

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.

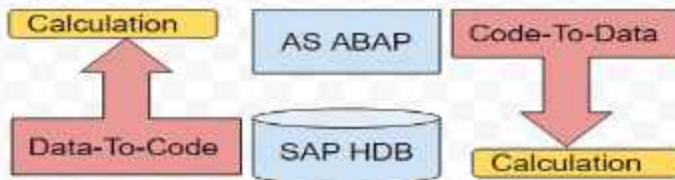
Let us begin!

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

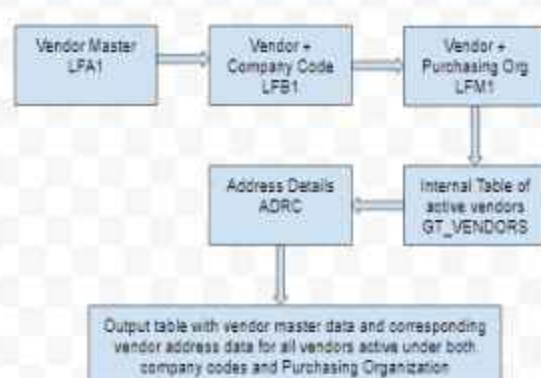


Because of C2D paradigm, the delay caused due to the latency of bringing large volumes of data to presentation layer is removed drastically resulting in high performance even with very large datasets.

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:



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:

```
1  *
2  * @author: sap
3  * @title: OpenSQL programming
4  * @description: OpenSQL programming
5  * @keywords: OpenSQL programming
6  * @language: ABAP
7  *
8  *
9  *
10 *
11 *
12 *
13 *
14 *
15 *
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *
24 *
25 *
26 *
27 *
28 *
29 *
30 *
31 *
32 *
33 *
34 *
35 *
36 *
37 *
38 *
39 *
40 *
41 *
42 *
43 *
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *
57 *
58 *
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *
121 *
122 *
123 *
124 *
125 *
126 *
127 *
128 *
129 *
130 *
131 *
132 *
133 *
134 *
135 *
136 *
137 *
138 *
139 *
140 *
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 *
151 *
152 *
153 *
154 *
155 *
156 *
157 *
158 *
159 *
160 *
161 *
162 *
163 *
164 *
165 *
166 *
167 *
168 *
169 *
170 *
171 *
172 *
173 *
174 *
175 *
176 *
177 *
178 *
179 *
180 *
181 *
182 *
183 *
184 *
185 *
186 *
187 *
188 *
189 *
190 *
191 *
192 *
193 *
194 *
195 *
196 *
197 *
198 *
199 *
200 *
201 *
202 *
203 *
204 *
205 *
206 *
207 *
208 *
209 *
210 *
211 *
212 *
213 *
214 *
215 *
216 *
217 *
218 *
219 *
220 *
221 *
222 *
223 *
224 *
225 *
226 *
227 *
228 *
229 *
230 *
231 *
232 *
233 *
234 *
235 *
236 *
237 *
238 *
239 *
240 *
241 *
242 *
243 *
244 *
245 *
246 *
247 *
248 *
249 *
250 *
251 *
252 *
253 *
254 *
255 *
256 *
257 *
258 *
259 *
260 *
261 *
262 *
263 *
264 *
265 *
266 *
267 *
268 *
269 *
270 *
271 *
272 *
273 *
274 *
275 *
276 *
277 *
278 *
279 *
280 *
281 *
282 *
283 *
284 *
285 *
286 *
287 *
288 *
289 *
290 *
291 *
292 *
293 *
294 *
295 *
296 *
297 *
298 *
299 *
300 *
301 *
302 *
303 *
304 *
305 *
306 *
307 *
308 *
309 *
310 *
311 *
312 *
313 *
314 *
315 *
316 *
317 *
318 *
319 *
320 *
321 *
322 *
323 *
324 *
325 *
326 *
327 *
328 *
329 *
330 *
331 *
332 *
333 *
334 *
335 *
336 *
337 *
338 *
339 *
340 *
341 *
342 *
343 *
344 *
345 *
346 *
347 *
348 *
349 *
350 *
351 *
352 *
353 *
354 *
355 *
356 *
357 *
358 *
359 *
360 *
361 *
362 *
363 *
364 *
365 *
366 *
367 *
368 *
369 *
370 *
371 *
372 *
373 *
374 *
375 *
376 *
377 *
378 *
379 *
380 *
381 *
382 *
383 *
384 *
385 *
386 *
387 *
388 *
389 *
390 *
391 *
392 *
393 *
394 *
395 *
396 *
397 *
398 *
399 *
400 *
401 *
402 *
403 *
404 *
405 *
406 *
407 *
408 *
409 *
410 *
411 *
412 *
413 *
414 *
415 *
416 *
417 *
418 *
419 *
420 *
421 *
422 *
423 *
424 *
425 *
426 *
427 *
428 *
429 *
430 *
431 *
432 *
433 *
434 *
435 *
436 *
437 *
438 *
439 *
440 *
441 *
442 *
443 *
444 *
445 *
446 *
447 *
448 *
449 *
450 *
451 *
452 *
453 *
454 *
455 *
456 *
457 *
458 *
459 *
460 *
461 *
462 *
463 *
464 *
465 *
466 *
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 *
476 *
477 *
478 *
479 *
480 *
481 *
482 *
483 *
484 *
485 *
486 *
487 *
488 *
489 *
490 *
491 *
492 *
493 *
494 *
495 *
496 *
497 *
498 *
499 *
500 *
501 *
502 *
503 *
504 *
505 *
506 *
507 *
508 *
509 *
510 *
511 *
512 *
513 *
514 *
515 *
516 *
517 *
518 *
519 *
520 *
521 *
522 *
523 *
524 *
525 *
526 *
527 *
528 *
529 *
530 *
531 *
532 *
533 *
534 *
535 *
536 *
537 *
538 *
539 *
540 *
541 *
542 *
543 *
544 *
545 *
546 *
547 *
548 *
549 *
550 *
551 *
552 *
553 *
554 *
555 *
556 *
557 *
558 *
559 *
560 *
561 *
562 *
563 *
564 *
565 *
566 *
567 *
568 *
569 *
570 *
571 *
572 *
573 *
574 *
575 *
576 *
577 *
578 *
579 *
580 *
581 *
582 *
583 *
584 *
585 *
586 *
587 *
588 *
589 *
590 *
591 *
592 *
593 *
594 *
595 *
596 *
597 *
598 *
599 *
600 *
601 *
602 *
603 *
604 *
605 *
606 *
607 *
608 *
609 *
610 *
611 *
612 *
613 *
614 *
615 *
616 *
617 *
618 *
619 *
620 *
621 *
622 *
623 *
624 *
625 *
626 *
627 *
628 *
629 *
630 *
631 *
632 *
633 *
634 *
635 *
636 *
637 *
638 *
639 *
640 *
641 *
642 *
643 *
644 *
645 *
646 *
647 *
648 *
649 *
650 *
651 *
652 *
653 *
654 *
655 *
656 *
657 *
658 *
659 *
660 *
661 *
662 *
663 *
664 *
665 *
666 *
667 *
668 *
669 *
670 *
671 *
672 *
673 *
674 *
675 *
676 *
677 *
678 *
679 *
680 *
681 *
682 *
683 *
684 *
685 *
686 *
687 *
688 *
689 *
690 *
691 *
692 *
693 *
694 *
695 *
696 *
697 *
698 *
699 *
700 *
701 *
702 *
703 *
704 *
705 *
706 *
707 *
708 *
709 *
710 *
711 *
712 *
713 *
714 *
715 *
716 *
717 *
718 *
719 *
720 *
721 *
722 *
723 *
724 *
725 *
726 *
727 *
728 *
729 *
730 *
731 *
732 *
733 *
734 *
735 *
736 *
737 *
738 *
739 *
740 *
741 *
742 *
743 *
744 *
745 *
746 *
747 *
748 *
749 *
750 *
751 *
752 *
753 *
754 *
755 *
756 *
757 *
758 *
759 *
760 *
761 *
762 *
763 *
764 *
765 *
766 *
767 *
768 *
769 *
770 *
771 *
772 *
773 *
774 *
775 *
776 *
777 *
778 *
779 *
780 *
781 *
782 *
783 *
784 *
785 *
786 *
787 *
788 *
789 *
790 *
791 *
792 *
793 *
794 *
795 *
796 *
797 *
798 *
799 *
800 *
801 *
802 *
803 *
804 *
805 *
806 *
807 *
808 *
809 *
810 *
811 *
812 *
813 *
814 *
815 *
816 *
817 *
818 *
819 *
820 *
821 *
822 *
823 *
824 *
825 *
826 *
827 *
828 *
829 *
830 *
831 *
832 *
833 *
834 *
835 *
836 *
837 *
838 *
839 *
840 *
841 *
842 *
843 *
844 *
845 *
846 *
847 *
848 *
849 *
850 *
851 *
852 *
853 *
854 *
855 *
856 *
857 *
858 *
859 *
860 *
861 *
862 *
863 *
864 *
865 *
866 *
867 *
868 *
869 *
870 *
871 *
872 *
873 *
874 *
875 *
876 *
877 *
878 *
879 *
880 *
881 *
882 *
883 *
884 *
885 *
886 *
887 *
888 *
889 *
890 *
891 *
892 *
893 *
894 *
895 *
896 *
897 *
898 *
899 *
900 *
901 *
902 *
903 *
904 *
905 *
906 *
907 *
908 *
909 *
910 *
911 *
912 *
913 *
914 *
915 *
916 *
917 *
918 *
919 *
920 *
921 *
922 *
923 *
924 *
925 *
926 *
927 *
928 *
929 *
930 *
931 *
932 *
933 *
934 *
935 *
936 *
937 *
938 *
939 *
940 *
941 *
942 *
943 *
944 *
945 *
946 *
947 *
948 *
949 *
950 *
951 *
952 *
953 *
954 *
955 *
956 *
957 *
958 *
959 *
960 *
961 *
962 *
963 *
964 *
965 *
966 *
967 *
968 *
969 *
970 *
971 *
972 *
973 *
974 *
975 *
976 *
977 *
978 *
979 *
980 *
981 *
982 *
983 *
984 *
985 *
986 *
987 *
988 *
989 *
990 *
991 *
992 *
993 *
994 *
995 *
996 *
997 *
998 *
999 *
1000 *
```

Result:

LFNR	BUKRS	ENCRD	NAME1	CITY1	REGION	COUNTRY	POST_CODE1
001730002	1710	1710	Domestic US Supplier 2	Barnack	ND	US	58504-5573
001730023	1710	1710	Domestic US Supplier CPD	San Diego	CA	US	92123-1096
001730030	1710	1710	Domestic US Supplier 12981 Wintheking T	Boston	MA	US	02115-5404
001730031	1710	1710	Foreign US Supplier DE 12425 Wintheking	Bahn	DE	DE	42627
001730032	1710	1710	Domestic US Supplier 12982 Wintheking T	Boston	MA	US	02115-5404
001730033	1710	1710	Domestic US Supplier 12983 Wintheking Boston		MA	US	02115-5404
001730034	1710	1710	Domestic US Supplier 12984 Wintheking T	Boston	MA	US	02115-5404
001730035	1710	1710	Domestic US Supplier 6 (Return)	Wierka	KS	US	67262-3723
001730080	1710	1710	Domestic US Supplier 8 (A-Isa Network)	Newark	DE	US	19725-0001
001730081	1710	1710	Domestic US Supplier 81 (A-Isa Sourcing)	New Smyrna Beach	FL	US	32103-0067
001730082	1710	1710	Domestic US Supplier 82 (A-Isa Sourcing)	Palo Alto	CA	US	94304-1112
001730083	1710	1710	Domestic US Supplier 83 (A-Isa Sourcing)	Albuquerque	NM	US	87110-5403

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:

```
1 @#oapCatalog.sqlViewName: 'ZCDS_ACT_VEN' //this is SQL view name that u can see in SE11
2 @#oapCatalog.compiler.compareFilter: true
3 @accessControl.authorizationCheck: #CHECK
4 @EndUserText.label: 'CDS View data definition'
5 define view ZCDS_ACT_VENDOR //CDS view name
6 as select from Ifal as a
7 inner join Ifb1 as b on a.lifnr = b.lifnr and b.sperr = ''
8 inner join Ifnd as c on a.lifnr = c.lifnr and c.sperr = ''
9 left outer join adrc as d on a.adrnr = d.adrnr
10 {
11 key a.lifnr,
12 key b.bukrs,
13 key c.ekorg,
14 d.name1,
15 d.city1,
16 d.region,
17 d.country,
18 d.post_codel
19 } where a.loevn = '' and a.sperr = '' and a.sperr = ''
20
```

Result:

bukrs	ekorg	name1	city1	region	country	post_codel
001730002	1710	Domestic US...	Bismarck	ND	US	58004-5570
001730003	1710	Domestic US...	San Diego	CA	US	92128-1096
001730004	1710	Domestic US...	Boston	MA	US	02115-5404
001730005	1710	Foreign US S...	Berlin	BE	DE	12027
001730006	1710	Domestic US...	Boston	MA	US	02115-5404
001730007	1710	Domestic US...	Boston	MA	US	02115-5404
001730008	1710	Domestic US...	Boston	MA	US	02115-5404
001730009	1710	Domestic US...	Wichita	KS	US	67202-1723
001730010	1710	Domestic US...	Newark	DE	US	19725-0001
001730011	1710	Domestic US...	New Smyrna	FL	US	32088-5887
001730012	1710	Domestic US...	Palo Alto	CA	US	94304-1112
001730013	1710	Domestic US...	Albuquerque	NM	US	87110-5409
001730014	1710	Domestic US...	Blacksburg	VA	US	24003-7256
001730015	1710	Domestic US...	El Dorado	AR	US	71701-7000
001730016	1710	Domestic US...	El Dorado	AR	US	71701-7000

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:

```

1 CLASS zcl_act_vendor_amlp DEFINITION PUBLIC FINAL CREATE PUBLIC .
2
3 PUBLIC SECTION.
4   INTERFACES IF_amlp_marker_hdb.
5
6   TYPES: BEGIN OF ty_vendor,
7           lifnr     TYPE lifnr,
8           buksr     TYPE buksr,
9           ekorg     TYPE ekorg,
10          name1     TYPE name1,
11          city1     TYPE adrc-city1,
12          region    TYPE adrc-region,
13          country   TYPE adrc-country,
14          post_code1 TYPE adrc-post_code1,
15        END OF ty_vendor.
16
17   ty_vendor TYPE SORTED TABLE OF ty_vendor WITH NON-UNIQUE KEY lifnr buksr ekorg.
18
19   METHODS get_vendors_amlp EXPORTING VALUE(iv_cint) TYPE cint
20                        EXPORTING VALUE(it_vendor) TYPE ty_vendor.
21
22 ENDMETHOD.

```

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.

```

1 CLASS zcl_act_vendor_amlp IMPLEMENTATION.
2
3   METHOD get_vendors_amlp
4     BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT
5     OPTIMIZE READ-ONLY MODE IFAL IFAL IFAL IFAL.
6
7     zt_vendor = SELECT DISTINCT a.lifnr
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

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

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500

```

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	46721
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	Alleingeraleben	ST	DE	39343
0010300080	1010	1010	Inlandslieferant DE (Anba Net	Bremen	HB	DE	28199
0010300081	1010	1010	Inlandslieferant DE (Anba Sou	Aachen	NW	DE	52062
0010300082	1010	1010	Inlandslieferant DE (Anba Sou	Bremen	HB	DE	28207
0010300083	1010	1010	Inlandslieferant DE (Anba Sou	Stuttgart	BW	DE	70184
0010300090	1010	1010	Inlandslieferant DE (Anba FIN	Karlsruhe	BW	DE	76133
0010300273	1010	1010	Inlandslieferant DE CPD	Mannheim	BW	DE	68159
0017200004	4740	4740	Flussufer 118 Schloß 4	Münster	BI	DE	47204-7747

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