Beginner's Guide
Programming with ABAP on HANA

by Pallavi Gupta, Accrete Solutions

What does this document have to offer?

The focus of this blog is to present an overview of the new programming techniques in ABAP after the introduction of HANA database. The focus will be towards providing a guideline on why and how an ABAP developer should start transitioning its code to use the new coding technique's.

Who should be reading this?

Here the target audience would be ABAP developers who are looking forward to getting a basic understanding of ABAP on HANA programming and to understand why to opt for these new features.

Areas covered in this blog…

Code to Data Paradigm, OpenSQL, CDS Views, AMDPs.

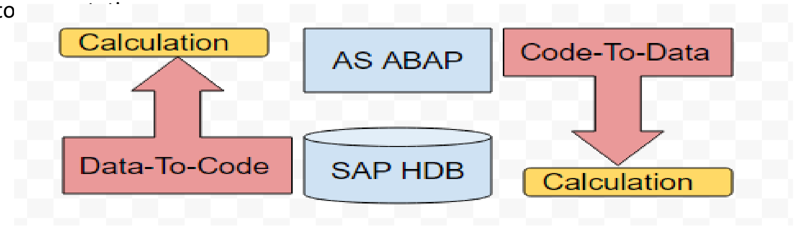
Let us begin!

SAP ABAP has been rapidly evolving over the years. With the introduction of S/4HANA, it went to graduate to become a far more impressive and productive language. If you ask me how ABAP has improved then the answer is "Code-To-Data" Paradigm.

What is Code-To-Data Paradigm?

The traditional approach involves bringing data from database to our presentation server, doing the data intensive calculations & filtering and then presenting the filtered data to a user.

The new HANA approach is to push our code to the database layer where all the data resides, do the calculations at database layer and bring only the relevant records to

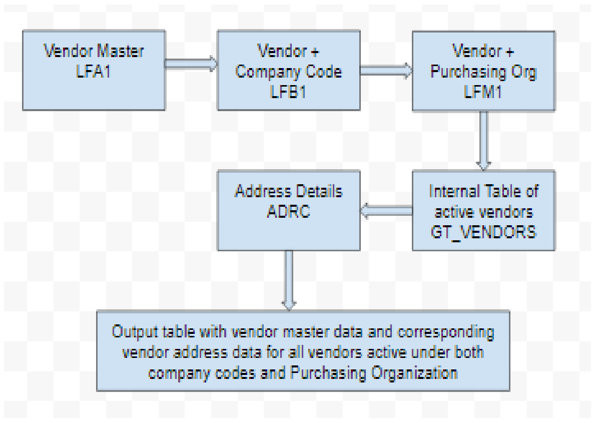


Because of C2D paradigm, the delay caused due to the latency of bringing large volumes of data to presentation layer is removed drastically resulting in high performance even with very large datasets.

To better understand this let me proceed using a basic scenario that any ABAP developer can easily relate to:

Example Scenario: An ALV report that returns the master data of **"ALL"** vendors and their addresses for vendors that are active, not marked for central deletion, not marked for deletion under **"ALL"** company code, not marked for deletion under **"ALL"** purchasing organization.

In this scenario, the performance would suffer because of fetching data for all vendors, for all company codes & for all Purchasing Organization. The resulting report will require a background run and the traditional ABAP report flow ,in this case, would fetch data as follows:


Vendor Master LFA1
Vendor + Company Code LFB1
Vendor + Purchasing Org LFM1
Internal Table of active vendors GT_VENDORS
Address Details ADRC
Output table with vendor master data and corresponding vendor address data for all vendors active under both company codes and Purchasing Organization

Here the presentation layer would interact with a couple of times if no joins are used under select statement. Moreover using joins to fetch data from these tables would also be very slow because of large volumes of vendor data in the system. So how must I improve the performance here?

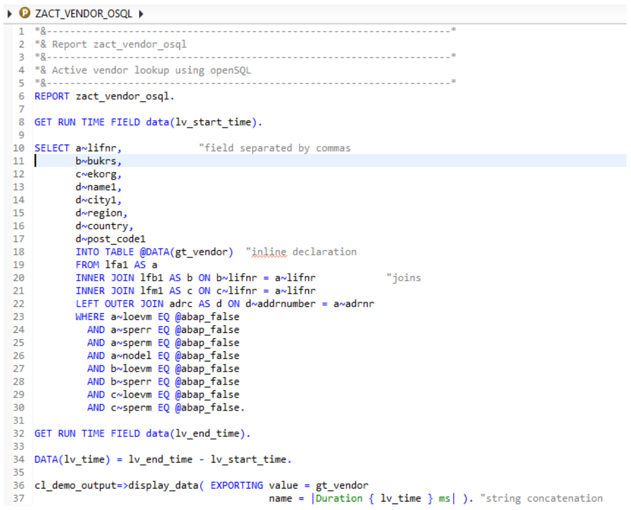
Answer!- Code PushDown using
OpenSQL programming
CDS Views
ABAP Managed Database Procedures

Let us visit each of the above features that were introduced with HANA DB.

1.OpenSQL Programming

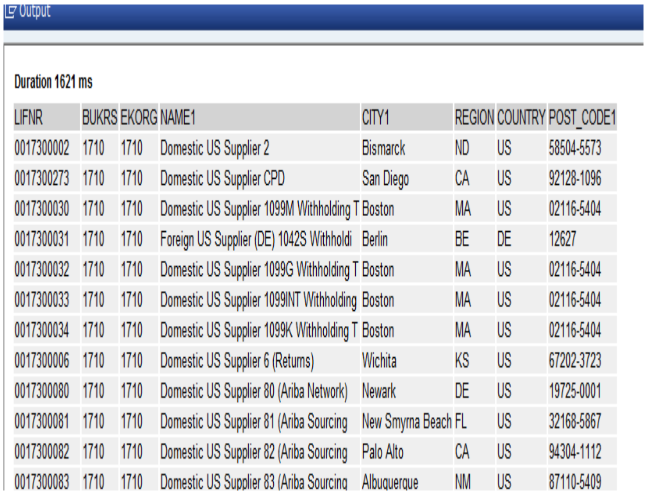
With OpenSQL programming, you can write openSQL syntax in your ABAP code. The syntax for OpenSQL differs from that of ABAP, for example, the fields in select statement are comma separated, all the host variables are escaped using '@' sign, the concatenation can be done in a single statement using '| |' and so on and so forth.

The above scenario will be written as follows:


ZACT_VENDOR_OSQL

```
*&---------------------------------------------------------------------*
*& Report zact_vendor_osql
*&---------------------------------------------------------------------*
*& Active vendor lookup using openSQL
*&---------------------------------------------------------------------*
REPORT zact_vendor_osql.

GET RUN TIME FIELD data(lv_start_time).

SELECT a~lifnr,                "field separated by commas
       b~bukrs,
       c~ekorg,
       d~name1,
       d~city1,
       d~region,
       d~country,
       d~post_code1
       INTO TABLE @DATA(gt_vendor)  "inline declaration
       FROM lfa1 AS a
       INNER JOIN lfb1 AS b ON b~lifnr = a~lifnr            "joins
       INNER JOIN lfm1 AS c ON c~lifnr = a~lifnr
       LEFT OUTER JOIN adrc AS d ON d~addrnumber = a~adrnr
       WHERE a~loevm EQ @abap_false
         AND a~sperr EQ @abap_false
         AND a~sperm EQ @abap_false
         AND a~nodel EQ @abap_false
         AND b~loevm EQ @abap_false
         AND b~sperr EQ @abap_false
         AND c~loevm EQ @abap_false
         AND c~sperm EQ @abap_false.

GET RUN TIME FIELD data(lv_end_time).

DATA(lv_time) = lv_end_time - lv_start_time.

cl_demo_output=>display_data( EXPORTING value = gt_vendor
                                        name = |Duration { lv_time } ms| ). "string concatenation
```

Result:


Output

```
Duration 1621 ms

LIFNR      BUKRS EKORG NAME1                             CITY1           REGION COUNTRY POST_CODE1
0017300002 1710  1710  Domestic US Supplier 2            Bismarck        ND     US      58504-5573
0017300273 1710  1710  Domestic US Supplier CPD          San Diego       CA     US      92128-1096
0017300030 1710  1710  Domestic US Supplier 1099M Withholding T Boston   MA     US      02116-5404
0017300031 1710  1710  Foreign US Supplier (DE) 1042S Withholdi Berlin  BE     DE      12627
0017300032 1710  1710  Domestic US Supplier 1099G Withholding T Boston   MA     US      02116-5404
0017300033 1710  1710  Domestic US Supplier 1099INT Withholding Boston   MA     US      02116-5404
0017300034 1710  1710  Domestic US Supplier 1099K Withholding T Boston   MA     US      02116-5404
0017300006 1710  1710  Domestic US Supplier 6 (Returns)  Wichita         KS     US      67202-3723
0017300080 1710  1710  Domestic US Supplier 80 (Ariba Network) Newark    DE     US      19725-0001
0017300081 1710  1710  Domestic US Supplier 81 (Ariba Sourcing New Smyrna Beach FL US   32168-5867
0017300082 1710  1710  Domestic US Supplier 82 (Ariba Sourcing Palo Alto CA     US      94304-1112
0017300083 1710  1710  Domestic US Supplier 83 (Ariba Sourcing Albuquerque NM   US      87110-5409
```

Here you can see that report ran for 1621 ms and returned us the desired results.
This is a very basic example that I took but in real time scenarios where you may be doing some aggregations, or you may be to translating some data during selection, or you may be grouping your result set based on some fields then the OpenSQL really does magic.

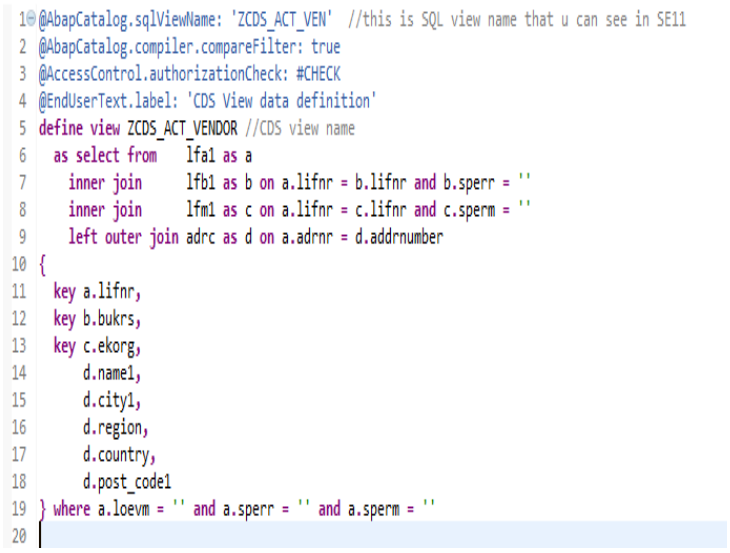
SAP has introduced a large number of syntax that can be utilized in the code to improve its performance. To start with you can find very descriptive examples and code snippets in the ABAP glossary itself.

SAP has introduced a large number of syntax that can be utilized in code to improve its performance. To start with you can find very descriptive examples and code snippets in the ABAP glossary itself.

2. Core Data Services (CDS) Views

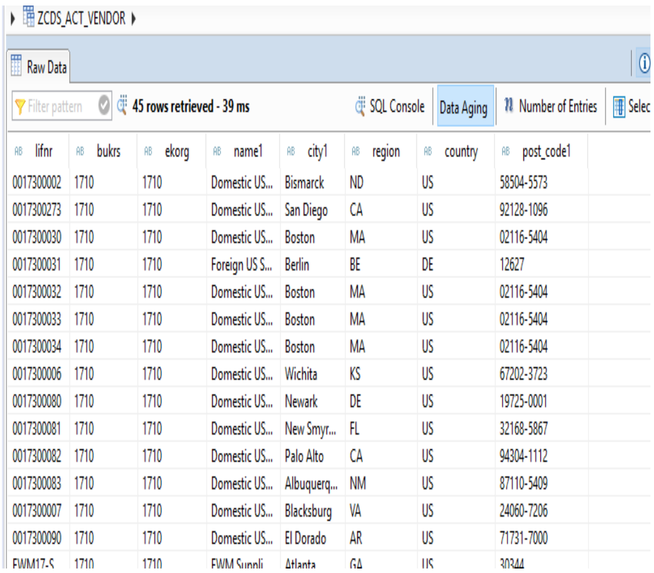
SAP introduced a new data modeling infrastructure known as core data services or CDS. With CDS, data models are defined and consumed on database server rather than on application server. As a result, the table result view is created at the database level. CDS are completely compatible with openSQL and can be written using ABAP development tools like Eclipse Oxygen. These can be consumed by reports and AMDPs as well.

The above code will be created as a data definition in Eclipse and defined as follows:



```
@AbapCatalog.sqlViewName: 'ZCDS_ACT_VEN'  //this is SQL view name that u can see in SE11
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'CDS View data definition'
define view ZCDS_ACT_VENDOR //CDS view name
  as select from    lfa1 as a
    inner join      lfb1 as b on a.lifnr = b.lifnr and b.sperr = ''
    inner join      lfm1 as c on a.lifnr = c.lifnr and c.sperm = ''
    left outer join adrc as d on a.adrnr = d.addrnumber
{
  key a.lifnr,
  key b.bukrs,
  key c.ekorg,
      d.name1,
      d.city1,
      d.region,
      d.country,
      d.post_code1
} where a.loevm = '' and a.sperr = '' and a.sperm = ''

```

Result:



ZCDS_ACT_VENDOR

Raw Data

Filter pattern | 45 rows retrieved - 39 ms | SQL Console | Data Aging | Number of Entries | Selec

lifnr	bukrs	ekorg	name1	city1	region	country	post_code1
0017300002	1710	1710	Domestic US...	Bismarck	ND	US	58504-5573
0017300273	1710	1710	Domestic US...	San Diego	CA	US	92128-1096
0017300030	1710	1710	Domestic US...	Boston	MA	US	02116-5404
0017300031	1710	1710	Foreign US S...	Berlin	BE	DE	12627
0017300032	1710	1710	Domestic US...	Boston	MA	US	02116-5404
0017300033	1710	1710	Domestic US...	Boston	MA	US	02116-5404
0017300034	1710	1710	Domestic US...	Boston	MA	US	02116-5404
0017300006	1710	1710	Domestic US...	Wichita	KS	US	67202-3723
0017300080	1710	1710	Domestic US...	Newark	DE	US	19725-0001
0017300081	1710	1710	Domestic US...	New Smyr...	FL	US	32168-5867
0017300082	1710	1710	Domestic US...	Palo Alto	CA	US	94304-1112
0017300083	1710	1710	Domestic US...	Albuquerq...	NM	US	87110-5409
0017300007	1710	1710	Domestic US...	Blacksburg	VA	US	24060-7206
0017300090	1710	1710	Domestic US...	El Dorado	AR	US	71731-7000
FWM17-S	1710	1710	FWM Suppli	Atlanta	GA	US	30344

The CDS view returned the result in 39ms. Awesome? Yes, it is.

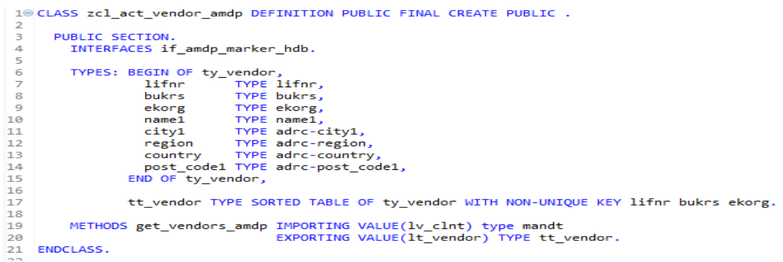
Now CDS views could also be created with parameters or with associations. You may choose to create a CDS with parameters if you have a fixed result set and some input parameters to pass.

You could also create a CDS with the association for a similar scenario if you have many tables to address in the view and if you want to keep the result set flexible.

3. ABAP Managed Database Procedures (AMDP)

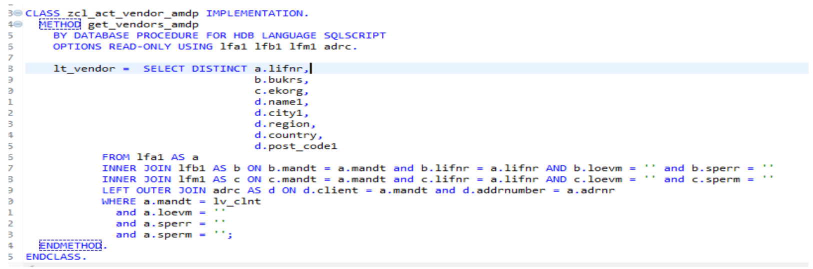
AMDPs, as the name says, are database procedures that run on the database directly and are written directly in ABAP. AMDPs are written using AMDP classes. Below is an example using the above scenario of how to create an AMDP class. The interface “IF_AMDP_MARKER_HDB” distinguishes an AMDP class from other classes.

Class definition:



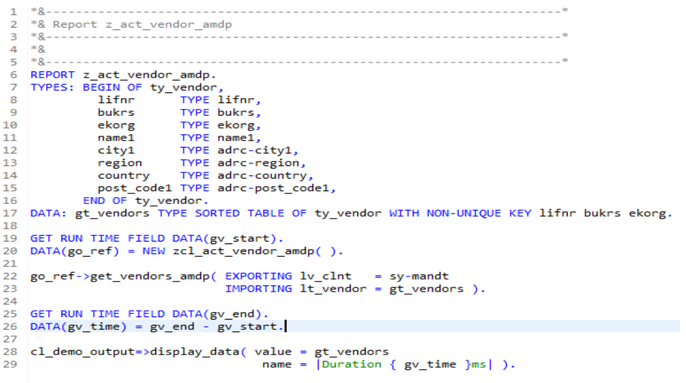
```
CLASS zcl_act_vendor_amdp DEFINITION PUBLIC FINAL CREATE PUBLIC .

  PUBLIC SECTION.
    INTERFACES if_amdp_marker_hdb.

    TYPES: BEGIN OF ty_vendor,
             lifnr      TYPE lifnr,
             bukrs      TYPE bukrs,
             ekorg      TYPE ekorg,
             name1      TYPE name1,
             city1      TYPE adrc-city1,
             region     TYPE adrc-region,
             country    TYPE adrc-country,
             post_code1 TYPE adrc-post_code1,
           END OF ty_vendor,

           tt_vendor TYPE SORTED TABLE OF ty_vendor WITH NON-UNIQUE KEY lifnr bukrs ekorg.

    METHODS get_vendors_amdp IMPORTING VALUE(lv_clnt) type mandt
                             EXPORTING VALUE(lt_vendor) TYPE tt_vendor.
ENDCLASS.
```

Similarly, an AMDP class implementation will have methods defined with a syntax “BY DATABASE PROCEDURE FOR <database> LANGUAGE <language>. In our case database will be HDB (HANA DB) and language will always be SQLSCRIPT.



```
CLASS zcl_act_vendor_amdp IMPLEMENTATION.
  METHOD get_vendors_amdp
    BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT
    OPTIONS READ-ONLY USING lfa1 lfb1 lfm1 adrc.

    lt_vendor =  SELECT DISTINCT a.lifnr,
                                 b.bukrs,
                                 c.ekorg,
                                 d.name1,
                                 d.city1,
                                 d.region,
                                 d.country,
                                 d.post_code1
           FROM lfa1 AS a
           INNER JOIN lfb1 AS b ON b.mandt = a.mandt and b.lifnr = a.lifnr AND b.loevm = '' and b.sperr = ''
           INNER JOIN lfm1 AS c ON c.mandt = a.mandt and c.lifnr = a.lifnr AND c.loevm = '' and c.sperm = ''
           LEFT OUTER JOIN adrc AS d ON d.client = a.mandt and d.addrnumber = a.adrnr
           WHERE a.mandt = lv_clnt
             and a.loevm = ''
             and a.sperr = ''
             and a.sperm = '';
  ENDMETHOD.
ENDCLASS.
```

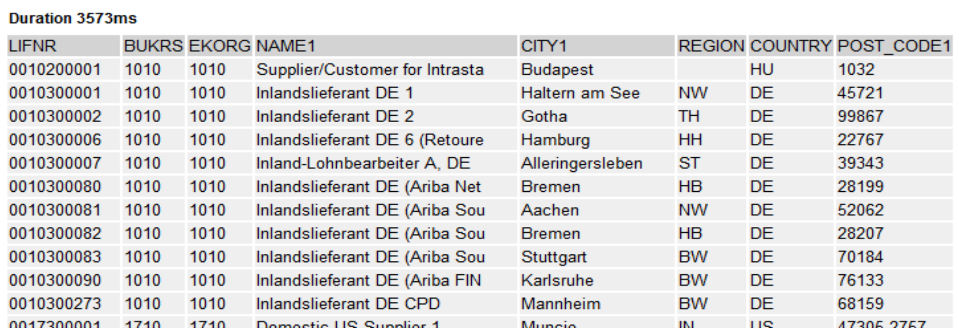
This AMDP class can then be consumed in an ABAP program to achieve the code push down functionality.



```
*&---------------------------------------------------------------------*
*& Report z_act_vendor_amdp
*&---------------------------------------------------------------------*
*&
*&---------------------------------------------------------------------*
REPORT z_act_vendor_amdp.
TYPES: BEGIN OF ty_vendor,
         lifnr      TYPE lifnr,
         bukrs      TYPE bukrs,
         ekorg      TYPE ekorg,
         name1      TYPE name1,
         city1      TYPE adrc-city1,
         region     TYPE adrc-region,
         country    TYPE adrc-country,
         post_code1 TYPE adrc-post_code1,
       END OF ty_vendor.
DATA: gt_vendors TYPE SORTED TABLE OF ty_vendor WITH NON-UNIQUE KEY lifnr bukrs ekorg.

GET RUN TIME FIELD DATA(gv_start).
DATA(go_ref) = NEW zcl_act_vendor_amdp( ).

go_ref->get_vendors_amdp( EXPORTING lv_clnt    = sy-mandt
                          IMPORTING lt_vendor = gt_vendors ).

GET RUN TIME FIELD DATA(gv_end).
DATA(gv_time) = gv_end - gv_start.

cl_demo_output=>display_data( value = gt_vendors
                              name = |Duration { gv_time }ms| ).
```

This AMDP class can then be consumed in an ABAP program to achieve the code push down functionality.

Result:



Duration 3573ms

LIFNR	BUKRS	EKORG	NAME1	CITY1	REGION	COUNTRY	POST_CODE1
0010200001	1010	1010	Supplier/Customer for Intrasta	Budapest		HU	1032
0010300001	1010	1010	Inlandslieferant DE 1	Haltern am See	NW	DE	45721
0010300002	1010	1010	Inlandslieferant DE 2	Gotha	TH	DE	99867
0010300006	1010	1010	Inlandslieferant DE 6 (Retoure	Hamburg	HH	DE	22767
0010300007	1010	1010	Inland-Lohnbearbeiter A, DE	Alleringersleben	ST	DE	39343
0010300080	1010	1010	Inlandslieferant DE (Ariba Net	Bremen	HB	DE	28199
0010300081	1010	1010	Inlandslieferant DE (Ariba Sou	Aachen	NW	DE	52062
0010300082	1010	1010	Inlandslieferant DE (Ariba Sou	Bremen	HB	DE	28207
0010300083	1010	1010	Inlandslieferant DE (Ariba Sou	Stuttgart	BW	DE	70184
0010300090	1010	1010	Inlandslieferant DE (Ariba FIN	Karlsruhe	BW	DE	76133
0010300273	1010	1010	Inlandslieferant DE CPD	Mannheim	BW	DE	68159
0017300001	1710	1710	Domestic US Supplier 1	Muncie	IN	US	47306-2757

What to choose OpenSQL or CDS or AMDP?

A question that would arise in any developers mind would be how to make a choice amongst the three programming techniques.

In the above example, you can see that the performance was CDS > OpenSQL > AMDP. Does that mean for the above scenario the best choice is to create a CDS? Not exactly!

If I do not reuse the CDS view then openSQL could be an equally effective choice.

Also, note that CDS views and AMDP can only be created using ABAP Development Tools like Eclipse Oxygen. Refer the following link to understand how to get eclipse on your system:

https://tools.hana.ondemand.com/#abap

There are no rules that can be adhered to when choosing from the above three programming techniques. It completely depends on the requirement and on what and how data needs to be handled. However, the points below can give an idea on how to proceed to make the most productive choice.

Choose Open SQL when:

1.The table selection is program specific and will not be reused

2.When you do not have an ABAP Development Tool to create CDS or AMDP. The two can be consumed in GUI but cannot be created in GUI.

3.When the data in question does not involve intensive calculations and can be managed easily by OpenSQL.

4.When you have a tricky selection screen with a lot of select options that will be passed as single values too**.**

Choose CDS views when:

1.The view can be reused among other views or programs.

2.When a large volume of data is involved from various data sources.

3.When you have good knowledge on how to write annotations to enhance your CDS view.

4.Only single result set is required.

Choose AMDPs When:

1.You are affluent with SQL scripting because your entire code will be written in SQL script and the compiler fails in determining the runtime SQL script errors like divide by zero.

2.When you have to handle cross client data because AMDP does not do client handling on its own.

3.When multiple result sets are required.

This blog was to give you a kick start on what HANA has to offer and what you can do with the new techniques. My advice to any beginner would be to get your hands on a system and just try.

For all enquiries please contact at : corp@accretesol.com , Tel : +1(877)-849-5838
Visit us at : www.accrete-solutions.com

USA
Head Office
3350 Scott Blvd, Bldg 34
Santa Clara, CA 95054

South Africa
609 Lanseria Corporate Estate,
Falcon Lane, Lanseria,
Gauteng

Chile
Galvarino Gallardo 1638,
Providencia,
Santiago

India
Development Centre
102A, HARTRON, Electronics City,
Gurgaon