



SAP HANA SQL scripting optimization: the CE Functions



By Matthieu Munch | June 12, 2015 | Development & Performance | 3 Comments



Post view(s) : 10,913

Search...



In SAP HANA, you have two possibilities to create the Calculated Views:

Using the graphical method

Using the scripting method with CE functions

In this blog, I will demonstrate that CE Functions can improve performances from a Calculated View.

First, I will give you some general information regarding the CE Functions. After that, I will show you the

CATEGORIES

[Application integration & Middleware](#)

[AWS](#)

[Big Data](#)

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#)

[Read More](#)



CE Functions

The CE Functions encapsulate data-transformation functionalities. They constitute an alternative to using SQL statements as their logic is directly implemented in the Sap HANA CALC engine. Direct use of the CALC engine allows implementers to influence the execution of a procedure or a query which, in some cases, is more efficient.

In the table below, you can find the list of the CE Functions and their use cases:

[Development & Performance](#)

[DevOps](#)

[Docker](#)

[Enterprise content management](#)

[Hardware & Storage](#)

[Kubernetes](#)

[MariaDB](#)

[NoSQL](#)

[Operation systems](#)

[Oracle](#)

[Postgres](#)

[Security](#)

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#) [Read More](#)

WHERE HAVING	SELECT A, B, C SUM(D) From "ANALYTICAL_VIEW" WHERE B = 'Value' AND C = 'Value'	var_tab = CE_COLUMN_TABLE("COLUMN_TABLE"); CE_PROJECTION(:var_tab,[A,B,C],"B" = "value" AND "C" = "value");
GROUP BY	SELECT A, B, C SUM(D) From "COLUMN_TABLE" GROUP BY A, B, C	var_tab = CE_COLUMN_TABLE("COLUMN_TABLE"); CE_AGGREGATION(:var_tab,SUM(D), [A,B,C]);
INNER JOIN	SELECT A, B, Y, SUM(D) FROM "COLTAB1" INNER JOIN "COLTAB2" WHERE "COLTAB1"."KEY1" AND "COLTAB1"."KEY2" AND "COLTAB1"."KEY2" = "COLTAB2"."KEY2"	CE_JOIN("COLTAB1", "COLTAB2",[KEY1,KEY2],[A,B,C,D])
LEFT OUTER JOIN	SELECT A, B, Y, SUM(D) FROM "COLTAB1" LEFT OUTER JOIN "COLTAB2" WHERE "COLTAB1"."KEY1" AND "COLTAB1"."KEY2" AND "COLTAB1"."KEY2" = "COLTAB2"."KEY2"	CE_LEFT_OUTER_JOIN("COLTAB1", "COLTAB2",[KEY1,KEY2], [A,B,Y,D])
SQL Expressions	SELECT A, B, C, SUBSTRING(D,2,5) FROM "COLUMN_TABLE"	var_tab = CE_COLUMN_TABLE("COLUMN_TABLE"); CE_PROJECTION(:var_tab,["A", "B", "C", CE_CALC('midstr("D",2,5)];
UNION ALL	VAR_TAB1 = SELECT A, B, C, D FROM "COLUMN_TABLE1"; VAR_TAB2 = SELECT A, B, C, D FROM "COLUMN_TABLE2"; SELECT * FROM: VAR_TAB1 UNION ALL SELECT * FROM :VAR_TAB2	VAR_TAB1 = CE_COLUMN_TABLE("COLUMN_TABLE1",[A,B,C,D]); VAR_TAB2 = CE_COLUMN_TABLE("COLUMN_TABLE2",[A,B,C,D]); CE_UNION_ALL(:var_tab1;var_tab2);

Starting an Oracle Database
when a first connection comes in

Cleanup a failed Oracle XE
installation on Linux Mint

Setup Oracle XE on Linux Mint –
a funny exercise

AWS RDS for PostgreSQL – 3 –
Creating the RDS PostgreSQL
instance

TAG CLOUD

12.2 Alfresco AlwaysOn Availability
groups AWS Cloud Cluster D2
database Data Guard DBA Docker
Documentum
enterprisedb High availability

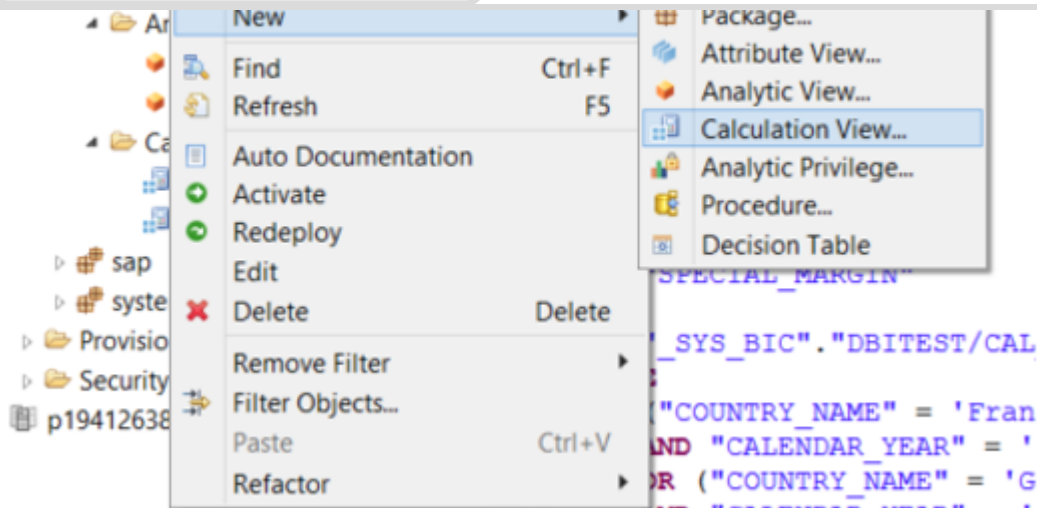
How to create a Calculation View

As I say at the beginning, you have two methods to create a Calculation View in Sap HANA:

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#)

[Read More](#)



[Oracle OpenWorld](#)

[Performance](#)

[PostgreSQL](#) [PowerShell](#)

[RMAN Security SQL](#) [SQL](#)

[Server](#) [SQL Server 2008 SQL](#)

[Server 2012 SQL Server](#)

[2014 SQL Server 2016](#)

[Troubleshooting](#)

BLOG ROLL

[Florian Haas' blog](#)

[Oracle Scratchpad - Jonathan Lewis' blog](#)

[The Tom Kyte Blog \(Ask Tom\)](#)

[Blog of Adar-Consult](#)

[The SQL Blog](#)

Select the "Graphical" type

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#)

[Read More](#)

details

Name:* Label: Package:* View Type: ▾☐ Copy From: Subtype: ▾

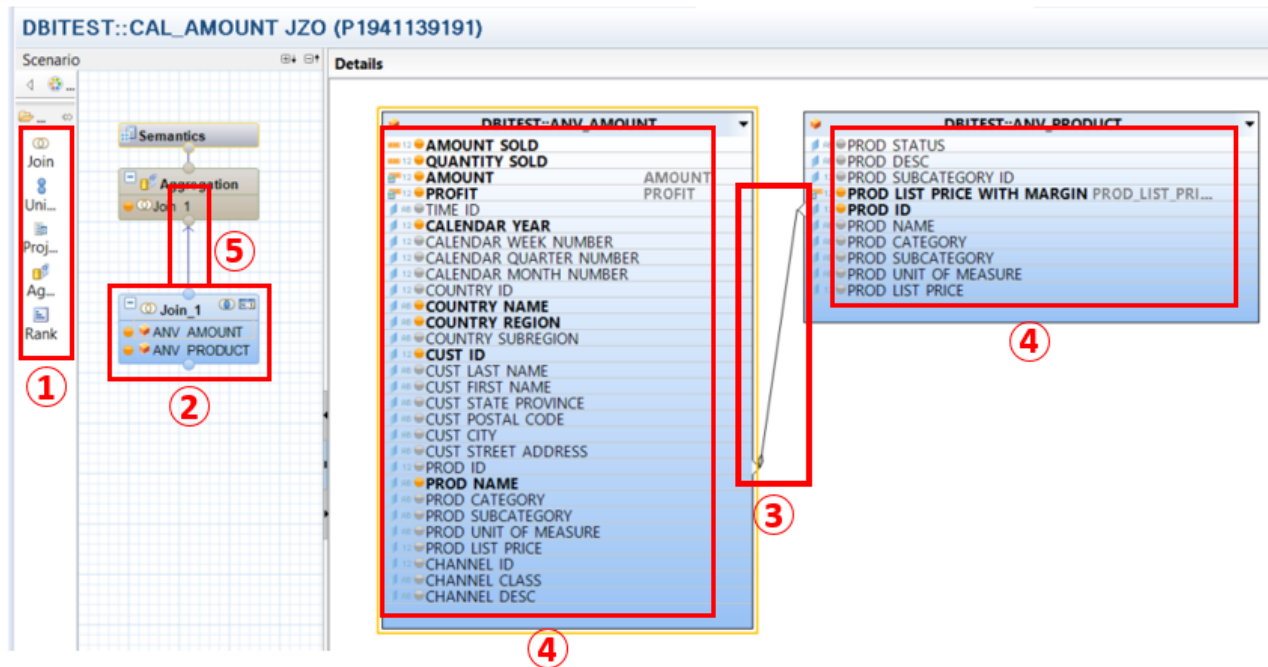
Calculation View

Type: ▾
Data **Note:** If you set the Data Category as Cube, the default node is Aggregation. You can change the default node to Star

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#)[Read More](#)

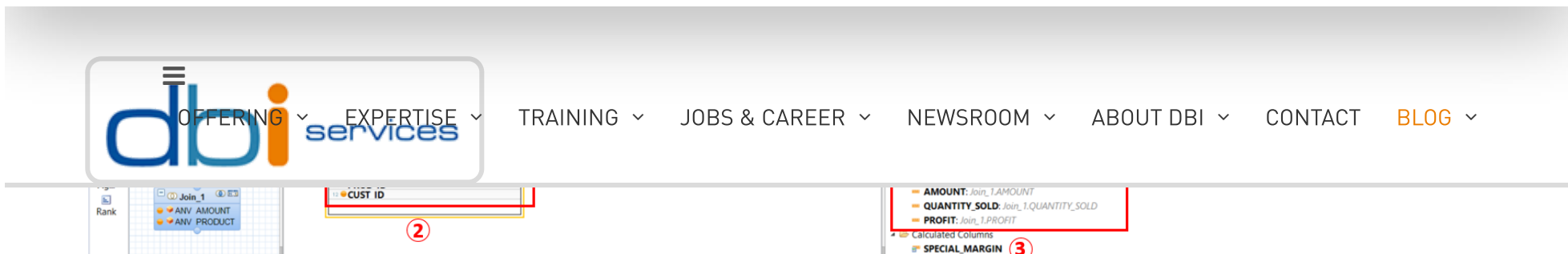
5. Join your "Aggregation operation" to the "Aggregation" box



Create your Calculation View Layout

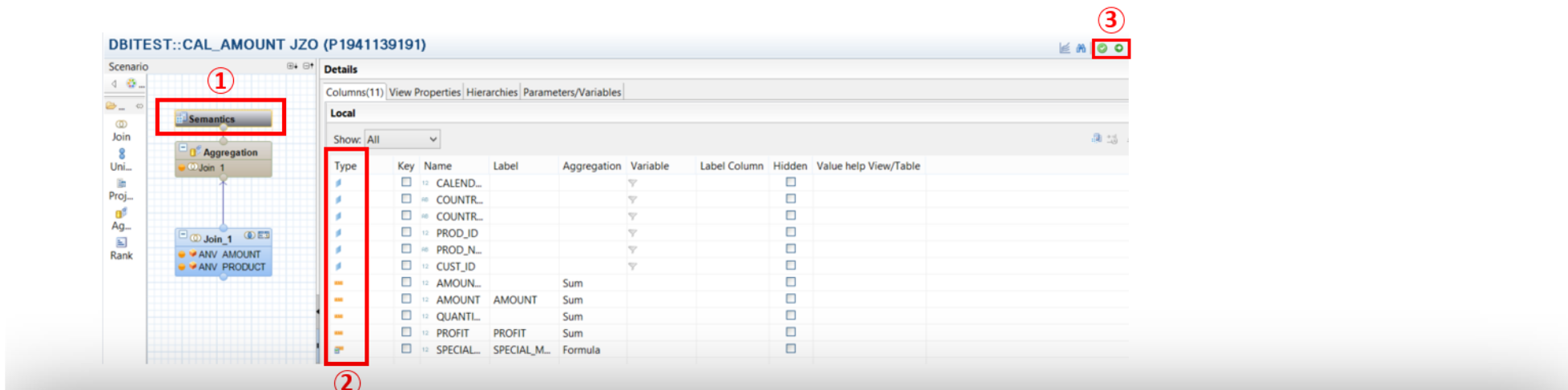
This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#) [Read More](#)



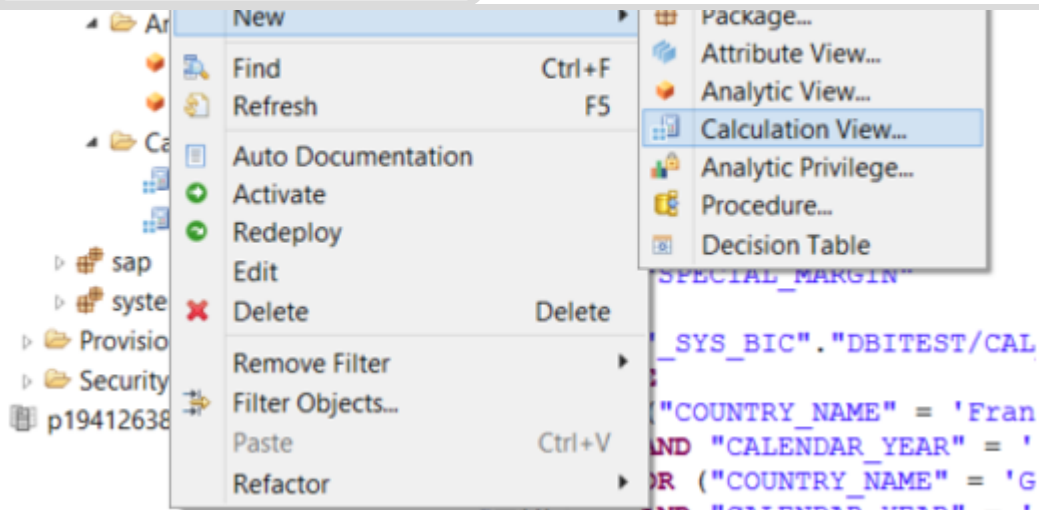
Check the objects

1. Click on "Semantics" box
2. Select the type of the objects
3. Validate and activate the view



This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#) [Read More](#)



Select the "SQL Script" type

details

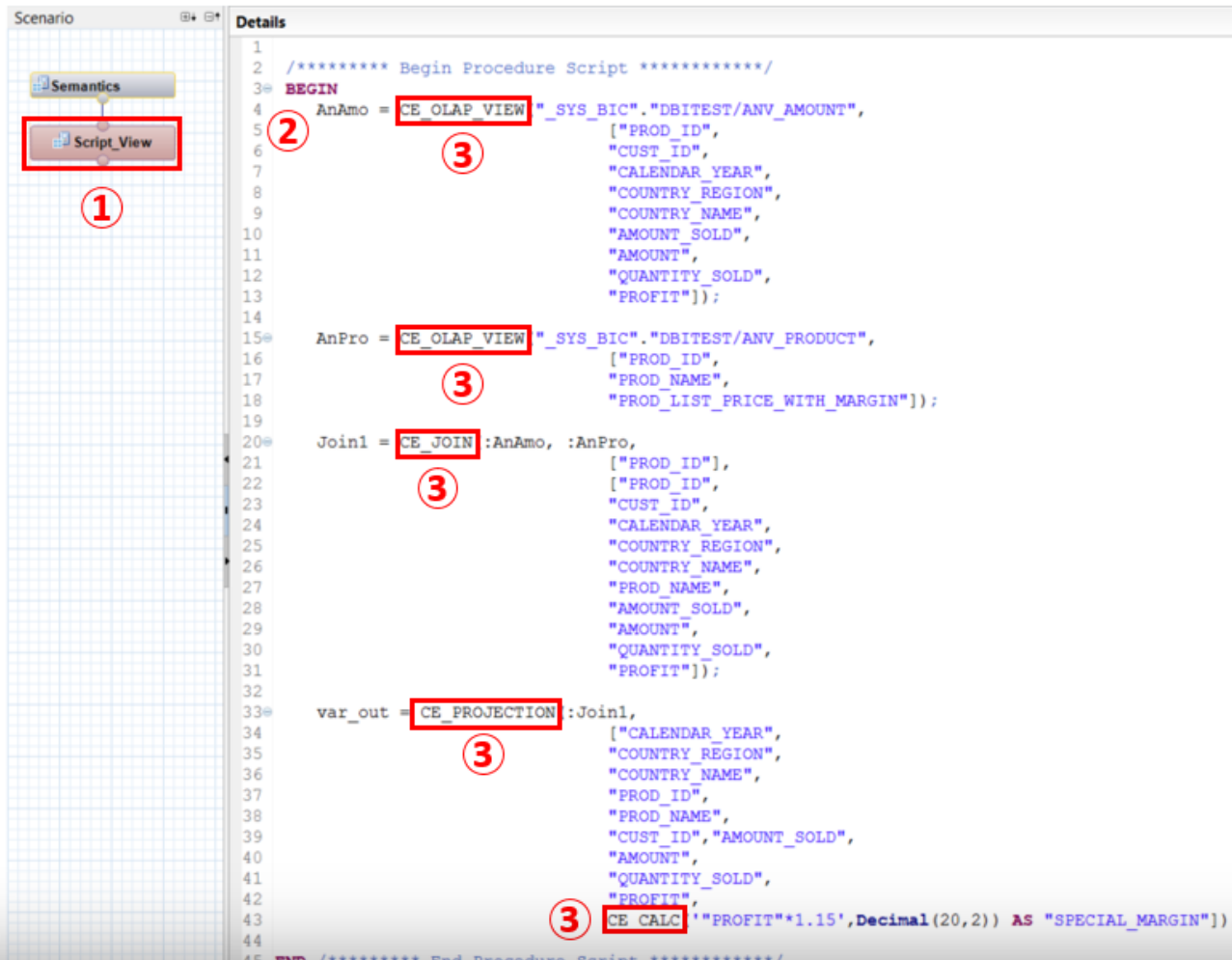
Name:* Label: Package:* View Type: ▾☐ Copy From: Subtype: ▾

Calculation View

Type: ▾
Data **Note:** If you set the Data Category as Cube, the default node is Aggregation. You can change the default node to Star

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#) [Read More](#)



The screenshot shows a SQL script editor with a 'Scenario' tab on the left and a 'Details' tab on the right. The 'Script_View' icon in the 'Scenario' tab is highlighted with a red box and a circled '1'. The script in the 'Details' tab is as follows:

```

1
2 /***** Begin Procedure Script *****/
3 BEGIN
4   AnAmo = CE_OLAP_VIEW "_SYS_BIC"."DBITEST/ANV_AMOUNT",
5           ["PROD_ID",
6           "CUST_ID",
7           "CALENDAR_YEAR",
8           "COUNTRY_REGION",
9           "COUNTRY_NAME",
10          "AMOUNT_SOLD",
11          "AMOUNT",
12          "QUANTITY_SOLD",
13          "PROFIT"]];
14
15   AnPro = CE_OLAP_VIEW "_SYS_BIC"."DBITEST/ANV_PRODUCT",
16          ["PROD_ID",
17          "PROD_NAME",
18          "PROD_LIST_PRICE_WITH_MARGIN"]];
19
20   Join1 = CE_JOIN :AnAmo, :AnPro,
21          ["PROD_ID"],
22          ["PROD_ID",
23          "CUST_ID",
24          "CALENDAR_YEAR",
25          "COUNTRY_REGION",
26          "COUNTRY_NAME",
27          "PROD_NAME",
28          "AMOUNT_SOLD",
29          "AMOUNT",
30          "QUANTITY_SOLD",
31          "PROFIT"]];
32
33   var_out = CE_PROJECTION :Join1,
34            ["CALENDAR_YEAR",
35            "COUNTRY_REGION",
36            "COUNTRY_NAME",
37            "PROD_ID",
38            "PROD_NAME",
39            "CUST_ID", "AMOUNT_SOLD",
40            "AMOUNT",
41            "QUANTITY_SOLD",
42            "PROFIT"],
43            CE_CALC "PROFIT*1.15',Decimal(20,2)) AS 'SPECIAL_MARGIN'";
44
45 *****/ End Procedure Script *****/
  
```

Annotations in the script include:

- A circled '2' next to line 4.
- A circled '3' next to the 'CE_OLAP_VIEW' call on line 4.
- A circled '3' next to the 'CE_JOIN' call on line 20.
- A circled '3' next to the 'CE_PROJECTION' call on line 33.
- A circled '3' next to the 'CE_CALC' call on line 43.

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#) [Read More](#)

DBITEST::CAL_AMOUNT_CE JZO (P1941139191)

Scenario

Semantics

Script View

Details

Columns(11) View Properties Hierarchies Parameters/Variables

Local

Show: All

Type	Key	Name	Label	Aggregation	Variable	Label Column	Hidden	Value help	View/Table
	<input type="checkbox"/>	12 CALENDAR_YEAR	CALENDAR_YEAR				<input type="checkbox"/>		
	<input type="checkbox"/>	46 COUNTRY_REGION	COUNTRY_REGION				<input type="checkbox"/>		
	<input type="checkbox"/>	46 COUNTRY_NAME	COUNTRY_NAME				<input type="checkbox"/>		
	<input type="checkbox"/>	12 PROD_ID	PROD_ID				<input type="checkbox"/>		
	<input type="checkbox"/>	46 PROD_NAME	PROD_NAME				<input type="checkbox"/>		
	<input type="checkbox"/>	12 CUST_ID	CUST_ID				<input type="checkbox"/>		
	<input type="checkbox"/>	12 AMOUNT_SOLD	AMOUNT_SOLD	Sum			<input type="checkbox"/>		
	<input type="checkbox"/>	12 AMOUNT	AMOUNT	Sum			<input type="checkbox"/>		
	<input type="checkbox"/>	12 QUANTITY_SOLD	QUANTITY_SOLD	Sum			<input type="checkbox"/>		
	<input type="checkbox"/>	12 PROFIT	PROFIT	Sum			<input type="checkbox"/>		
	<input type="checkbox"/>	12 SPECIAL_MARGIN	SPECIAL_MARGIN	Sum			<input type="checkbox"/>		

SQL Performance comparison

Goal of the test

In this part, I will compare the SQL performance from two calculated views that have been built with the two different methods:

“Graphical” method

“SQL assistant” method

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#)

[Read More](#)



OFFERING ▾

EXPERTISE ▾

TRAINING ▾

JOBS & CAREER ▾

NEWSROOM ▾

ABOUT DBI ▾

CONTACT

BLOG ▾

CAL_AMOUNT_CE (SQL Scripting method)

Test with the “Graphical” calculated view

SQL Query

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#) [Read More](#)

```
"COUNTRY_NAME",  
"PROD_ID",  
"PROD_NAME",  
"CUST_ID",  
"AMOUNT_SOLD",  
"AMOUNT",  
"QUANTITY_SOLD",  
"PROFIT",  
"SPECIAL_MARGIN"  
FROM  
  "_SYS_BIC"."DBITEST/CAL_AMOUNT"  
WHERE  
  ("COUNTRY_NAME" = 'France'  
  AND "CALENDAR_YEAR" = '2011')  
  OR ("COUNTRY_NAME" = 'Germany'  
  AND "CALENDAR_YEAR" = '2010');
```

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#)

[Read More](#)



OFFERING ▾

EXPERTISE ▾

TRAINING ▾

JOB & CAREER ▾

NEWSROOM ▾

ABOUT DBI ▾

CONTACT

BLOG ▾

Test with the “SQL Scripting” calculated view

SQL Query

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#) [Read More](#)

```
"COUNTRY_NAME",  
"PROD_ID",  
"PROD_NAME",  
"CUST_ID",  
"AMOUNT_SOLD",  
"AMOUNT",  
"QUANTITY_SOLD",  
"PROFIT",  
"SPECIAL_MARGIN"  
FROM  
  "_SYS_BIC"."DBITEST/CAL_AMOUNT_CE"  
WHERE  
  ("COUNTRY_NAME" = 'France'  
  AND "CALENDAR_YEAR" = '2011')  
  OR ("COUNTRY_NAME" = 'Germany'  
  AND "CALENDAR_YEAR" = '2010');
```

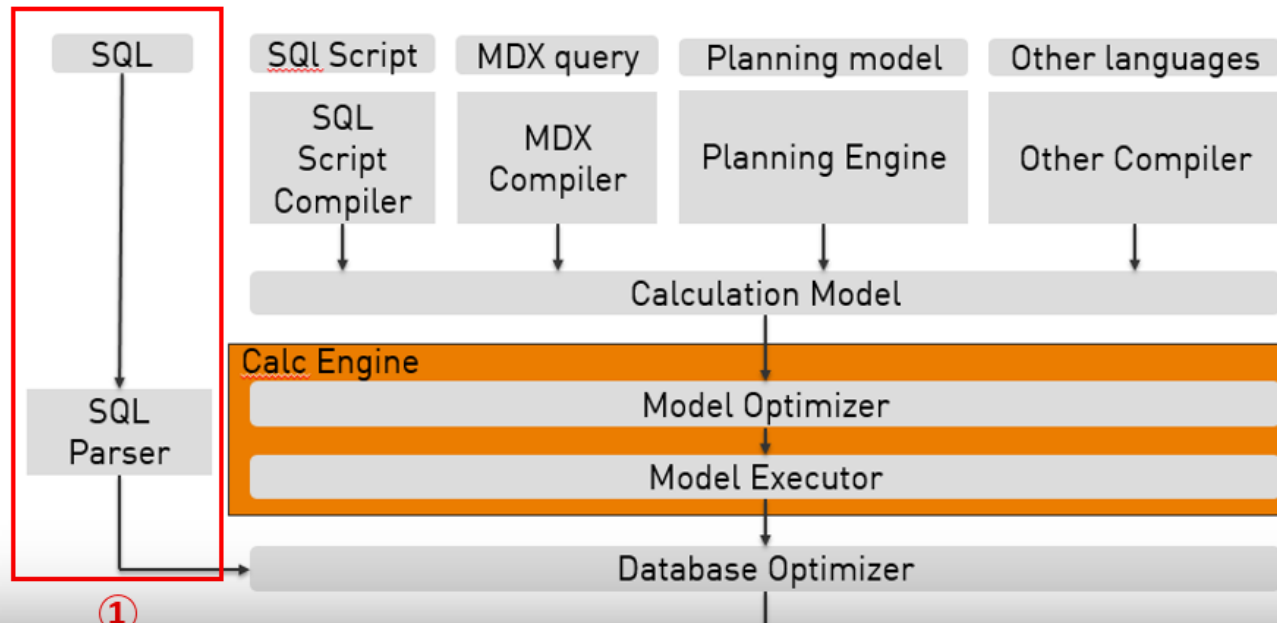
This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#)[Read More](#)

Performance decoding

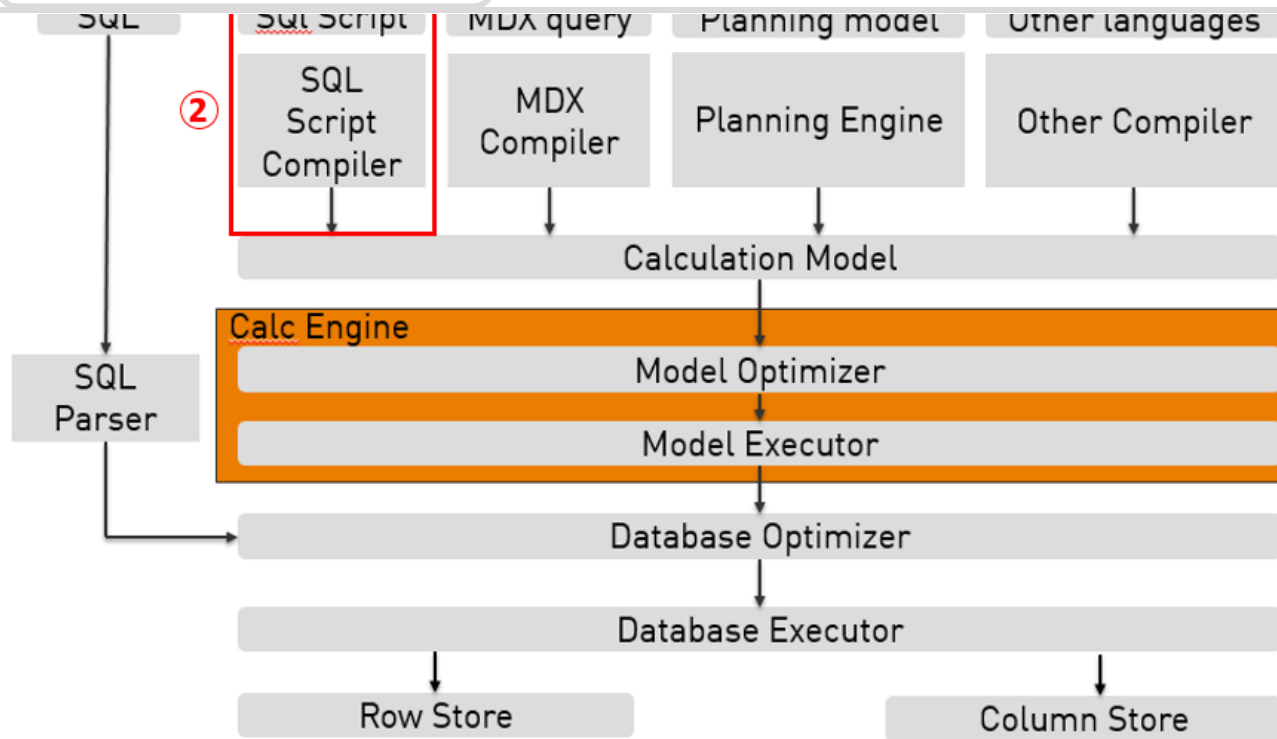
Why the use of these CE functions boost the performances of your queries? The explanation is located in the query execution plan and especially in the use of the CALC engine from the SAP HANA database.

When you send a “normal” SQL query in the SAP HANA database, the CALC engine is not used. The SQL parser send the query directly to the “Database optimizer” to optimize the execution of the query (1).

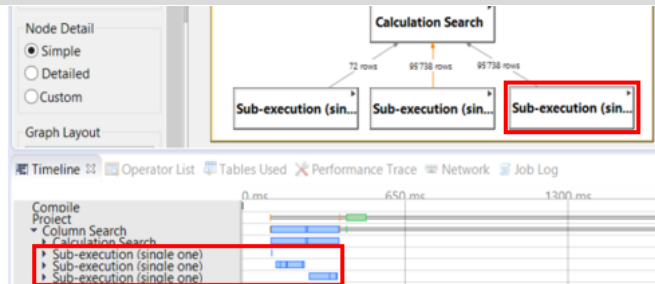


This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

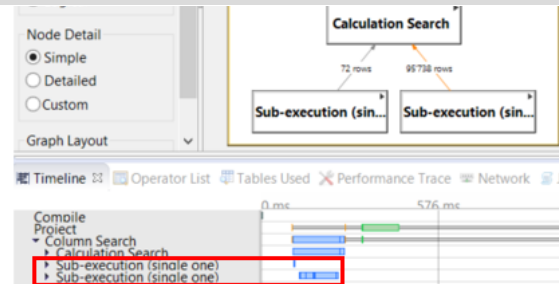
[Reject](#)
[Read More](#)



In our case, when we analyze the “normal” SQL query, the “calculation search” task has been split in 3 different sub-queries that can’t start at the same time.



SQL without view using CE functions



SQL With view using CE functions

Conclusion

The use of CE functions in the creation of calculated views can significantly accelerate the execution of your SQL queries. The CALC engine from SAP HANA is optimized to use these functions.

There's only one restriction using this kind of functions. The performance will dramatically reduce if you try to create a SQL query mixing "normal" and "optimized" calculated views.

**Michael Stannard says:**

June 20, 2016 at 18 h 20 min

[Reply to Michael](#)

Hi

Interesting article.

Do you have information on SQL scripted CV (traditional SQL coding) verses Graphical CVs or indeed your CE functions based CVs?

Regards

Michael

**Palani Ramu says:**

April 13, 2018 at 17 h 26 min

[Reply to Palani](#)

Thanks for the document posted.

I have small query , I have to collect columns more that 10 table, such situation how to use CE function ?



OFFERING ▾

EXPERTISE ▾

TRAINING ▾

JOB & CAREER ▾

NEWSROOM ▾

ABOUT DBI ▾

CONTACT

BLOG ▾

Name *

Email *

Website

☐ Save my name, email, and website in this browser for the next time I comment.



I'm not a robot

reCAPTCHA
[Privacy](#) - [Terms](#)

SUBMIT COMMENT

This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#)

[Read More](#)

[OFFERING](#) ▾[EXPERTISE](#) ▾[TRAINING](#) ▾[JOBS & CAREER](#) ▾[NEWSROOM](#) ▾[ABOUT DBI](#) ▾[CONTACT](#)[BLOG](#) ▾

EXPERTISE IN DATABASE & MIDDLEWARE

[Oracle database expertise](#)[SQL Server expertise](#)[MySQL/MariaDB expertise](#)[PostgreSQL expertise](#)[NoSQL expertise](#)[SharePoint expertise](#)[OpenText Documentum expertise](#)[Linux expertise \(Oracle Linux, Red Hat\)](#)

TRAININGS IN DATABASE & MIDDLEWARE

[Microsoft](#)[Oracle](#)[Open Source DB](#)[Operating system](#)

USEFUL INFORMATION

[News & Events](#)[Jobs openings](#)[Offices](#)[Blog of dbi services](#)[Imprint](#)

This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish. [Accept](#)

[Reject](#)[Read More](#)