMDP method or not is determined in the implementation part .

So , in addition to our definition , let us make one Structure type (this is done same as in ABAP (
`TYPES : BEGIN OF ...`
`END OF TY_XX.`
)
We will be writing a normal AMDP method , which will be a procedure

Difference between AMDP procedure (BY DATABASE PROCEDURE), AMDP function (BY DATABASE FUNCTION) , CDS TABLE FUNCTION (DEFINE TABLE FUNCTION , basic view)

- **AMDP procedures** can be called like methods of an ABAP class. They are implemented in SQLScript. The user of the method does not know that it is actually a database procedure.
- **CDS table** functions are AMDP functions that are encapsulated by a Core Data Services (CDS) object and can thus be called from ABAP or Open SQL like a normal database view with a SELECT query.
- **AMDP functions** for AMDP methods cannot be called directly from ABAP or Open SQL. However, they can be used when implementing other AMDP objects in SQLScript source code.

In the following table you can find a comparison of the most important features of the three objects.

	AMDP Procedure	CDS table function	AMDP function
Call from ABAP	like an ABAP method	in a SELECT statement	Not possible
Implementation	in a public instance method	in a public, static method of a static class	in a static or an instance method
Two-track development	Realizable via inheritance	possible by case distinction	not relevant, because only callable from other AMDP methods
Type of data access	read and write	read only	read only
Where are the parameters defined?	in the definition of the method	in the definition of the CDS object	in the definition of the method
Type of parameters	any IMPORTING , EXPORT and CHANGE	any scalar IMPORTING parameter and exactly one table-like RETURNING parameter	any scalar IMPORTING parameter and exactly one table-like RETURNING parameter

Comparison of the three AMDP objects

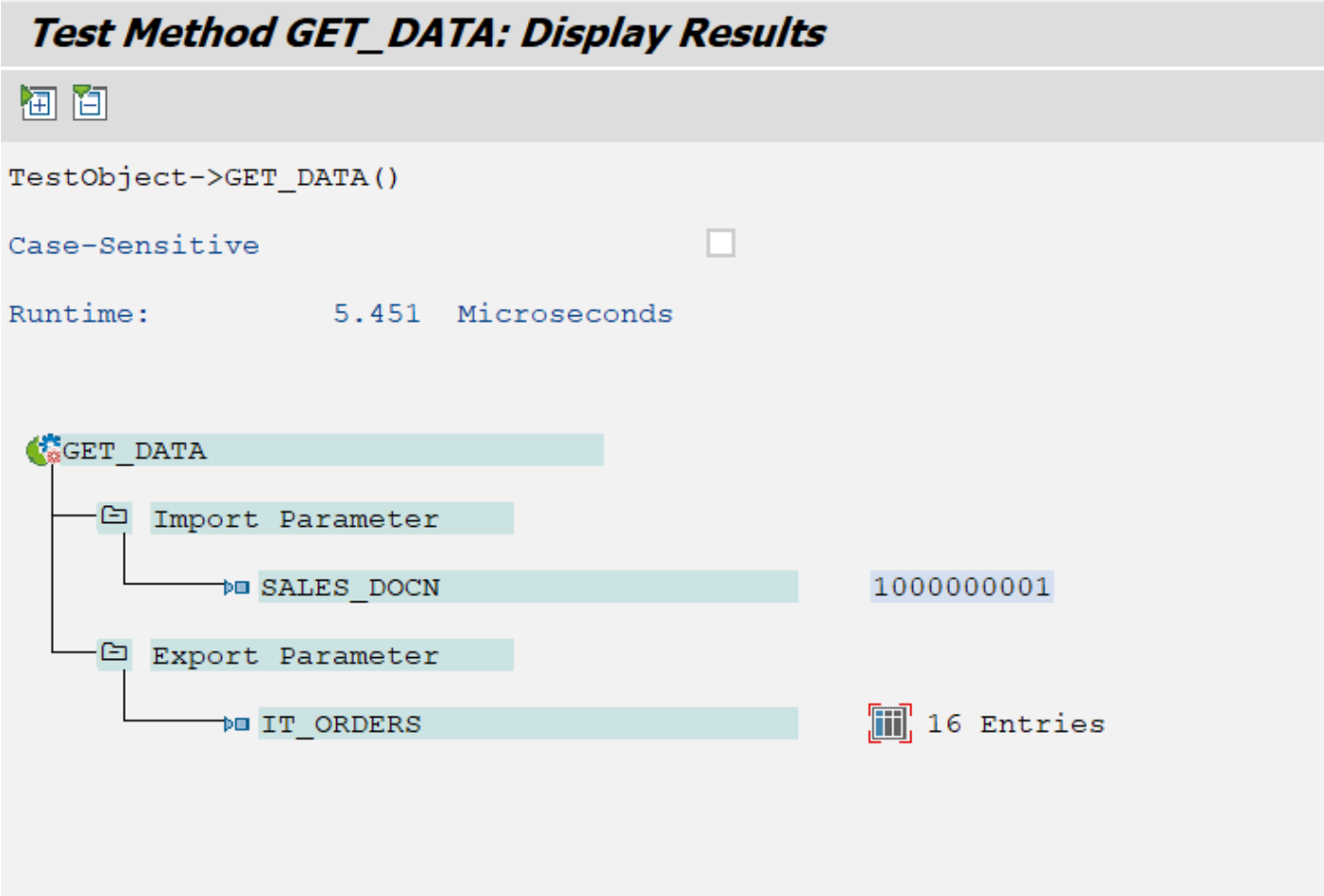
Self implementation

```
CLASS zcl_amdp_test DEFINITION
PUBLIC
FINAL
CREATE PUBLIC .
PUBLIC SECTION.
```

```
INTERFACES: if_amdp_marker_hdb. "marking this as an AMDP class as in , it can contain AMDP methods.
```

```
TYPES: BEGIN OF ty_orders,
vbeln TYPE vbak-vbeln,
posnr TYPE vbap-posnr,
bstnk TYPE vbak-bstnk,
NETWR type vbap-netwr,
end of TY_ORDERS.

TYPES: lt_orders TYPE STANDARD TABLE OF ty_orders.

METHODS: get_data
importing
value(sales_docn) TYPE vbak-vbeln
exporting
value(it_orders) TYPE lt_orders.
PROTECTED SECTION.
PRIVATE SECTION.
ENDCLASS.

CLASS zcl_amdp_test IMPLEMENTATION.

METHOD get_data BY DATABASE PROCEDURE FOR HDB
LANGUAGE SQLSCRIPT
OPTIONS READ-ONLY
USING vbak vbap.
it_orders = SELECT a.vbeln, b.posnr, a.bstnk, b.netwr
FROM vbak as a INNER JOIN vbap as b
ON a.vbeln = b.vbeln
WHERE a.vbeln = sales_docn;
ENDMETHOD.
ENDCLASS.
```

Output -


Test Method GET_DATA: Display Results
TestObject->GET_DATA()
Case-Sensitive
Runtime: 5.451 Microseconds
GET_DATA
Import Parameter
SALES_DOCN
1000000001
Export Parameter
IT_ORDERS
16 Entries

Structure Editor: Display GET_DATA.IT_ORDERS from Entry

Column Entry Metadata

16 Entries

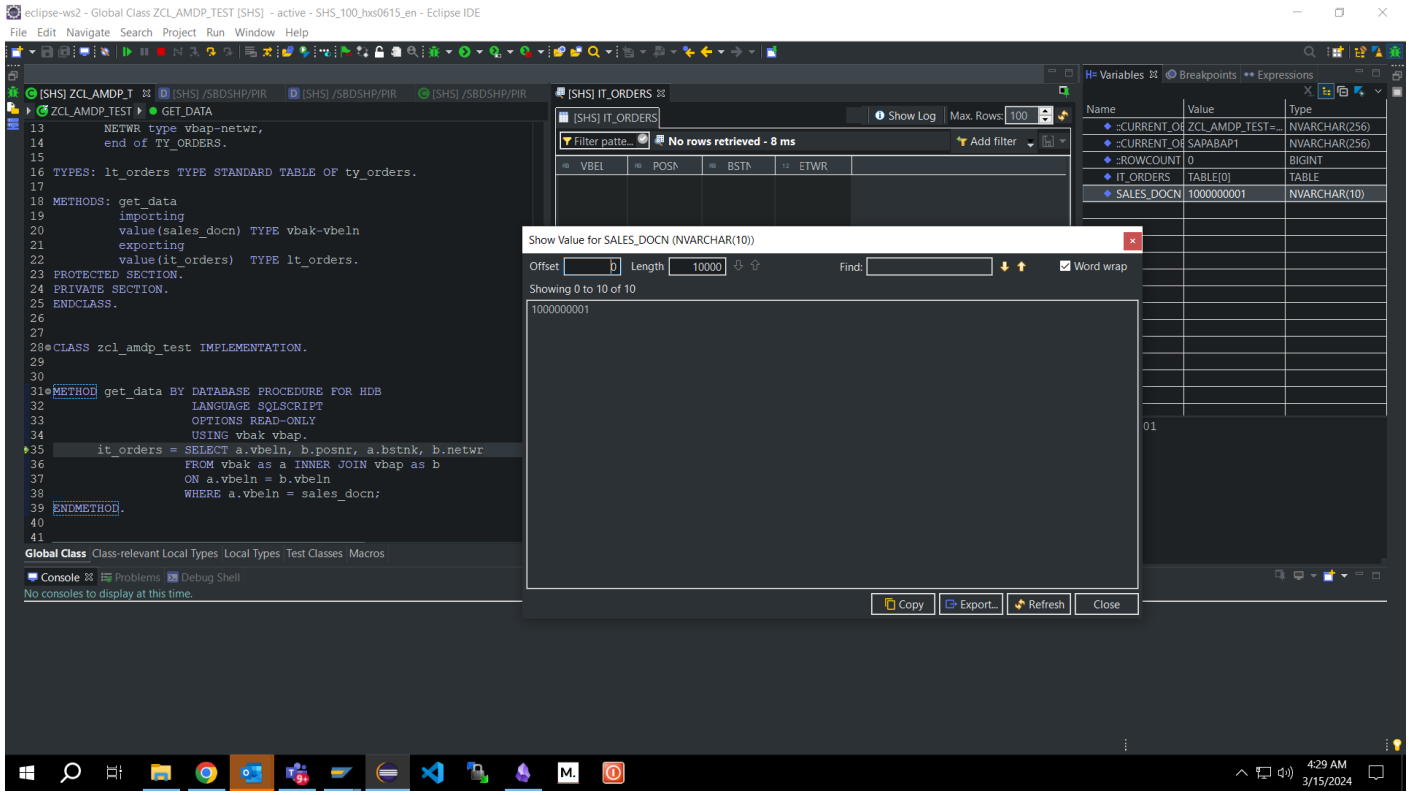
VBELN	POSNR	BSTNK	NETWR
1000000001	000010	4500206515 batch spl	0,00
1000000001	000010	test	0,00
1000000001	000010	test consigment fill	0,00
1000000001	000010	test	0,00
1000000001	000010	4500206515 batch spl	10.000,00
1000000001	000010	test	10.000,00
1000000001	000010	test consigment fill	10.000,00
1000000001	000010	test	10.000,00
1000000001	000010	4500206515 batch spl	2.030,00
1000000001	000010	test	2.030,00
1000000001	000010	test consigment fill	2.030,00
1000000001	000010	test	2.030,00
1000000001	000010	4500206515 batch spl	2.030,00
1000000001	000010	test	2.030,00
1000000001	000010	test consigment fill	2.030,00
1000000001	000010	test	2.030,00

Same can be debugged by AMDP and found -

Breakpoint -

```
34                    USING vbak vbap.
35        it_orders = SELECT a.vbeln, b.posnr, a.bstnk, b.netwr
36                    FROM vbak as a INNER JOIN vbap as b
37                    ON a.vbeln = b.vbeln
38                    WHERE a.vbeln = sales_docn;
39 ENDMETHOD
```

Value of the input -



F6 and the table input -

Filter patte... | 16 rows retrieved - 8 ms | Add filter

VBELN	POSNR	BSTNK	ETWR
1000000001	000010	4500206515 batch spl	0.00
1000000001	000010	4500206515 batch spl	10000.00
1000000001	000010	4500206515 batch spl	2030.00
1000000001	000010	4500206515 batch spl	2030.00
1000000001	000010	test	0.00
1000000001	000010	test	10000.00
1000000001	000010	test	2030.00
1000000001	000010	test	2030.00
1000000001	000010	test consigment fill	0.00
1000000001	000010	test consigment fill	10000.00
1000000001	000010	test consigment fill	2030.00
1000000001	000010	test consigment fill	2030.00
1000000001	000010	test	0.00
1000000001	000010	test	10000.00
1000000001	000010	test	2030.00
1000000001	000010	test	2030.00

Same data can be seen here .
Note for database procedures always the value( ) function is to be used while defining importing and exporting parameter like below - only to be used specifically and importantly with DATBASE PROCEDURE

```
CLASS zcl_amdp_test DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .
  PUBLIC SECTION.

    INTERFACES: if_amdp_marker_hdb. "marking this as an AMDP class

    TYPES: BEGIN OF ty_orders,
             vbeln TYPE vbak-vbeln,
             posnr TYPE vbap-posnr,
             bstnk TYPE vbak-bstnk,
             netwr TYPE vbap-netwr,
           END OF ty_orders.

    TYPES: lt_orders TYPE STANDARD TABLE OF ty_orders.

    METHODS: get_data
      IMPORTING
        VALUE(sales_docn) TYPE vbak-vbeln
      EXPORTING
        VALUE(it_orders)  TYPE lt_orders.
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.
```

Table functions can only be made from the SAP HANA perspective -
Follow this -
`https://help.sap.com/docs/SAP_HANA_PLATFORM/fc5ace7a367c434190a8047881f92ed8/4a8422b5de3c4a249f43ce71aefd0a9b.html`

These table functions are always implemented with the AMDP framework.

An example of how they are defined -

We use annotations as same that are used in CDS views ,
VDM, viewtype and all .
Now basic difference in the definition of the table function with that of a CDS view is : -
That when defining a table function we have to use the following syntax -

```
DEFINE table function <name>
returns{
field1 :data_element;
field2 :data_element;
field3 :data_element;
} implemented by amdp class=>method;
```

Example-

```
@VDM.viewType: #BASIC
@ClientHandling.type: #CLIENT_DEPENDENT
@EndUserText.label: 'Table function to get PIR'
define table function /SBDSHP/PIR_EKPO
returns {
          mandt    :mandt;
          ebeln    :ebeln;
          ebelp    :ebelp;
          aedat    :aedat;
          matnr    :matnr;
          werks    :ewerk;
          matkl    :matkl;
          infnr    :infnr;
          netpr    :netpr;
          peinh    :epein;
          brtwr    :bbwert;
          waers    :waers;
          mtart    :mtart; /*ENHC0058176 */
          lifnr    :lifnr;
          ltssf    :ltssf;
          idnlf    :idnlf;
          urzla    :uland;
          loekz    :loekz;
          datab    :datab;
          datbi    :datbi;
          knumh    :knumh;

}
implemented by method /sbdshp/pir_cl=>/sbdshp/pir_m;
```

This can be consumed in any CDS view or ABAP report by a simple select statement.
for the implementation of the AMDP method for any table function we have to use the clause for table function then the name of our table function -

```
class /SBDSHP/PIR_CL definition
  public
  final
  create public .

public section.
INTERFACES : if_amdp_marker_hdb.
CLASS-METHODS /SBDSHP/PIR_M FOR TABLE FUNCTION /SBDSHP/PIR_EKPO.
protected section.
private section.
ENDCLASS.
```

Basic declaration of AMDP variables -

`declare lv_clnt nvarchar( 3 ) := session_context('CLIENT');`

Constants -

`declare lc_clnt CONSTANT nvarchar( 3 ) :='100';`

CDS -

Basic Syntax -

`DEFINE VIEW <VIEW_NAME> AS` SELECT FROM TABLES JOIN TABLE `{`

`KEY <pr_key1>,`

`normal fields`

`}`

Example -

```
define view Z_TEST_CDS_SALES_MODEL
as select from mara as m_tab inner join vbap as v_tab on m_tab.matnr = v_tab.matnr {
    key v_tab.vbeln as sales_doc,
    key m_tab.matnr as material,
        m_tab.mtart as mat_type,
        v_tab.posnr as sales_item,
        v_tab.waerk as plant,
        v_tab.netwr as net_qty,
        v_tab.zmeng as target_qty,
        v_tab.zieme as uom
    }
```

Parameters how are they defined ?

Syntax -

`'DEFINE VIEW <SQL_VIEW> WITH PARAMETERS <param_name> : <data_elemnt> , as many as we want`

`, as select from tables JOIN {`

`key name,`

`key name ,`

`fields ,`

`fields,`
`}`
`where clause < no ambiguous filed > =: <param_name>`

Example -

```
define view z_test_cds_ship_sales with parameters p_matnr:matnr , p_vbeln:vbeln as select from Z_TEST_CDS_SALES_MODEL as test_sls
```

```
    _sales // Make association public
}   where _sales.vbelv is not initial
    and   _sales.vbelv=:p_vbeln
    or   test_sls.material=:p_matnr
```

Associations -
`https://community.sap.com/t5/enterprise-resource-planning-blogs-by-members/part-8-cds-views-joins-and-associations/ba-p/13406513`
Naming convention of an association (these are defined as aliases all of it is defined with alias)
`_name_of_association`

Now if we want to specify only few fields from the association then we can by simply mentioning the fields as
name.field_name

also , in the curly brackets keep the association alone as it makes the association go publis and useful !

Example -

```
define view z_test_cds_ship_sales with parameters p_matnr:matnr , p_vbeln:vbeln as select from Z_TEST_CDS_SALES_MODEL as test_sls
association [1..1] to vbfa as _sales on test_sls.sales_doc = _sales.vbelv and test_sls.material = _sales.matnr
{
    @Consumption.filter.mandatory: false
    key test_sls.material,
    key test_sls.sales_doc,
    test_sls.mat_type,
    test_sls.sales_item,
    test_sls.plant,
    test_sls.net_qty,
    test_sls.target_qty,
    test_sls.uom,
--    _sales.ruuid as Ruuid,
    _sales.posnv as Posnv,
    _sales.vbeln as Vbeln,
    _sales.vbelv as Vbelv,
    _sales.posnn as Posnn,
    _sales.vbtyp_n as VbtypN,
@DefaultAggregation: #SUM
    _sales.rfmng as Rfmng,
    _sales.meins as Meins,
    _sales.rfwrt as Rfwrt,
    _sales.waers as Waers,
    _sales.vbtyp_v as VbtypV,
    _sales // Make association public
}   where _sales.vbelv is not initial
    and   _sales.vbelv=:p_vbeln
    or   test_sls.material=:p_matnr
```

How to utilize the cds view in your abap report ?
Simply by using the select statement but if your cds has parameters then after the FROM keyword

when specifying the CDS view we should use the parenthesis and specify the cds parameter first then specify the parameter or value from abap end.
As we know value is assigned from right to left.

Example -

```
*&---------------------------------------------------------------------*
*& Form call_slsmdl_cds
*&---------------------------------------------------------------------*
*& text
*&---------------------------------------------------------------------*
*& -->  p1        text
*& <--  p2        text
*&---------------------------------------------------------------------*

FORM call_vbfa_cds.
  SELECT * FROM z_test_cds_ship_sales( p_matnr = @p_matnr , p_vbeln = @p_vbeln )
  INTO CORRESPONDING FIELDS OF TABLE @lt_sales.
  IF sy-subrc <> 0.
    MESSAGE: 'error occurred' TYPE 'E' DISPLAY LIKE 'I'.
  ENDIF.
ENDFORM.
```