

Innovating with SAP HANA

What are my options?

Lars Breddemann

SAP



This Presentation Comprises:

- ACT 1: Who am I & what is this about?
 - > Lars Breddemann
 - > SAP HANA development outside the classic use cases
 - > Intel NUC as a SAP HANA development system
- ACT 2: Small systems, small problems, large systems, ...
 - > Finding performance issues
 - > Just not your type
 - > Wrinkles in date and time
 - > That DUMMY has to go...
 - > Performance left out(er)
 - > One for everyone
- ACT 3: Finale
 - > Conclusion

That's me – Lars Breddemann



- Since 2003 with SAP (Austria, Germany, Australia)
 - Support, Development, Custom Development
 - Now: SAP Connected Health Platform Development
- In Melbourne since 2015
- Certified Oracle DBA, SAP BI, SAP HANA Professional
- [SAP HANA Distinguished Engineer](#), top contributor/moderator on [SAP Community Network](#), [stackoverflow](#), [SAP \(internal\) JAM group](#)
- Co-Author of the **SAP HANA Administration** book
- Interested in growing knowledge sharing culture



SAP HANA Administration (HANA Admin) (SAP PRESS) Hardcover – October 3, 2014
by Richard Bremer (Author), Lars Breddemann (Author)
★★★★★ 10 customer reviews

Sponsored by



What I talk about today

- (Database) application development on SAP HANA
- What does HANA development look like outside of classic ERP use cases?
- Example: **Connected Healthcare Platform Apps**
- Hints on how to not ruin the database performance
- Titbits you can use today to make your programs better looking, faster performing and less error prone
- Techniques to find out about what HANA does so that you can write better code

Connected Health

- SAP HANA is the foundation technology on which SAP builds the standard software suite and enables innovative solutions based on S/4 HANA.
- That's true, but:
- Beyond that, it's a general data processing application platform that allows virtually any kind of data centric development
- One example for such a development is the SAP Connected Healthcare (CH) platform.

Connected Health

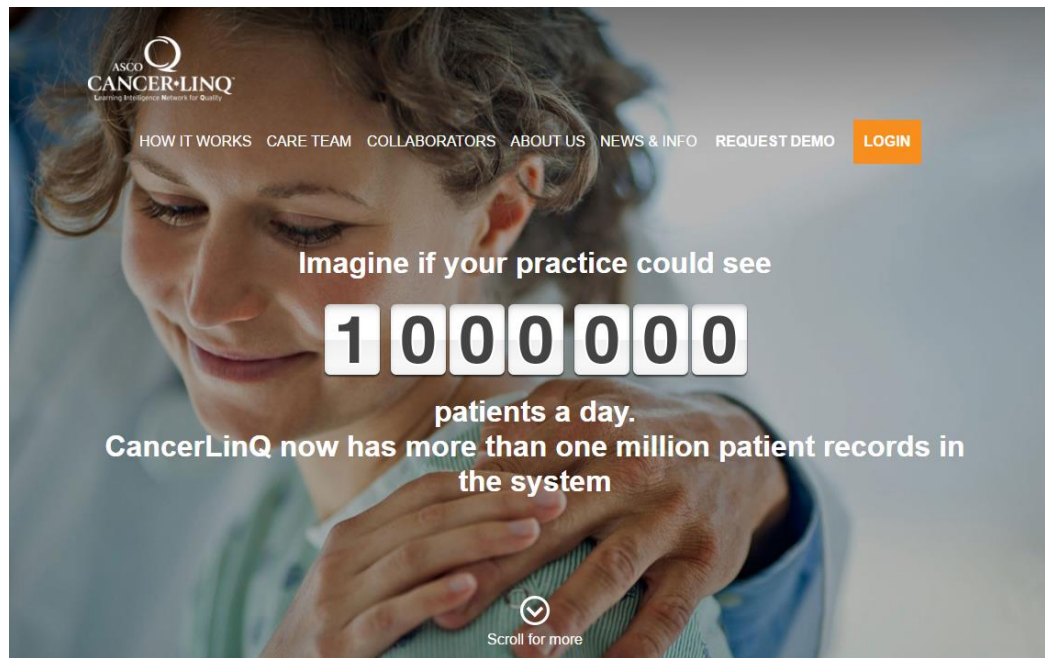
- CH allows to integrate medical data and makes information, Big Data analytics and scientific data processing available to practitioners, doctors and researchers.
- With CH and the applications build on top of CH SAP joined the fight for better healthcare

“Doctors in my case were sending each other dictation to each other’s cell phones,” Mr. McDermott said in his first interview since the accident, the scars of his wound still visible under dark glasses. “There were digital files of M.R.I.s, data from my surgery in the trauma unit and scans from the emergency response team that never went into the case file.”



ASCO CancerLinQ

- Information on Cancer treatments, therapies, patient histories
- Co-created by
 - > ASCO
 - > SAP (Standard Dev.)
 - > SAP Innovation Center Network
 - > SAP Custom Development
- Based on Connected Health Platform



WE HAVE STRENGTH IN OUR NUMBERS

ASCO CancerLinQ

SAP Connected Health Platform

- Data ingestion

- › data cleansing, de-identification
- › Natural language processing NLP (doctor letters free text into structured information), based on SAP HANA text analysis
- › automatic codification of information, Ontology services to allow ad-hoc matching of codes and free text across different code-systems:

e.g.

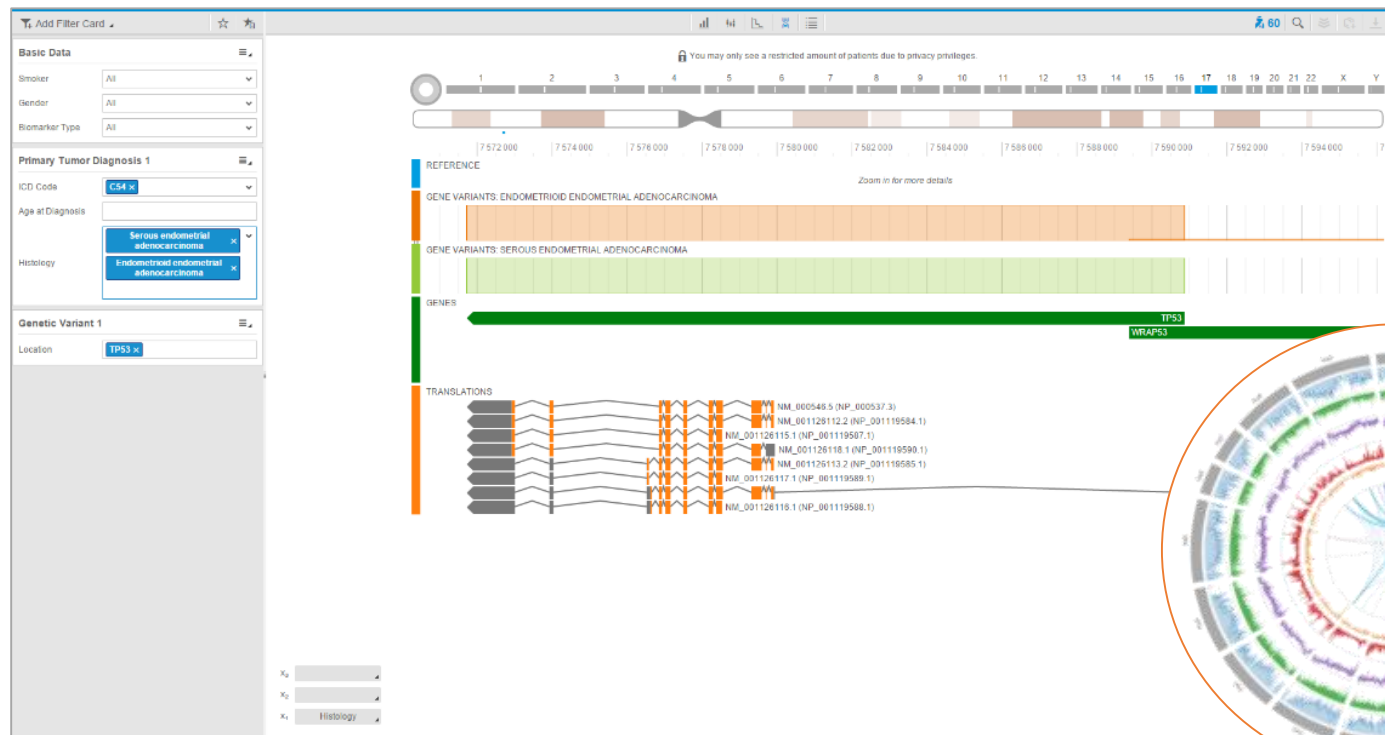
‘ICD9CM’ - 174.0 – “Malignant neoplasm of female breast”

↔ ‘ICD10CM’ - C50.01 – “Malignant neoplasm of nipple and areola, female”

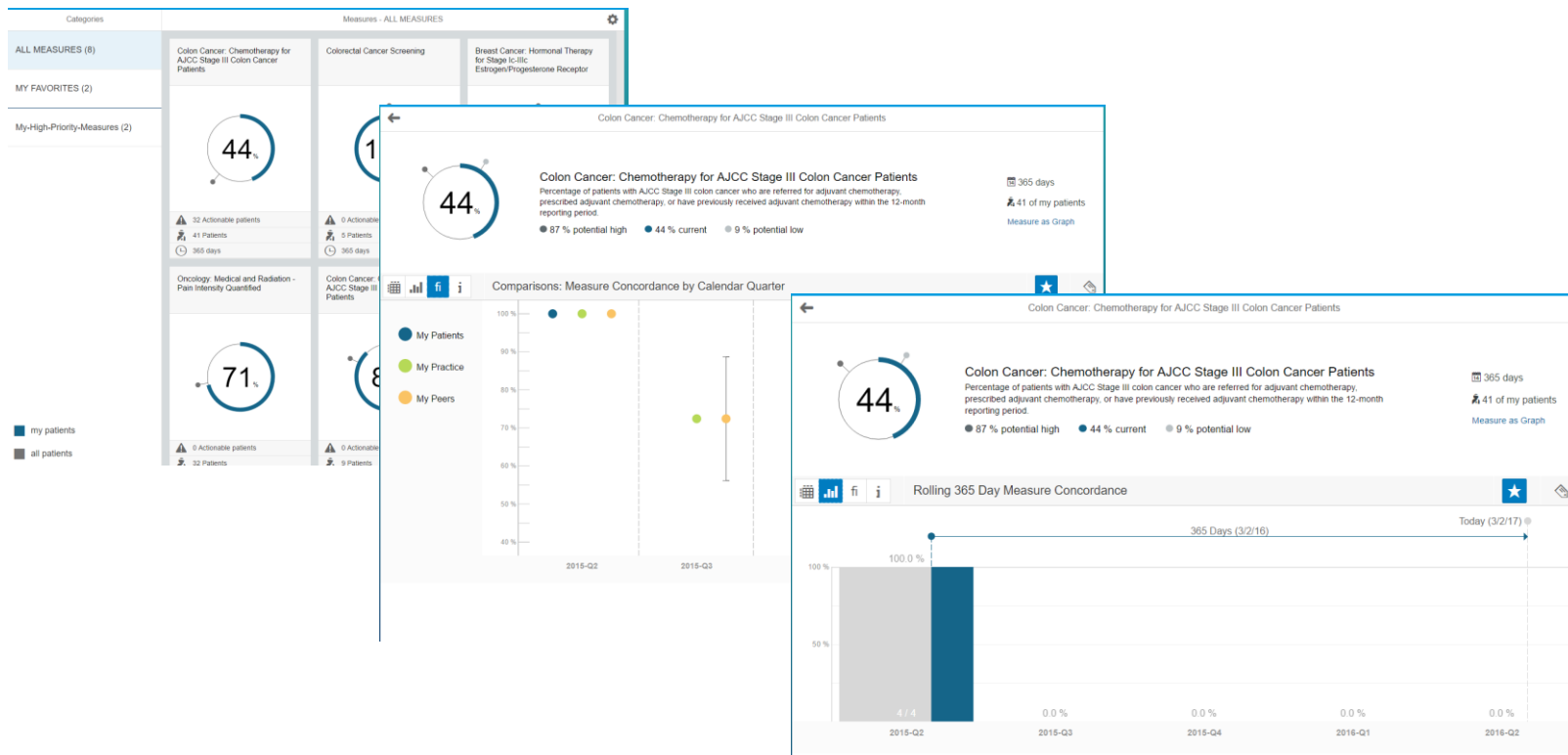
- Data analytics

- › Ad-hoc queries via SAP Medical Research Insights
- › Genomic Variants browser
- › Clinical Measure Analytics
- › Datamart functionality for data scientists

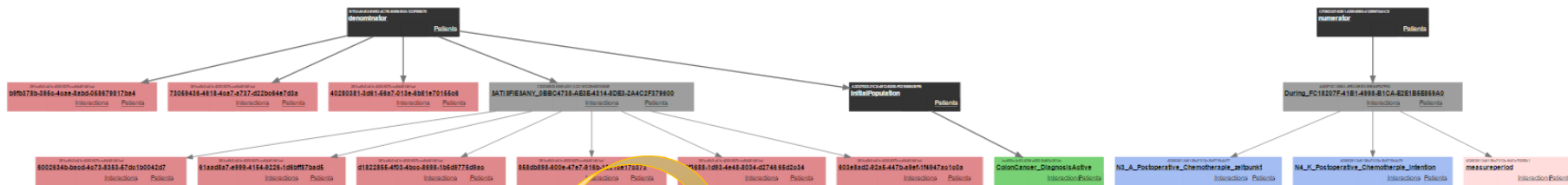
Genomic Variants Browser (MRI)



Clinical Measure Analytics (CMA)



Clinical Measure Analytics (CMA)



54fbac_d1822855-4f03-4bcc-8698-1b5d9775d9ac-281ca9b3

Instance: CM5, Measure ID: onk-26

54fbac_d1822855-4f03-4bcc-8698-1b5d9775d9ac-281ca9b3

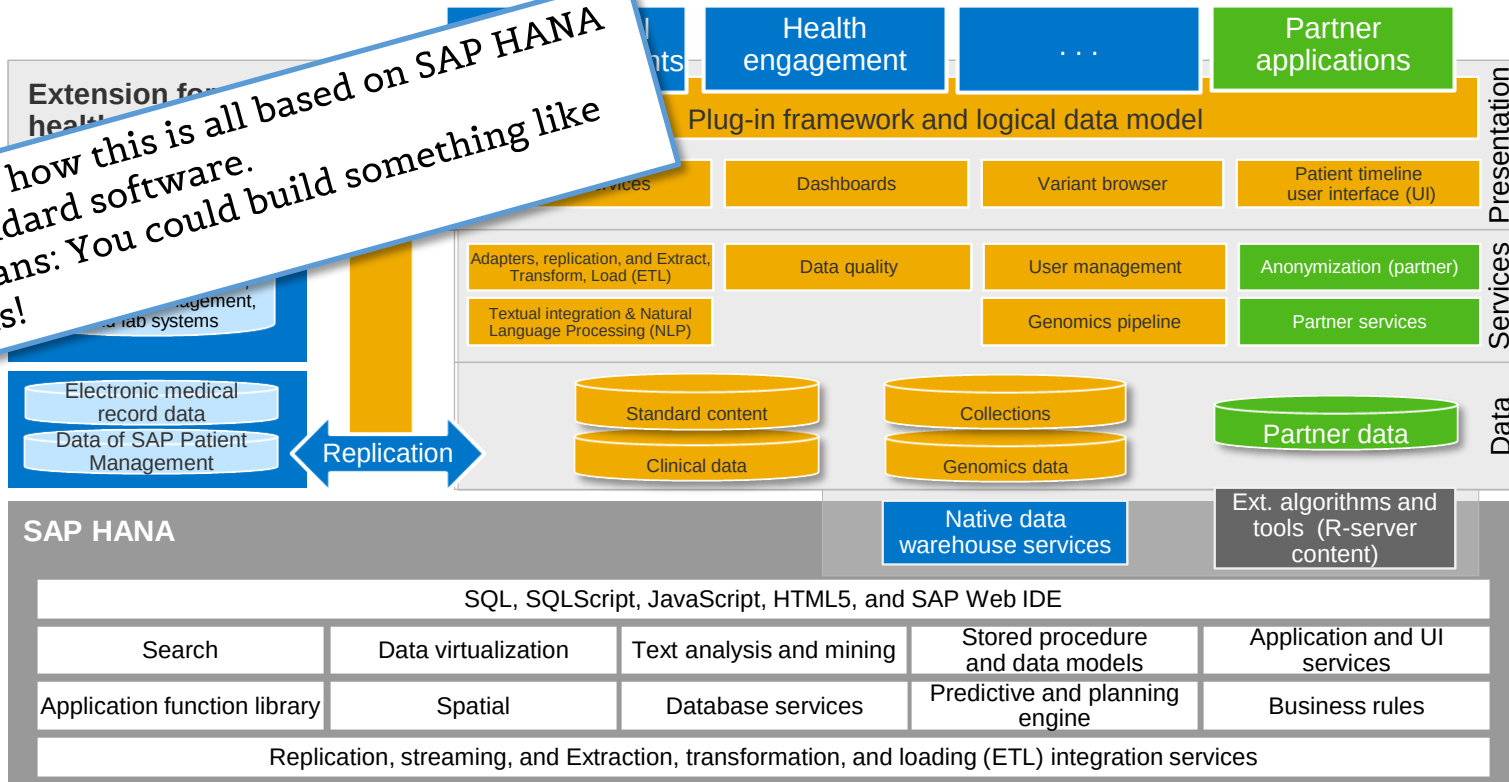
Instance: CM5, Measure ID: onk-26

Criterion XML	Translated SQL
<pre> 1 <procedureCriteria classCode="PROC" moodCode="EVN"> 2 <templateId> 3 <item extension="2014-11-24" root="2.16.840.1.113883.10.20.28.3.67"/> 4 </templateId> 5 <cid extension="d1822855-4f03-4bcc-8698-1b5d9775d9ac" root="281ca9b3-d41e-4000-827b-ca94481461ad"/> 6 <code valueSet="2.16.840.1.113883.3.526.3.1098"> 7 <displayName value="Clinical Staging Procedure Grouping Value Set"/> 8 </code> 9 <title value="Procedure, Performed"/> 10 <statusCode code="COMPLETED"/> 11 <outboundRelationship typeCode="REFR"> 12 <observationCriteria classCode="OBS" moodCode="EVN"> 13 <templateId> 14 <item root="2.16.840.1.113883.10.20.28.3.101"/> 15 </templateId> 16 <cid extension="result" root="d1822855-4f03-4bcc-8698-1b5d9775d9ac"/> 17 <code code="394617004" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED CT"/> 18 <title value="result"/> 19 <value valueSet="2.16.840.1.113883.3.526.3.1328" xsi:type="CD"> 20 <displayName value="Postoperative Histologic/Staging - pN Status N1b Grouping Value Set"/> 21 </value> 22 </observationCriteria> 23 </outboundRelationship> 24 </procedureCriteria> </pre>	

Criterion XML	Translated SQL
<pre> 1 2 (3 SELECT s68 . "PATIENT_EMPI_ID" , s68 . "INTERACTION_ID" , s68 . "START" , IFNULL ("END" , '9999-12-31') AS "END" 4 FROM "SYS_BIC" . "sap.hc.asco.cancerling.cml.db/INTERACTIONS" (PLACEHOLDER . "\$\$TF_MIN\$\$" => :TF_MIN , PLACEHOLDER 5 INNER JOIN "SAP_HPH" . "sap.hph.fnd::INTERACTION_DETAILS" s69 ON s68 . "INTERACTION_ID" = s69 . "INTERACTION_ID" 6 INNER JOIN "SAP_HPH" . "sap.hph.fnd::INTERACTION_DETAILS" s70 ON s68 . "INTERACTION_ID" = s70 . "INTERACTION_ID" 7 WHERE s68 . "ITYPE_CS_CODE" = 'Procedure, Performed' AND (s69 . VALUE_CODING_SYSTEM , s69 . VALUE_CS_CODE) IN (8 SELECT IFNULL (ucsm . "UmlsCodeSystem" , vsc . "CodeSystem") AS "CodeSystem" , vsc . "Code" 9 FROM "SAP_HPH" . "sap.hc.hph.ots.models::Core.ValuesetCodes" vsc 10 LEFT OUTER JOIN "SAP_HPH" . "sap.hc.hph.ots.models.reference.ums::Utils.UmlsCodeSystemMappings" ucsm ON ucsm . " 11 WHERE vsc . "ValueSetOID" = '2.16.840.1.113883.3.526.3.1098' 12) AND (s70 . VALUE_CODING_SYSTEM , s70 . VALUE_CS_CODE) IN (13 SELECT IFNULL (ucsm . "UmlsCodeSystem" , vsc . "CodeSystem") AS "CodeSystem" , vsc . "Code" 14 FROM "SAP_HPH" . "sap.hc.hph.ots.models::Core.ValuesetCodes" vsc 15 LEFT OUTER JOIN "SAP_HPH" . "sap.hc.hph.ots.models.reference.ums::Utils.UmlsCodeSystemMappings" ucsm ON ucsm . " 16 WHERE vsc . "ValueSetOID" = '2.16.840.1.113883.3.526.3.1328' 17) 18) </pre>	

SAP Connected Health Platform

Note how this is all based on SAP HANA standard software.
Means: You could build something like this!



Intel NUC “dev box”

- Getting a HANA dev environment used to be hard:
 - HCP (trial) – slow due to latency (trial server located in Germany), no full access to the system, dependent on internet access
 - Cloud (AWS, Azure, ...) instance – pay per use, dependent on internet access
 - Access to an actual HANA server, expensive, shared with others, usually admin required
- Intel **N**(ext) **U**(nit) **C**(omputing) systems are available with up to 32GB RAM, SSDs & Intel i7 quad-core CPUs
 - That's enough to do some actual development work!

Intel NUC – Skull Canyon

- Relatively easy to set up
- Can run multiple HANA instances
- Can be used to run [SAP HANA Express Edition \(HXE\)](#)
 - Supported for productive use up to 32 GB
 - For free 😊
- SCN Blog post [“HANA in a pocket, a skull and some dirty hands on Linux”](#)
- Runs circles around cloud systems.



This Presentation Comprises:

- ACT 1: Who am I & what is this about?
 - > Lars Breddemann
 - > SAP HANA development outside the classic use cases
 - > Intel NUC as a SAP HANA development system
- ACT 2: Small systems, small problems, large systems, ...
 - > Finding performance issues
 - > Just not your type
 - > Wrinkles in date and time
 - > That DUMMY has to go...
 - > Performance left out(er)
 - > One for everyone
- ACT 3: Finale
 - > Conclusion

Small systems, small problems, large systems, ...



- Systems are (hopefully) made up from small functional blocks that are stacked/chained/combined for larger functions and processes.
 - E.g. think of the way the virtual data models of S/4 or HANA live are constructed
- Minor mistakes in base functions can accumulate and have profound effects on the overall system performance
- Once a problem becomes obvious, typically the question is to locate the cause of it in (a meanwhile) large setup of tables, views, functions and procedures.

Sponsored by



Finding Performance issues

- It's about:
 - > Finding out why a query runs too slow (for whatever it is supposed to do)
 - > Look for alternative (better) ways to yield the result.
(doesn't necessarily mean the same query, table or data needs to be used. Remember: we're the developers around here ;-))
- But also:
 - > Understanding memory consumption
(fast query, but uses all memory?)
 - > Understanding CPU time consumption
(fast query, but system is practically blocked while running?)
 - > Understanding how the query depends on the processed data size
(does "more data" have to mean "longer processing time"?)

Can you have a look at this query?

```
select
  'e02b540f-cc52-4526-acce-eb570099da87' as CACHE_ID
  , 'CMA_numeratorPatients' as COHORT_NAME
  , PATIENT_EMPI_ID
from "_SYS_BIC"."sap.hc.xxxx.yyyyyyyyy.cml.measures.staging-il/CMA_numeratorPatients"
(PPLACEHOLDER = ('$$TF_MIN$$', '0001-01-01',
                  '$$TF_MAX$$', '9999-12-31',
                  '$$BENCHMARK_ID$$', 'my',
                  '$$PARAM1$$', '',
                  '$$PARAM2$$', '')) ;
```

... sure, why not!

- EXPLAIN PLAN is a great first and relatively un-intrusive option to get information on query execution

- This is a snippet from a fairly small EXPLAIN PLAN with roughly 140 single operations

- Plans can become a lot larger – latest I've worked on was 2700 operations

- Text display is not practical for analysis

- Take the output and paste it into EXCEL!



1. Explain Plan II

Text Import Wizard - Step 2 of 3

This screen lets you set the delimiters your data contains. You can see how your text is affected in the preview below.

Delimiters

☐ Tab

☒ Semicolon

☐ Comma

☐ Space

☐ Other:

☐ Treat consecutive delimiters as one

Text qualifier:

Data preview

1	STATEMENT_NAME	OPERATOR_NAME
2	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	INSERT
3	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	DISTINCT
4	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	HASH JOIN
5	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	DISTINCT

Cancel < Back Next > Finish

- Text-Import Wizard!
- Delimited by ';' (semicolon)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
1		STATEMENT	OPERATOR	OPERATOR	EXECUTIO	DATABASE	SCHEMA	TABLE	TABLE	TABLE	TABLE	TABLE	TABLE	TABLE	TABLE	TABLE	TABLE	
2	1	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	INSERT	?	COLUMN		SAP_ASCC	sap.hc.asc	COLUMN	?	?	?	?	1	?	1	1 vhascqht	
3	2	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	DISTINCT	GROUPIN	ROW	?	?	?	?	?	?	1	0,8608157	2	1	2	1 vhascqht	
4	3	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	HASH	JC	HASH	BUIL	ROW	?	?	?	?	1	0,8608132	3	2	3	1 vhascqht	
5	4	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	DISTIN	GROUPIN	ROW	?	?	?	?	?	?	1	0,3157296	4	3	4	1 vhascqht	
6	5	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	HASH	HASH	BUIL	ROW	?	?	?	?	?	200	0,3153166	5	4	5	1 vhascqht	
7	6	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	COLU	BI	INTERA	COLUMN	?	?	?	?	?	10	0,0015151	6	5	6	1 vhascqht	
8	7	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	COL		COLUMN		SAP_ASCC	SYS_SS	C	CALCULAT	10	10	?	7	6	7	1 vhascqht	
9	8	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	HAVI	DAYS	_BET	ROW	?	?	?	?	?	1	0,3063803	8	5	6	2 vhascqht	
10	9	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	DIST	GROUPIN	ROW	?	?	?	?	?	?	1	0,3063797	9	8	7	1 vhascqht	
11	10	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	HA	HASH	BUIL	ROW	?	?	?	?	?	1	0,3063772	10	9	8	1 vhascqht	
12	11	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	CC	sap.hhp.fr	COLUMN	?	?	?	?	?	?	#####	0,0024823	11	10	9	1 vhascqht	
13	12	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	J	JOIN	CON	COLUMN	?	?	?	?	?	#####	0,0024732	12	11	10	1	
14	13	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45			COLUMN		SAP_HHP	sap.hhp.fr	COLUMN	?	42.79	42.79	?	13	12	11	1 vhascqht	
15	14	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45			COLUMN		SAP_HHP	sap.hhp.fr	COLUMN	?	3.507	3.507	?	14	12	11	2 vhascqht	
16	15	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	FILTER	CO	COLUMN		SAP_HHP	sap.hhp.fr	COLUMN	?	16.894	2.410,3264	?	15	12	11	3 vhascqht	
17	16	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	VSC	Code	COLUMN	?	?	?	?	?	?	#####	0,0003832	16	12	11	4 vhascqht	
18	17	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	JOIN	CON	COLUMN	?	?	?	?	?	?	#####	0,0003623	17	16	12	1	
19	18	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	FILTER	CO	COLUMN		SAP_HPH	sap.hc.hpl	COLUMN	?	139.264	#####	?	18	17	13	1 vhascqht	
20	19	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45			COLUMN		SAP_HPH	sap.hc.hpl	COLUMN	?	16	16	?	19	17	13	2 vhascqht	
21	20	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	M	(TO	_INT	T	ROW	?	?	?	?	?	2	0,3038063	20	10	9	2 vhascqht
22	21	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	C	GROUPIN	ROW	?	?	?	?	?	?	1	0,1519017	21	20	10	1 vhascqht	
23	22	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	CRITREF	S	ROW	?	?	?	?	?	?	1	0,1518992	22	21	11	1 vhascqht	
24	23	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	WINDOW		ROW	?	?	?	?	?	?	1	0,1518986	23	22	12	1 vhascqht	
25	24	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	I	INTERAC	COLUMN	?	?	?	?	?	?	11.428,5710,0259259	24	23	13	1 vhascqht		
26	25	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	JOIN	CON	COLUMN	?	?	?	?	?	?	11.428,5710,0241943	25	24	14	1		
27	26	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45			COLUMN		SAP_HHP	sap.hhp.fr	COLUMN	?	42.79	42.79	?	26	25	15	1 vhascqht	
28	27	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	FILTER	CO	COLUMN		SAP_HHP	sap.hhp.fr	COLUMN	?	16.894	8.056,1364	?	27	25	15	2 vhascqht	
29	28	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45			COLUMN		SAP_HHP	sap.hhp.fr	COLUMN	?	3.507	3.507	?	28	25	15	3 vhascqht	
30	29	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	VSC	Code	COLUMN	?	?	?	?	?	?	1,323,008	0,0005484	29	25	15	4 vhascqht	
31	30	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	GROUPIN	COLUMN	?	?	?	?	?	?	?	132	0,0005284	30	29	16	1	
32	31	2797a12f-9e34-4eae-asf1-ab1a0d6d7c45	JOIN	CON	COLUMN	?	?	?	?	?	?	139,264	0,0003625	31	30	17	1	

1. Explain Plan III

Create Table

?

×

Where is the data for your table?

☒ My table has headers

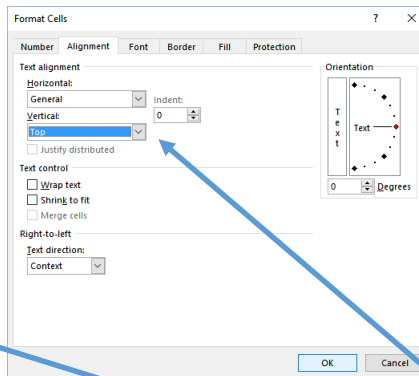
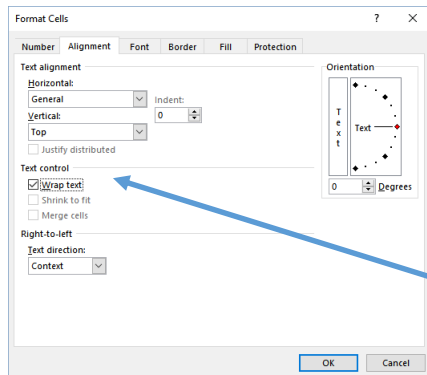
OK

Cancel

- CTRL+T (create table)
- Delimited by ';' (semi-colon)

	A	B	C	D	E
1	STATEMENT_NAME	OPERATOR_NAME	OPERATOR_DETAILS	EXECUTION_ENGINE	DATABASE
2	1 2797a12f-9e34-4eae-aaf1-ab1a0d	INSERT	?	COLUMN	
3	2 2797a12f-9e34-4eae-aaf1-ab1a0d	DISTINCT	GROUPING: TO_INT(TO_NCHAR(PE.PATIENT_EMPI_ID))	ROW	
4	3 2797a12f-9e34-4eae-aaf1-ab1a0d	HASH JOIN	HASH BUILD: RIGHT, JOIN CONDITION: I.START = I.START AND TO_INT(TO_NCHAR(PE.ROW	GROUPING: TO_INT(TO_NCHAR(PE.PATIENT_EMPI_ID)), I.INTERACTION_ID, IFNULL(I.ROW	
5	4 2797a12f-9e34-4eae-aaf1-ab1a0d	DISTINCT			
6	5 2797a12f-9e34-4eae-aaf1-ab1a0d	HASH JOIN	HASH BUILD: RIGHT, JOIN CONDITION: BI.INTERACTION_ID = I.INTERACTION_ID	ROW	
7	6 2797a12f-9e34-4eae-aaf1-ab1a0d	COLUMN SEARCH	BI.INTERACTION_ID	COLUMN	
8	7 2797a12f-9e34-4eae-aaf1-ab1a0d	COLUMN VIEW		COLUMN	
9	8 2797a12f-9e34-4eae-aaf1-ab1a0d	HAVING	DAYS_BETWEEN(CRITREF.START, I.START) <= 31 AND DAYS_BETWEEN(CRITREF.START, ROW		
10	9 2797a12f-9e34-4eae-aaf1-ab1a0d	DISTINCT	GROUPING: sap.hhp.fnd::INTERACTION_DETAILS.INTERACTION_DETAILS_ID, sap.hhp.fnd::	ROW	
11	10 2797a12f-9e34-4eae-aaf1-ab1a0d	HASHED RANGE JOIN	HASH BUILD: LEFT, HASH CONDITION: TO_INT(TO_NCHAR(PE.PATIENT_EMPI_ID)) = C.ROW		
12	11 2797a12f-9e34-4eae-aaf1-ab1a0d	COLUMN SEARCH	sap.hhp.fnd::INTERACTION_DETAILS.INTERACTION_DETAILS_ID, I.INTERACTION_ID, COLUMN		
13	12 2797a12f-9e34-4eae-aaf1-ab1a0d	JOIN	JOIN CONDITION: (INNER) PE.PATIENT_ID = I.PATIENT_ID, (INNER) PE.ORG_ID = I.ORG_ID	COLUMN	
14	13 2797a12f-9e34-4eae-aaf1-ab1a0d	COLUMN TABLE		COLUMN	
15	14 2797a12f-9e34-4eae-aaf1-ab1a0d	COLUMN TABLE		COLUMN	
16	15 2797a12f-9e34-4eae-aaf1-ab1a0d	COLUMN TABLE	FILTER CONDITION: I.ITYPE_CS_CODE = n'Procedure, Performed' AND I.START > TO_DATE('2021-01-01', 'YYYY-MM-DD')	COLUMN	
17	16 2797a12f-9e34-4eae-aaf1-ab1a0d	COLUMN SEARCH	VSC.Code, IFNULL(UCSM.UmlsCodeSystem, VSC.CodeSystem)	COLUMN	
18	17 2797a12f-9e34-4eae-aaf1-ab1a0d	JOIN	JOIN CONDITION: (LEFT OUTER) VSC.CodeSystem = UCSM.CodeSystemName	COLUMN	
19	18 2797a12f-9e34-4eae-aaf1-ab1a0d	COLUMN TABLE	FILTER CONDITION: VSC.ValueSetOID = n'2.16.840.1.113883.3.526.3.1098'	COLUMN	
20	19 2797a12f-9e34-4eae-aaf1-ab1a0d	COLUMN TABLE		COLUMN	
21	20 2797a12f-9e34-4eae-aaf1-ab1a0d	MATERIALIZED UNION ALL	(TO_INT(TO_NCHAR(PE.PATIENT_EMPI_ID)), TO_INT(TO_NCHAR(PE.PATIENT_EMPI_ID)), ROW		
22	21 2797a12f-9e34-4eae-aaf1-ab1a0d	DISTINCT	GROUPING: TO_INT(TO_NCHAR(PE.PATIENT_EMPI_ID)), I.INTERACTION_ID, IFNULL(I.ROW		
23	22 2797a12f-9e34-4eae-aaf1-ab1a0d	FILTER	CRITREF.START < '2021-01-01' AND CRITREF.START < '9999-12-31 00:00:00.0000000' AND ROW		
24	23 2797a12f-9e34-4eae-aaf1-ab1a0d	WINDOW	WINDOW FUNC: ROW_NUMBER() PARTITIONING: TO_NCHAR(PE.PATIENT_EMPI_ID) ROW		
25	24 2797a12f-9e34-4eae-aaf1-ab1a0d	COLUMN SEARCH	I.INTERACTION_ID, CRITREF.START, I.END, PE.PATIENT_EMPI_ID (OLTP SEARCH)	COLUMN	
26	25 2797a12f-9e34-4eae-aaf1-ab1a0d	JOIN	JOIN CONDITION: (INNER) S1313.VALUE_CS_CODE = VSC.Code, (INNER) S1313.VALUE_CS_CODE = VSC.Code	COLUMN	
27	26 2797a12f-9e34-4eae-aaf1-ab1a0d	COLUMN TABLE		COLUMN	
28	27 2797a12f-9e34-4eae-aaf1-ab1a0d	COLUMN TABLE	FILTER CONDITION: I.ITYPE_CS_CODE = n'Encounter, Performed' AND I.START >= TO_DATE('2021-01-01', 'YYYY-MM-DD')	COLUMN	

1. Explain Plan IV

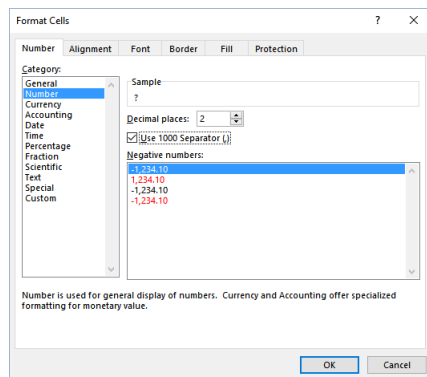


- **DELETE columns:**

- > STATEMENT_NAME
- > DATABASE_NAME
- > SUBTREE_COST
- > OPERATOR_ID
- > PARENT_OPERATOR_ID
- > LEVEL;POSITION
- > HOST
- > PORT
- > TIMESTAMP
- > CONNECTION

- **FORMAT cells**

- > Vertical alignment TOP
- > OPERATOR_DETAILS:
WRAP TEXT
- > TABLE_SIZE/OUTPUT_SIZE:
NUMERIC, 2 decimals



2. Reading Explain Plan

	A	B	C	D	E	F	G	H	I	J
		OPERATOR_NAME	OPERATOR_DETAILS	EXECUTION_ENGINE	SCHEMA_NAME	TABLE_NAME	TABLE_TYPE	TABLE_SIZE	OUTPUT_SIZE	
1	1	INSERT	?	COLUMN	SAP_ASCO_CMA	sap.hc.asco.cancerling.cml.db::MEASURE_P/	COLUMN TABLE	?	?	
2	2	DISTINCT	GROUPING: TO_INT(TO_NCHAR(PE.PATIENT_EMPI_ID))	ROW	?	?	?	?	1.00	
3	3	HASH JOIN	HASH BUILD: RIGHT, JOIN CONDITION: I.START = I.START AND TO_INT(TO_NCHAR(PE.PATIENT_EMPI_ID)) = TO_INT(TO_NCHAR(PE.PATIENT_EMPI_ID)) AND (NULL MATCH) I.INTERACTION_ID = I.INTERACTION_ID AND (NULL MATCH) IFNULL(I.END, '9999-12-31') = IFNULL(I.END, '9999-12-31')	ROW	?	?	?	?	1.00	
4	4	DISTINCT	GROUPING: TO_INT(TO_NCHAR(PE.PATIENT_EMPI_ID)), I.INTERACTION_ID, IFNULL(I.END, '9999-12-31')	ROW	?	?	?	?	1.00	
5	5	HASH JOIN	HASH BUILD: RIGHT, JOIN CONDITION: BI.INTERACTION_ID = I.INTERACTION_ID	ROW	?	?	?	?	200.00	
6	6	COLUMN SEARCH	BI.INTERACTION_ID	COLUMN	?	?	?	10.00	10.00	
7	7	COLUMN VIEW		COLUMN	SAP_ASCO_CMA	_SYS_SS_CE_327010_140286155681024_3_RE	CALCULATION VIEW	10.00	10.00	
8	8	HAVING	DAYS_BETWEEN(CRITREF.START, I.START) <= 31 AND DAYS_BETWEEN(CRITREF.START, I.START) >= 0	ROW	?	?	?	?	1.00	
9	9	DISTINCT	GROUPING: sap.hhp.fnd::INTERACTION_DETAILS.INTERACTION_DETAILS_ID, sap.hhp.fnd::PATIENT_EMPL.ORG_ID, sap.hhp.fnd::PATIENT_EMPI.SOURCE, sap.hhp.fnd::PATIENT_EMPI.PATIENT_ID, CRITREF.INTERACTION_ID, CRITREF.START, CRITREF.END	ROW	?	?	?	?	1.00	
10	10	HASHED RANGE JOIN	HASH BUILD: LEFT, HASH CONDITION: TO_INT(TO_NCHAR(PE.PATIENT_EMPI_ID)) = CRITREF.PATIENT_EMPI_ID, RANGE CONDITION: I.START > CRITREF.START	ROW	?	?	?	?	1.00	
11	11	COLUMN SEARCH	sap.hhp.fnd::INTERACTION_DETAILS.INTERACTION_DETAILS_ID, I.INTERACTION_ID, I.START, I.END, sap.hhp.fnd::PATIENT_EMPL.ORG_ID, sap.hhp.fnd::PATIENT_EMPI.SOURCE, sap.hhp.fnd::PATIENT_EMPI.PATIENT_ID, PE.PATIENT_EMPI_ID (OLTP SEARCH)	COLUMN	?	?	?	?	60,205,533,596,837,900.00	
12	12	JOIN	JOIN CONDITION: (INNER) PE.PATIENT_ID = I.PATIENT_ID, (INNER) PE.ORG_ID = I.ORG_ID, (INNER) sap.hhp.fnd::INTERACTION_DETAILS.INTERACTION_ID = sap.hhp.fnd::INTERACTIONS.INTERACTION_ID, (INNER) S1311.VALUE_CS_CODE = VSC.Code, (INNER) S1311.VALUE_CODING_SYSTEM = IFNULL(UCSM.UmlsCodeSystem, VSC.CodeSystem)	COLUMN	?	?	?	?	60,205,533,596,837,900.00	
13	13	COLUMN TABLE		COLUMN	SAP_HHP	sap.hhp.fnd::INTERACTION_DETAILS	COLUMN TABLE	42.79	42.79	
14	14	COLUMN TABLE		COLUMN	SAP_HHP	sap.hhp.fnd::PATIENT_EMPI	COLUMN TABLE	3.51	3.51	
15	15	COLUMN TABLE	FILTER CONDITION: I.ITYPE_CS_CODE = n'Procedure, Performed' AND I.START > TO_TIMESTAMP('2000-01-01') AND I.START >= TO_TIMESTAMP('0001-01-01 00:00:00.0000000') AND I.START >= TO_TIMESTAMP('2000-01-01') AND I.START <= TO_TIMESTAMP('2021-01-01') AND I.START <= TO_TIMESTAMP('9999-12-31 00:00:00.0000000')	COLUMN	SAP_HHP	sap.hhp.fnd::INTERACTIONS	COLUMN TABLE	16.89	2,410,3264781491002	
16	16	COLUMN SEARCH	VSC.Code, IFNULL(UCSM.UmlsCodeSystem, VSC.CodeSystem)	COLUMN	?	?	?	?	13,761,264,822,134,300.00	
17	17									

2. Reading Explain Plan

	B	C	D	E	F	G	H	I	J	K
		OPERATOR_NAME	OPERATOR_DETAILS	EXECUTION	SCHEMA_NAME	TABLE_NAME	TABLE_TYPE	TABLE_SIZE	OUTPUT_SIZE	
4		1 INSERT	?	COLUMN	SAP_ASCO_CMA	sap.hc.asco.cancerling.cml.db::MEASURE_PATIENTS	COLUMN TABLE	?	?	
5		2 DISTINCT	GROUPING: TO_INT(TO_NCHAR(PE.PATIENT_EMPL_ID))	ROW	?	?	?	?	?	
6		3 HASH JOIN	HASH BUILD: RIGHT, JOIN CONDITION: I.START = I.START AND TO_INT(TO_NCHAR(PE.PATIENT_EMPL_ID)) = TO_INT(TO_NCHAR(PE.PATIENT_EMPL_ID)) AND (NULL MATCH) I.INTERACTION_ID = I.INTERACTION_ID AND (NULL MATCH) IFNULL(I.END, '9999-12-31') = IFNULL(I.END, '9999-12-31')	ROW	?	?	?	?	?	
7		4 DISTINCT	GROUPING: TO_INT(TO_NCHAR(PE.PATIENT_EMPL_ID)), I.INTERACTION_ID, IFNULL(I.END, '9999-12-31')	ROW	?	?	?	?	?	
8		5 HASH JOIN	HASH BUILD: RIGHT, JOIN CONDITION: BI.INTERACTION_ID = I.INTERACTION_ID	ROW	?	?	?	?	?	
9		6 COLUMN SEARCH	BI.INTERACTION_ID	COLUMN	?	?	?	?	10.00	
10		7 COLUMN VIEW		COLUMN	SAP_ASCO_CMA	_SYS_SS_CE_327010_140286155681024_3_RET	CALCULATION VIEW	?	10.00	
11		8 HAVING	DAYS_BETWEEN(CRITREF.START, I.START) <= 31 AND DAYS_BETWEEN(CRITREF.START, I.START) >= 0	ROW	?	?	?	?	?	
12		9 DISTINCT	GROUPING: sap.hhp.fnd::INTERACTION_DETAILS.INTERACTION_DETAILS_ID, sap.hhp.fnd::PATIENT_EMPL_ORG_ID, sap.hhp.fnd::PATIENT_EMPL_SOURCE, sap.hhp.fnd::PATIENT_EMPL_PATIENT_ID, CRITREF.INTERACTION_ID, CRITREF.START, CRITREF.END	ROW	?	?	?	?	?	
13		10 HASHED RANGE JOIN	HASH BUILD: LEFT, HASH CONDITION: TO_INT(TO_NCHAR(PE.PATIENT_EMPL_ID)) = CRITREF.PATIENT_EMPL_ID, RANGE CONDITION: I.START > CRITREF.START	ROW	?	?	?	?	?	
14		11 COLUMN SEARCH	sap.hhp.fnd::INTERACTION_DETAILS.INTERACTION_DETAILS_ID, I.INTERACTION_ID, I.START, I.END, sap.hhp.fnd::PATIENT_EMPL_ORG_ID, sap.hhp.fnd::PATIENT_EMPL_SOURCE, sap.hhp.fnd::PATIENT_EMPL_PATIENT_ID, PE.PATIENT_EMPL_ID (OLTP SEARCH)	COLUMN	?	?	?	?	60,205,533.5	
15		12 JOIN	JOIN CONDITION: (INNER) PE.PATIENT_ID = I.PATIENT_ID, (INNER) PE.ORG_ID = I.ORG_ID, (INNER) sap.hhp.fnd::INTERACTION_DETAILS.INTERACTION_ID = sap.hhp.fnd::INTERACTIONS.INTERACTION_ID, (INNER) S1311.VALUE_CS_CODE = VSC.Code, (INNER) S1311.VALUE_CODING_SYSTEM = IFNULL(UCSM.UmlsCodeSystem, VSC.CodeSystem)	COLUMN	?	?	?	?	60,205,533.5	
16		13 COLUMN TABLE		COLUMN	SAP_HHP		COLUMN TABLE	?	42.79	
17		14 COLUMN TABLE		COLUMN	SAP_HHP		COLUMN TABLE	?	3.51	
18		15 COLUMN TABLE	FILTER CONDITION: I.ICTYPE_CS_CODE = n'Procedure, Performed' AND I.START > TO_TIMESTAMP('2000-01-01') AND I.START >= TO_TIMESTAMP('0001-01-01 00:00:00.0000000') AND I.START >= TO_TIMESTAMP('2000-01-01') AND I.START <= TO_TIMESTAMP('2021-01-01') AND I.START <= TO_TIMESTAMP('9999-12-31 00:00:00.0000000')	COLUMN	SAP_HHP		COLUMN TABLE	?	16.89 2,410,326478	
19		16 COLUMN SEARCH	VSC.Code, IFNULL(UCSM.UmlsCodeSystem, VSC.CodeSystem)	COLUMN	?	?	?	?	13,761,264.8	

Why to we see TO_INT() and
TO_NVARCHAR() conversions here?
On JOIN columns??

Data type conversion?
Are those in the SQL code?

ROW Engine execution - WHY?

Filter condition type conversion
computed on constant or on column
values?



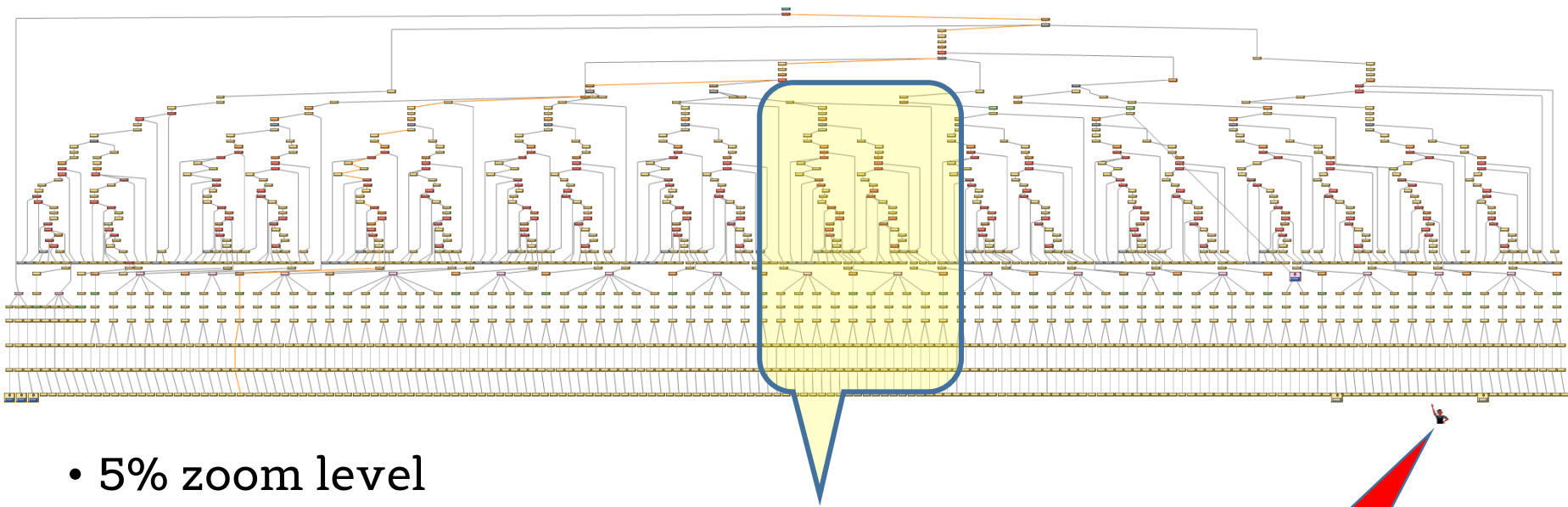
2. Reading Explain Plan

- With the notes and questions at hand, the next step is typically:
 - Reading the SQL
 - Reviewing the models
- Important to break down the scenario into smallest possible pieces in order to explain each of the noted issues
- Running traces (`sqlopt`, statement rewrite, etc.) is usually **not** helpful for large scenarios – too much irrelevant detail information
- Once small scenario is identified, find reason for current problem (see examples later).

3. PlanViz

- Good tool for identifying which part of a query consumes most time
- Good for comparing statements (runtime & memory usage)
- Good for understanding data transfer in multi-node systems
- Good for seeing execution pattern
- Not so good for finding modelling issues – it's hard to map back to SQL or Information View
- Can easily be overwhelming

3. PlanViz

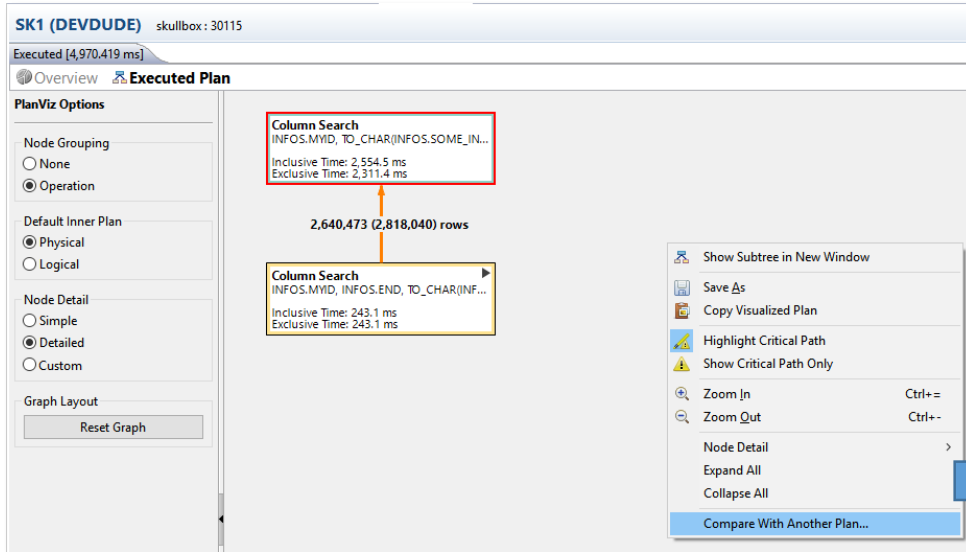


- 5% zoom level
- See: there are repeating pattern!
 - > Why? What are those for in your query?

You are
here!

3. PlanViz remarks

- Make use of the “compare plans” feature



SK1 (DEVDUDE) skullbox : 30115

Executed [4,970,419 ms]

Overview Executed Plan

PlanViz Options

Node Grouping

☐ None

☒ Operation

Default Inner Plan

☒ Physical

☐ Logical

Node Detail

☐ Simple

☒ Detailed

☐ Custom

Graph Layout

Reset Graph

Column Search
INFOS.MYID, TO_CHAR(INFOS.SOME_IN...
Inclusive Time: 2,554.5 ms
Exclusive Time: 2,311.4 ms

2,640,473 (2,818,040) rows

Column Search
INFOS.MYID, INFOS.END, TO_CHAR(INF...
Inclusive Time: 243.1 ms
Exclusive Time: 243.1 ms

Show Subtree in New Window

Save As

Copy Visualized Plan

Highlight Critical Path

Show Critical Path Only

Zoom In Ctrl+=

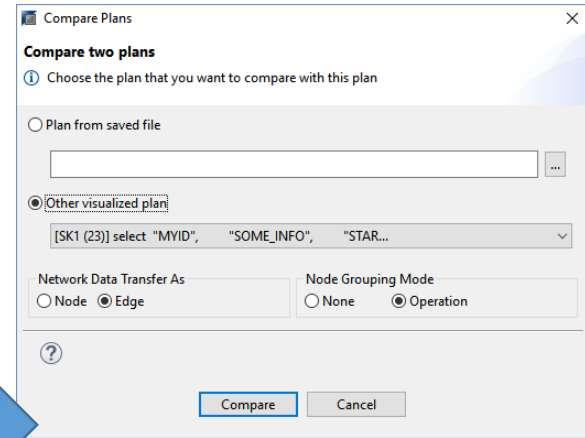
Zoom Out Ctrl+-

Node Detail >

Expand All

Collapse All

Compare With Another Plan...



Compare Plans

Compare two plans

Choose the plan that you want to compare with this plan

☐ Plan from saved file

☒ Other visualized plan

[SK1 (23)] select "MYID", "SOME_INFO", "STAR...

Network Data Transfer As

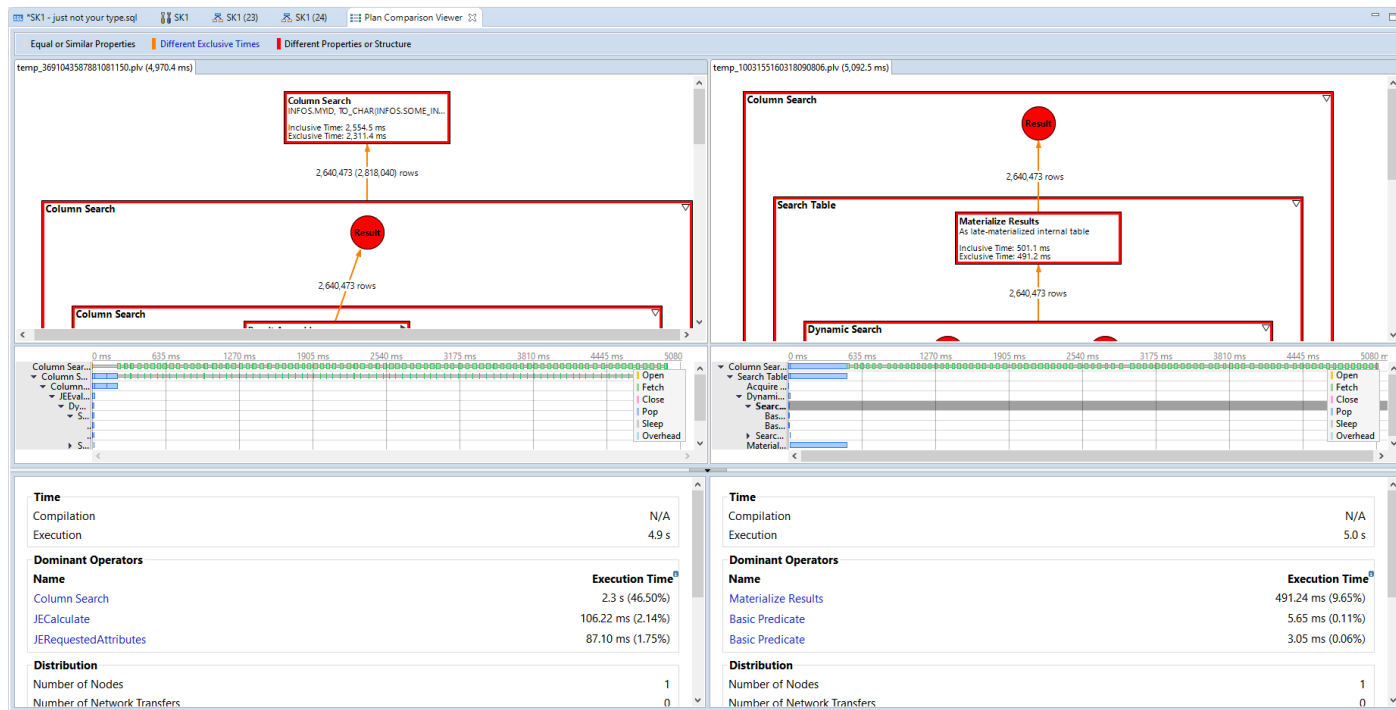
☐ Node ☒ Edge

Node Grouping Mode

☐ None ☒ Operation

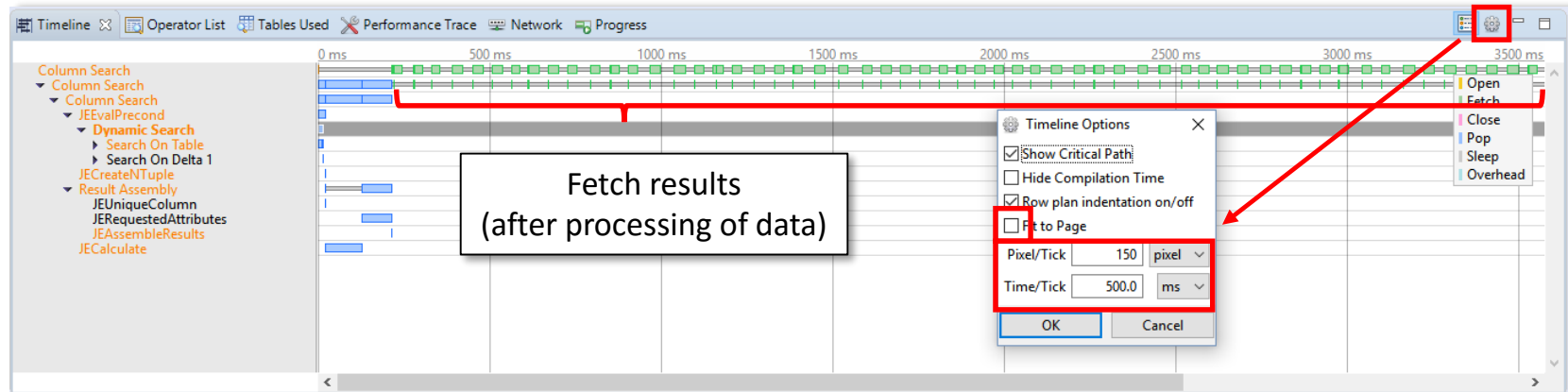
Compare Cancel

3. PlanViz Remarks



3. PlanViz Remarks

- When using the timeline, make sure to understand what you're seeing...
- “green” top lane is fetching results after all processing
- Set scale of graph, especially when comparing two graphs



Just not your type...

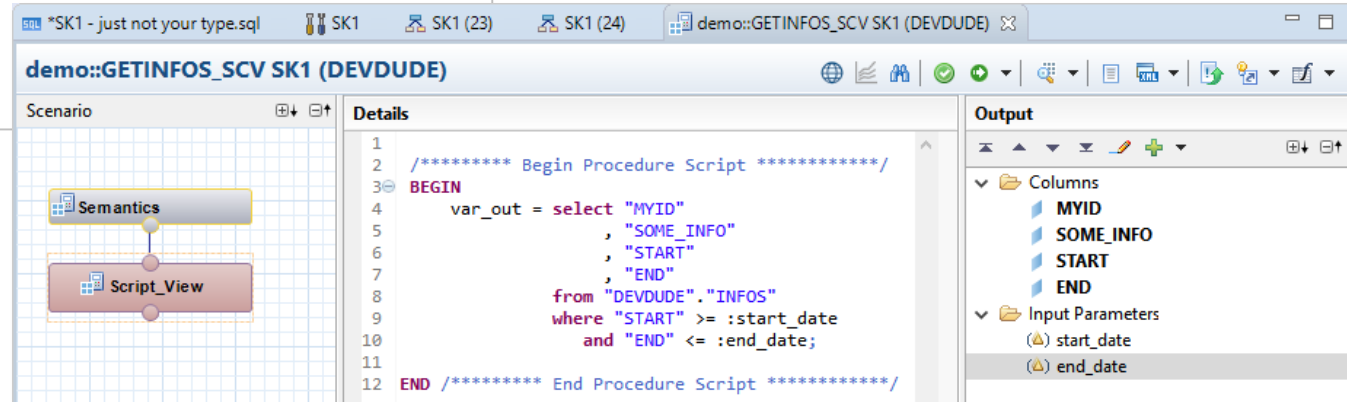
- Let's have a look at three ways of running a query:

```
select  "MYID",  
        "SOME_INFO",  
        "START",  
        "END"  
from infos  
where  
    "START" >= '2017-01-01'  
and "END" <= '2018-10-31';
```

```
drop function getMyInfos;  
create function getMyInfos  
    (IN startDate daydate, IN endDate daydate)  
returns table ("MYID" NVARCHAR (1024), "SOME_INFO" NVARCHAR(20),  
              "STARTD" date, "ENDD" date)  
as  
begin  
  
return  
    select "MYID",  
           "SOME_INFO",  
           "START" as "STARTD",  
           "END" as "ENDD"  
  
    from infos  
    where  
        "START" >= :startDate  
    and "END" <= :enddate;  
  
end;  
  
select * from getMyInfos ('2017-01-01', '2018-10-31');
```

Just not your type...

```
SELECT
    "MYID",
    "SOME_INFO",
    "START",
    "END"
FROM "_SYS_BIC"."demo/GETINFOS_SCV"
('PLACEHOLDER' =
    ('$$start_date$$', '2017-01-01'),
'PLACEHOLDER' =
    ('$$end_date$$', '2018-10-31') );
```



The screenshot shows the SAP Studio IDE interface. The top toolbar includes icons for SQL, Scenario, and other development tools. The main window is titled "demo::GETINFOS_SCV SK1 (DEVDUDE)". It is divided into three panes:

- Scenario:** A diagram showing a flow from a "Semantics" node to a "Script_View" node.
- Details:** A code editor showing the following SQL script:

```
1 /***** Begin Procedure Script *****/
2 BEGIN
3     var_out = select "MYID"
4                   , "SOME_INFO"
5                   , "START"
6                   , "END"
7                   from "DEVDUDE"."INFOS"
8                   where "START" >= :start_date
9                     and "END" <= :end_date;
10
11
12 END /***** End Procedure Script *****/
```
- Output:** A pane showing the results of the script execution. It includes a "Columns" section with the following fields: MYID, SOME_INFO, START, and END. Below this is an "Input Parameters" section with two parameters: start_date and end_date.

Just not your type...

- Which one runs fastest?
- Which one uses least memory?
- Why?

query	time	memory
plain SQL	4.5 secs	282 MB
table function	4.6 secs	449 MB
scripted calcview	6.3 secs	591 MB

Just not your type...

Executed [4,608.529 ms]

Overview Executed Plan

Time	
Compilation	N/A
Execution	4.6 s

Dominant Operators

Name	Execution Time [ⓘ]
Materialize Results	85.36 ms (1.85%)
Basic Predicate	7.54 ms (0.16%)
Basic Predicate	4.19 ms (0.09%)

Distribution

Number of Nodes	1
Number of Network Transfers	0

Context

SQL Query	select "MYID", "SOME_INFO", ...
System	skullbox:30103
System Version	1.00.112.07.1477423516
System Compile Type	rel
Memory Allocated	282.2 MByte(s)

Data Flow

Number of Tables Used [ⓘ]	1
Result Record Count	2640473

plain SQL

Executed [6,286.563 ms]

Overview Executed Plan

Time	
Compilation	N/A
Execution	6.2 s

Dominant Operators

Name	Execution Time [ⓘ]
Column Search	2.2 s (35.61%)
JECalculate	259.75 ms (4.13%)
JERequestedAttributes	77.01 ms (1.22%)

Distribution

Number of Nodes	1
Number of Network Transfers	0

Context

SQL Query	select * from getMyInfos ('2017-01-01'; ...
System	skullbox:30103
System Version	1.00.112.07.1477423516
System Compile Type	rel
Memory Allocated	591 MByte(s)

Data Flow

Number of Tables Used [ⓘ]	1
Result Record Count	2640473

table function

Executed [4,496.37 ms]

Overview Executed Plan

Time	
Compilation	N/A
Execution	4.4 s

Dominant Operators

Name	Execution Time [ⓘ]
Column Search	1.9 s (44.38%)
JECalculate	108.46 ms (2.41%)
JERequestedAttributes	77.95 ms (1.73%)

Distribution

Number of Nodes	1
Number of Network Transfers	0

Context

SQL Query	SELECT "MYID", "SOME_INFO", ...
System	skullbox:30103
System Version	1.00.112.07.1477423516
System Compile Type	rel
Memory Allocated	449.7 MByte(s)

Data Flow

Number of Tables Used [ⓘ]	1
Result Record Count	2640473

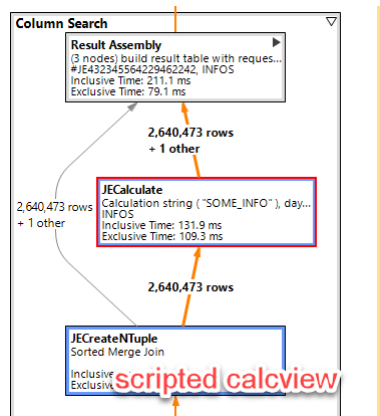
scripted calcview

What are the table function
and the scripted calcview
“calculating” in the JECalculate
POP (PlanOperator)?

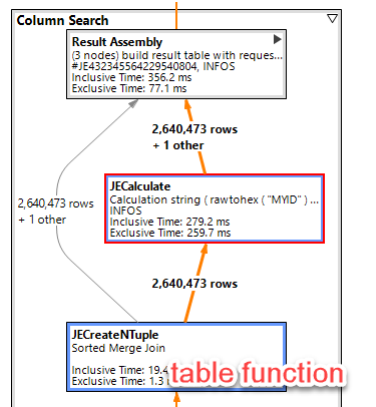
Sponsored by



Just not your type...



Console	Properties	Error Log	Job Log
General			
Details			
Property	Value		
Name	JECalculate		
Schema	DEVDUDE		
Location(s)	skullbox:30103		
Table Name	INFOS		
Operator Info	Calculation string ("SOME_INFO"), daydate ("START") on table INFOS		
Execution Time (Exclusive)	109.26 ms		
Execution Time (Inclusive)	131.919 ms		
Execution Start Time	23.367 ms		
Execution End Time	132.627 ms		
CPU Time (User)	30 ms		



Console	Properties	Error Log	Job Log
General			
Details			
Property	Value		
Name	JECalculate		
Schema	DEVDUDE		
Location(s)	skullbox:30103		
Table Name	INFOS		
Operator Info	Calculation string (rawtohex ("MYID")), daydate ("START") on tabl...		
Execution Time (Exclusive)	259.749 ms		
Execution Time (Inclusive)	279.173 ms		
Execution Start Time	19.867 ms		
Execution End Time	279.616 ms		
CPU Time (User)	30 ms		

Just not your type...

- `string()` and `rawtohex()` are type conversion functions.
- so, there is some kind of conversion happening...
- who ordered that?



Image courtesy of
<https://www.flickr.com/photos/stevendepolo/10444770884/>

Just not your type...

- Apparently we did ☹️

Table Name:		Schema:		Type:
INFOS		DEVDUDE		Column
Columns	Indexes	Further Properties	Runtime Information	
	Name	SQL Data Type	Dimension	Column Store Data Type
1	MYID	VARBINARY	32	RAW
2	SOME_INFO	NVARCHAR	20	STRING
3	START	TIMESTAMP		LONGDATE
4	END	DATE		DAYDATE

	Name	Data type	Length	Scale
1	MYID	VARBINARY	32	
2	SOME INFO	VARCHAR	20	
3	START	DATE		
4	END	DATE		
5	<Click to add>			

```
create function getMyInfos
(IN startDate daydate, IN endDate daydate)
returns table
("MYID" NVARCHAR (1024)
, "SOME INFO" NVARCHAR (20)
, "STARTD" date
, "ENDD" date)
```

Just not your type...

- Differences in types are handled by implicit type conversion
- That takes time and memory
- Also applies to joins

```
EXPLAIN PLAN FOR
select * from getjoinedinfos()
```

	OPERATOR_NAME	OPERATOR_DETAILS
1	COLUMN SEARCH	INFOS.MYID, INFOS.SOME INFO, TO_DATE(INFOS.START), INFOS.ENDD (LATE MATERIALI...
2	JOIN	JOIN CONDITION: (INNER) TO_NCHAR(INFOS.MYID) = BINTONHEX(INFOS.MYID)
3	COLUMN TABLE	FILTER CONDITION: INFOS.START >= TO_TIMESTAMP('2017-01-01') AND INFOS.END <= T...
4	COLUMN TABLE	FILTER CONDITION: INFOS.START >= TO_TIMESTAMP('2017-01-01') AND INFOS.END <= T...

- Use explicit type conversion when necessary and be suspicious of every conversion in Explain plan or PlanViz that you did not put there

Just not your type...

- General SQL know-how and common sense applies to HANA development.
- Pushing computation to the DB layer means understanding what happens there and how to best use it.

Wrinkles in time and date

- A lot of data processing happens related to time or date or both
- SAP HANA provides many well-known functions for that
 - > YEAR(), DAY(), MONTH(), ADD_DAYS(), DAYS_BETWEEN() ...
- One major use case for SAP HANA is to run on top of ABAP table structures – time and date looks different in those: TIMS, DATS are VARCHAR in SQL-land.

Type	Valid Places m	Initial Value	Meaning	ABAP Type
DATS	8	00000000	Date in the format YYYYMMDD	d
TIMS	6	000000	Time in the format HHMMSS	t

Wrinkles in time and date

- Solution approaches:

HANA Date and Time Comparison

This question is answered

Dear All,

I would like ask for your help regarding Date conversion in HANA. In my view I have two columns: DueDate (NVARCHAR(8)) and DueTime (NVARCHAR(8)). These columns have values like this:

DueDate = '20140213'
DueTime = '092345'

I need to decide if an item is overdue. The calculated value should be 'Y' or 'N'. I tried to create a Calculated Column with a conversion like this:

1.) This seemed the simplest way, but I got a syntax error.
`if(date("DueDate")<now() OR (date("DueDate")=now() AND time("DueTime")<now()))<('X';)`

2.) This is more complex with more conversions. I don't get any syntax error, only when I try to display the preview.
`if(date("DueDate")<date(string(now())) OR (date("DueDate")=date(string(now())) AND time(string("DueTime"))<time(string(now()))))<('X';)`

Where is the problem? What am I doing wrong?



replied

Hi Sudarshan,

Somehow my reply from yesterday is not being shown.

Can you please try using `leftstr(string(<date>, 4))` to get the year information.

Regards,



0



replied

`date(component("DATE", 1), component("DATE", 2), component("DATE", 3))`

0

Get the month, year and day from given

Hi Experts,

I am trying to do the leftstring, rightstring functions for given date to get the month and year columns.

like below,

`leftstr("ZZSTUFFINGDATE", 5)`

I am not getting proper results.

se suggest in this case.

ks.

Regards,

Answer by Lars Breddemann

lies



replied

Hi

Move your date into a character based field, in this example we name it lv_date
If your date looks like this: 20150219

You can derive the values as follows: (Define the variables accordingly as character based)
`lv_year = lv_date+0(4). - Result 2015`
`lv_month = lv_date+4(2). - Result 02`
`lv_day = lv_date+6(2) - Result 19`

Then you can do with the variables whatever you like, after they've been split
Regards

sponsored by



Wrinkles in time and date

- Wouldn't it be nice, if SAP HANA would know how to deal with this?
- I bet it does... unfortunately the SQL documentation doesn't contain any reference to DATS or TIMS
- Let's see what functions SAP HANA's SQL parser knows about...

```
> call get_functionmap()
```

Wrinkles in time and date

SQL Result

call get_function_map()

	FUNCTION_NAME	RETURN_TYPE	ARGUMENT_TYPE
193	abap_extract_hour	Integer(&)	NString(&)
194	abap_extract_minute	Integer(&)	NString(&)
195	abap_extract_month	Integer(&)	NString(&)
196	abap_extract_second	Integer(&)	NString(&)
197	abap_extract_year	Integer(&)	NString(&)
198	abap_lower	NString(&)	NString(&)
199	abap_system_timezone	NString(&)	NString(&),NString...
200	abap_system_timezone_fail	NString(&)	NString(&),NString...
201	abap_system_timezone_initial	NString(&)	NString(&),NString...
202	abap_system_timezone_null	NString(&)	NString(&),NString...
203	abap_upper	NString(&)	NString(&)
204	abap_user_timezone	NString(&)	NString(&),NString...
205	abap_user_timezone_fail	NString(&)	NString(&),NString...
206	abap_user_timezone_initial	NString(&)	NString(&),NString...
207	abap_user_timezone_null	NString(&)	NString(&),NString(&)
208	abs	Integer(&)	Integer(&)
209	abs	BigInt(&)	BigInt(&)
210	abs	Decimal(&)	Decimal(&)
211	abs	Real(&)	Real(&)
212	abs	Double(&)	Double(&)
213	abs	SmallDecim...	SmallDecimal(&)
214	abs	Fixed16(&)	Fixed16(&)

	FUNCTION_NAME	RETURN_TYPE	ARGUMENT_TYPE
1777	dates_add_days_fail	NString(&)	NString(&),Integer(...
1778	dates_add_days_initial	NString(&)	NString(&),Integer(...
1779	dates_add_days_null	NString(&)	NString(&),Integer(...
1780	dates_add_days_unchanged	NString(&)	NString(&),Integer(...
1781	dates_add_months	NString(&)	NString(&),Integer(...
1782	dates_add_months_fail	NString(&)	NString(&),Integer(...
1783	dates_add_months_initial	NString(&)	NString(&),Integer(...
1784	dates_add_months_null	NString(&)	NString(&),Integer(...
1785	dates_add_months_unchang...	NString(&)	NString(&),Integer(...
1786	dates_days_between	Integer(&)	NString(&),NString...
1787	dates_from_date	NString(&)	Daydate(&)
1788	dates_from_date_fail	NString(&)	Daydate(&)
1789	dates_from_date_initial	NString(&)	Daydate(&)
1790	dates_from_date_null	NString(&)	Daydate(&)
1791	dates_is_initial	Integer(&)	NString(&)
1792	dates_is_valid	Integer(&)	NString(&)
1793	dates_tims_to_tstmp	Fixed8(&)	NString(&),NString...
1794	dates_tims_to_tstmp	Fixed8(&)	NString(&),NString...
1795	dates_tims_to_tstmp_fail	Fixed8(&)	NString(&),NString...
1796	dates_tims_to_tstmp_fail	Fixed8(&)	NString(&),NString...
1797	dates_tims_to_tstmp_initial	Fixed8(&)	NString(&),NString...
1798	dates_tims_to_tstmp_initial	Fixed8(&)	NString(&),NString...

	FUNCTION_NAME	RETURN_TYPE	ARGUMENT_TYPE
1810	dates_to_date	Daydate(&)	NString(&),Daydate...
1811	dates_to_date	Daydate(&)	NString(&),Daydate...
1812	dates_to_date_fail_empty	Daydate(&)	NString(&)
1813	dates_to_date_fail_empty	Daydate(&)	NString(&),Daydate...
1814	dates_to_date_fail_invalid	Daydate(&)	NString(&),Daydate...

Wrinkles in time and date

- Looks promising, so let's try this out:

```
select top 5
    "HZDAT",
    "FADAT",
    "BUDAT"
from CE1IDEA
```

	HZDAT	FADAT	BUDAT
1	20110802	20070418	20070418
2	20110802	20070418	20070418
3	20110802	20070418	20070418
4	20110802	20070418	20070418
5	20110802	20070418	20070418

SQL Result

```
select top 5
    to_integer(left("HZDAT", 4)) as hzdate_yr ,
    hzdat,
    to_integer(left("FADAT", 4)) as fadate_yr ,
    "FADAT",
    to_integer(left("BUDAT", 4)) as budate_yr ,
    "BUDAT"
from CE1IDEA
```

	HZDATE_YR	HZDAT	FADATE_YR	FADAT	BUDATE_YR	BUDAT
1	2011	20110802	2007	20070418	2007	20070418
2	2011	20110802	2007	20070418	2007	20070418
3	2011	20110802	2007	20070418	2007	20070418
4	2011	20110802	2007	20070418	2007	20070418
5	2011	20110802	2007	20070418	2007	20070418

SQL Result

```
select top 5
    abap_extract_year("HZDAT" ) as hzdate_yr ,
    hzdat,
    abap_extract_year("FADAT" ) as fadate_yr ,
    "FADAT",
    abap_extract_year("BUDAT") as budate_yr ,
    "BUDAT"
from CE1IDEA
```

	HZDATE_YR	HZDAT	FADATE_YR	FADAT	BUDATE_YR	BUDAT
1	2011	20110802	2007	20070418	2007	20070418
2	2011	20110802	2007	20070418	2007	20070418
3	2011	20110802	2007	20070418	2007	20070418
4	2011	20110802	2007	20070418	2007	20070418
5	2011	20110802	2007	20070418	2007	20070418

Wrinkles in time and date

- What's the benefit? Performance? Memory?

Executed [5.864 ms] Executed [4.266 ms] Executed [5.449 ms]

Overview Executed Plan

Time		Context	
Compilation	N/A	SQL Query	select top 5 to_nvarchar(add_da...
Execution	5.45 ms	System	skullbox:30103
		System Version	1.00.112.07.1477423516
		System Compile Type	rel
		Memory Allocated	2.1 MByte(s)
Dominant Operators		Data Flow	
Name	Execution Time	Number of Tables Used	Result Record Count
JECalculate	0.38 ms (7.05%)	1	5
Column Search	0.29 ms (5.36%)		
Search	0.26 ms (4.68%)		
Distribution			
Number of Nodes	1		
Number of Network Transfers	0		

```

select top 5
  to_nvarchar(add_days(to_date("HZDAT", "YYYYMMDD"), 128), "YYYYMMDD") as hzdate_add,
  hzdat,
  to_nvarchar(add_days(to_date("FADAT", "YYYYMMDD"), 128), "YYYYMMDD") as fadate_add,
  "FADAT",
  to_nvarchar(add_days(to_date("BUDAT", "YYYYMMDD"), 128), "YYYYMMDD") as budate_add,
  "BUDAT"
from CE1IDEA
  
```

Executed [4.518 ms] Executed [4.755 ms] Executed [4.206 ms]

Overview Executed Plan

Time		Context	
Compilation	N/A	SQL Query	select top 5 dats_add_days("HZD...
Execution	4.21 ms	System	skullbox:30103
		System Version	1.00.112.07.1477423516
		System Compile Type	rel
		Memory Allocated	2.1 MByte(s)
Dominant Operators		Data Flow	
Name	Execution Time	Number of Tables Used	Result Record Count
JECalculate	0.38 ms (9.06%)	1	5
Search	0.25 ms (6.04%)		
Column Search	0.22 ms (5.14%)		
Distribution			
Number of Nodes	1		
Number of Network Transfers	0		

```

select top 5
  dats_add_days("HZDAT", 128) as hzdate_add,
  hzdat,
  dats_add_days("FADAT", 128) as fadate_add,
  "FADAT",
  dats_add_days("BUDAT", 128) as budate_add,
  "BUDAT"
from CE1IDEA
  
```

- None of the above!
- But less written code, easier to understand, less prone to mistakes

Wrinkles in time and date

- List of functions

FUNCTION NAME	RETURN TYPE
abap_extract_day	Integer(&)
abap_extract_hour	Integer(&)
abap_extract_minute	Integer(&)
abap_extract_month	Integer(&)
abap_extract_second	Integer(&)
abap_extract_year	Integer(&)
DATE	
dates_is_valid	Integer(&)
dates_add_days	NString(&)
dates_add_months	NString(&)
dates_days_between	Integer(&)
dates_from_date	NString(&)
dates_is_initial	Integer(&)
dates_to_date	Daydate(&)
dates_tims_to_tstmp	Fixed8(&)
dates_tims_to_tstmpl	Fixed12(&)
TIME	
tims_is_valid	Integer(&)
tims_from_time	NString(&)
tims_to_int	Integer(&)
tims_to_time	Secondtime(&)

TIMESTAMP	
tstmp_is_valid	Integer(&)
tstmp_add_seconds	Fixed8(&)
tstmp_seconds_between	Fixed8(&)
tstmp_current_utctimestamp	Fixed8(&)
tstmp_from_seconddate	Fixed8(&)
tstmp_to_dates	NString(&)
tstmp_to_dst	NString(&)
tstmp_to_seconddate	Seconddate(&)
tstmp_to_tims	NString(&)
TIMESTAMP LONG	
tstmpl_is_valid	Integer(&)
tstmpl_add_seconds	Fixed12(&)
tstmpl_current_utctimestamp	Fixed12(&)
tstmpl_from_timestamp	Fixed12(&)
tstmpl_seconds_between	Fixed12(&)
tstmpl_to_dates	NString(&)
tstmpl_to_dst	NString(&)
tstmpl_to_timestamp	Longdate(&)
tstmpl_to_tims	NString(&)

Wrinkles in time and date

- If SQL code looks convoluted and ugly, it's probably not the best possible code.
- Look for features/commands, that can help with your task.

That DUMMY has to go

Did you recently wrote/read code like this?

```
create procedure pt_selinto (IN loops INT)
language sqlscript
reads sql data
as
begin
    declare cur_date date;
    declare i int;
    declare j int;

    for i IN 0 .. :loops do
        select current_date into cur_date
        from dummy;

        select :i into j from dummy;

    end for;

    select :cur_date, :j from dummy;

end;
```



Image courtesy of essentialbaby.com:
<http://www.essentialbaby.com.au/content/dam/images/2/8/7/r/0/image.related.articleLeadwide.620x349.287qh.png/1351142385313.jpg>

That DUMMY has to go

Use direct assignments instead:

```
create procedure pt_selinto (IN loops INT)
language sqlscript
reads sql data
as
begin
    declare cur_date date;
    declare i int;
    declare j int;

    for i IN 0 .. :loops do
        select current_date into cur_date
        from dummy;

        select :i into j from dummy;

    end for;

    select :cur_date, :j from dummy;

end;
```

```
create procedure pt_directassign (IN loops INT)
language sqlscript
reads sql data
as
begin
    declare cur_date date;
    declare i int;
    declare j int;

    for i IN 0 .. :loops do
        cur_date := current_date;

        j := :i;

    end for;

    select :cur_date, :j from dummy;

end;
```

That DUMMY has to go

Let's see the difference...

```
call pt_selinto(1000);
```

```
/*  
Statement 'call pt_selinto(1000)'  
successfully executed in 782 ms 480 µs  
(server processing time: 274 ms 952 µs)  
  
successfully executed in 608 ms 310 µs  
(server processing time: 267 ms 911 µs)  
  
successfully executed in 617 ms 800 µs  
(server processing time: 279 ms 956 µs)  
*/
```

```
call pt_directassign(1000);
```

```
/*  
Statement 'call pt_directassign(1000)'  
successfully executed in 362 ms 55 µs  
(server processing time: 1 ms 799 µs)  
  
successfully executed in 352 ms 704 µs  
(server processing time: 1 ms 638 µs)  
  
successfully executed in 340 ms 70 µs  
(server processing time: 2 ms 86 µs)  
*/
```

Direct assignment procedure **2 times faster** and **279 times less CPU time usage**.

Contributing factors: SQL parsing, DUMMY table access, tuple creation, function evaluation, return result set...

That DUMMY has to go

- Direct assignments are faster and use less memory
- Direct assignments are easier to read and understand
- As of SPS 12 all scalar SQL functions supported (SPS 11 had a few exceptions, e.g. `HASH_SHA256()`)
- Always good candidates for additional code cleanup when changing the code anyhow

That DUMMY has to go

- Pushing computation to the DB layer means understanding what happens there and how to best use it.
- Don't rely on (workaround) patterns you learned with early HANA revisions

Performance left out(er)

- One group of performance issues are the unused optimizations and the 'too much work done' cases.
- Rather common here:

UNNECESSARY EXECUTION OF JOINS



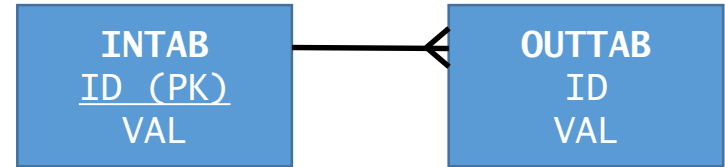
Image courtesy
<http://maxpixel.freegreatpicture.com/Game-Asset-Call-Comic-Horror-Flee-Fear-Man-Fright-1296117>

sponsored by



Performance left out(er)

- When is a join unnecessary?
Whenever it cannot change the result set.
- Example: left outer join
- “return all rows from INTAB with matching OUTTAB rows or with NULL values”
- ➔ INTAB cardinality is minimum for result set cardinality
- Result set would not change if we only ask for data from INTAB and OUTTAB would have max. 1 matching row for every INTAB row (1:[0,1] cardinality).



Performance left out(er)

Example 1

```
drop table intab;
create column table intab (id int primary key,
                           val nvarchar(20));

drop table outtab;
create column table outtab (id int,
                           val nvarchar(20));

insert into intab values (1, 'one');
insert into intab values (2, 'two');
insert into intab values (3, 'three');
insert into intab values (4, 'four');
insert into intab values (5, 'five');
insert into intab values (6, 'six');
insert into intab values (7, 'seven');
insert into intab values (8, 'eight');
insert into intab values (9, 'nine');
insert into intab values (10, 'ten');

insert into outtab (select * from intab);

select * from intab;
Select * from outtab;
```

Performance left out(er)

Example 2

```
-- let's do an outer join

select count(*)
from intab i
     left outer join outtab o
       on i.id = o.id;

-- COUNT(*)
-- 10
-- count is correct, for every ID in INTAB there is exactly one ID in OUTTAB

/*
OPERATOR_NAME          OPERATOR_DETAILS                                TABLE_NAME TABLE_SIZE  OUTPUT_SIZE
COLUMN SEARCH          COUNT(*) (LATE MATERIALIZATION, OLTP SEARCH, ENUM_BY: CS_JOIN) ?           ?           1.0
AGGREGATION            AGGREGATION: COUNT(*)                               ?           ?           1.0
JOIN                   JOIN CONDITION: (LEFT OUTER) I.ID = O.ID             ?           ?           10.0
COLUMN TABLE          INTAB                                                  10.0         10.0
COLUMN TABLE          OUTTAB                                                 10.0         10.0
*/
-- we don't select any information from OUTTAB but still have to do the join?
```

Performance left out(er)

Example 3

```
select i.id
from intab i
      left outer one to one join outtab o
      on i.id = o.in_id;
```

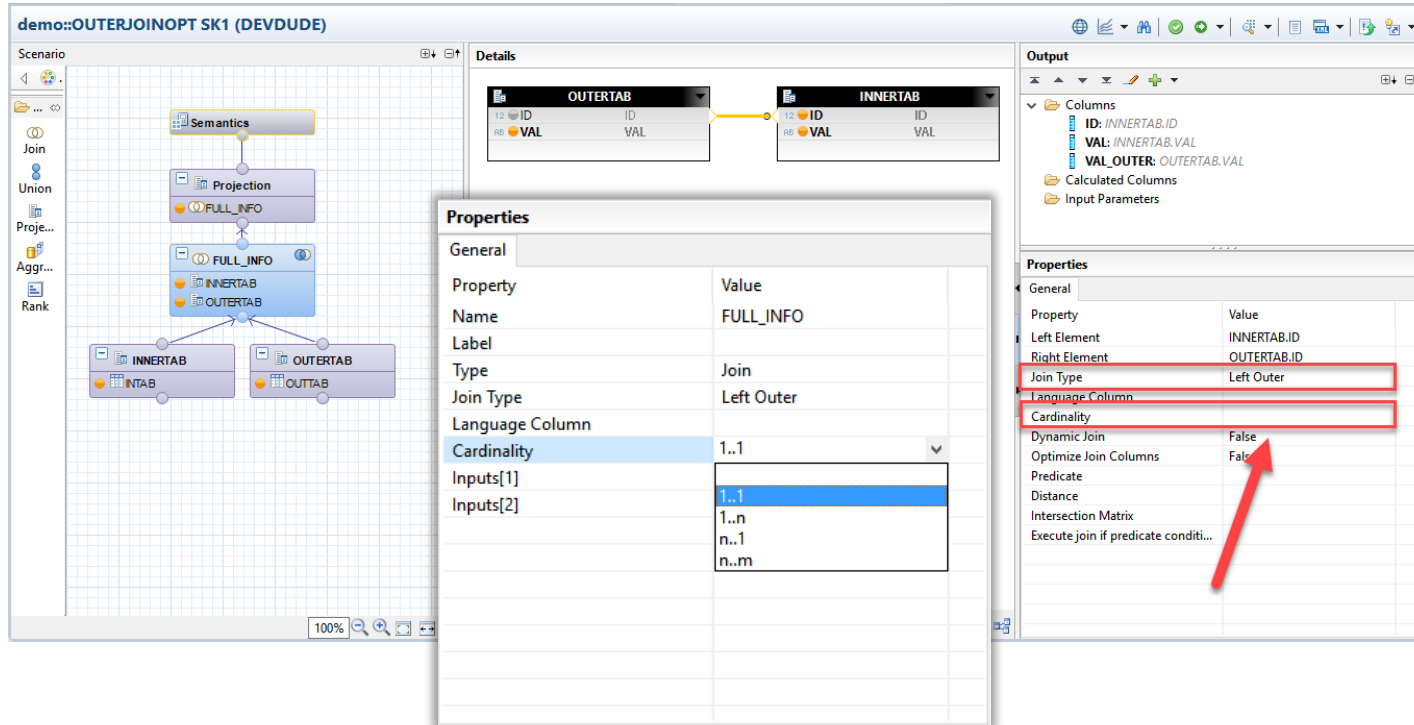
```
/*
OPERATOR_NAME      OPERATOR_DETAILS                                     TABLE_NAME TABLE_SIZE  OUTPUT_SIZE
COLUMN SEARCH      COUNT(*) (LATE MATERIALIZATION, OLTP SEARCH, ENUM_BY: CS_TABLE) ?           ?           1.0
AGGREGATION        AGGREGATION: COUNT(*)                             ?           ?           1.0
COLUMN TABLE      INTAB                                           10.0        10.0
```

By specifying the join-cardinality we can provide the optimizer important information on optimization options
*/

Performance left out(er)

Graphical Calculation View

demo::OUTERJOINOPT SK1 (DEVDUDE)



The screenshot displays the SAP BW Graphical Calculation View editor. The main workspace shows a calculation view named 'demo::OUTERJOINOPT SK1 (DEVDUDE)'. It features a join between two tables: 'INNETAB' and 'OUTERTAB'. The join is configured as a 'Left Outer' join with a cardinality of '1..1'. The 'Properties' window for the join is open, showing the following details:

Property	Value
Name	FULL_INFO
Label	
Type	Join
Join Type	Left Outer
Language Column	
Cardinality	1..1
Inputs[1]	1..1
Inputs[2]	1..n
	n..1
	n..m

The 'Output' window on the right shows the columns of the calculation view:

- Columns:
 - ID: INNETAB.ID
 - VAL: INNETAB.VAL
 - VAL_OUTER: OUTERTAB.VAL
- Calculated Columns
- Input Parameters

A red arrow points to the 'Cardinality' property in the Properties window, highlighting the '1..1' value.

Performance left out(er)

Example 4

```
SELECT
    count(*)
FROM "_SYS_BIC"."demo/OUTERJOINOPT";
/*cardinality unspecified
```

OPERATOR_NAME	OPERATOR_DETAILS	TABLE_NAME	TABLE_SIZE	OUTPUT_SIZE
COLUMN SEARCH	COUNT(*) (LATE MATERIALIZATION, OLTP SEARCH, ENUM_BY: CS_JOIN)	?	?	1.0
AGGREGATION	AGGREGATION: COUNT(*)	?	?	1.0
JOIN	JOIN CONDITION: (LEFT OUTER) INTAB.ID = OUTTAB.ID	?	?	10.0
COLUMN TABLE		INTAB	10.0	10.0
COLUMN TABLE		OUTTAB	10.0	10.0

```
*/
```

```
SELECT
    count(*)
FROM "_SYS_BIC"."demo/OUTERJOINOPT";

/* cardinality 1:1
```

OPERATOR_NAME	OPERATOR_DETAILS	TABLE_NAME	TABLE_SIZE	OUTPUT_SIZE
COLUMN SEARCH	COUNT(*) (LATE MATERIALIZATION, OLTP SEARCH, ENUM_BY: CS_TABLE)	?	?	1.0
AGGREGATION	AGGREGATION: COUNT(*)	?	?	1.0
COLUMN TABLE		INTAB	10.0	10.0

```
*/
```

Performance left out(er)

Example 5

```
/*
what happens if the actual relation is different?
e.g. the join says "one-to-one" and the data is actually "one-to-many"?
*/

insert into outtab (select * from outtab);
select * from outtab;

-- now outtab has TWO records for every id

select count(i.id)
from intab i
     left outer one to one join outtab o
     on i.id = o.id;

-- COUNT(ID)
-- 10
➔  !!!! WRONG RESULT !!! ---

-- so JOIN CARDINALITY indication is a WEAK bit of information on the data model.
-- It might be WRONG leading to WRONG result sets.
-- BUT: if applied correctly, it can avoid join execution and save whole
-- branches of computation.
```

Performance left out(er)

Example 6

```
-- alternative way to provide the information:
-- PRIMARY KEY and UNIQUE/NOT NULL constraints:
```

```
truncate table outtab;
insert into outtab (select * from intab);

alter table outtab alter (id integer unique);
```

```
select count(i.id)
from intab i
  left outer join outtab o
    on i.id = o.id;
```

```
/*
OPERATOR_NAME      OPERATOR_DETAILS      EXECUTION_ENGINE  TABLE_NAME  TABLE_TYPE  TABLE_SIZE  OUTPUT_SIZE
COLUMN SEARCH      COUNT(*) (LATE MATERIALIZATION, OLTP SEARCH, ENUM_BY: CS_JOIN)  COLUMN          ?           ?           ?           1.0
AGGREGATION        AGGREGATION: COUNT(*)          COLUMN          ?           ?           ?           1.0
JOIN               JOIN CONDITION: (LEFT OUTER) I.ID = O.ID  COLUMN          ?           ?           ?           10.0
      COLUMN TABLE          COLUMN          INTAB        COLUMN TABLE  10.0         10.0
      COLUMN TABLE          COLUMN          OUTTAB       COLUMN TABLE  10.0         10.0
*/
```

-> WHY? Because OUTTAB.ID may still contain NULLs!

```
*/
```

Performance left out(er)

Example 7 – getting rid of unique constraints...

```
alter table outtab alter (id integer not null unique);
/*
Could not execute 'alter table outtab alter (id integer not null unique)'
SAP DBTech JDBC: [261]: invalid index name: column list already indexed
*/
```

➔ great! how to drop the UNIQUE constraint now?

```
alter table outtab alter (id integer );

-- doesn't change a bit
-- we need to manually drop the constraint via DROP CONSTRAINT
select * from constraints where table_name='OUTTAB';
/*
SCHEMA_NAME  TABLE_NAME  COLUMN_NAME  POSITION  CONSTRAINT_NAME  IS_PRIMARY_KEY  IS_UNIQUE_KEY
DEVDUDE      OUTTAB      ID           1        _SYS_TREE_CS_#204986_#14_#0  FALSE          TRUE
*/

alter table outtab drop constraint _SYS_TREE_CS_#204986_#14_#0;
/*
successfully executed in 6 ms 987 µs (server processing time: 5 ms 850 µs) - Rows Affected: 0

-- note how we NOT provide quotation marks here!
➔ hello inconsistent syntax ...
*/
```

Performance left out(er)

Example 8 – table structure based join pruning

```
alter table outtab alter (id integer not null unique);
```

```
select count(i.id)
from intab i
  left outer join outtab o
    on i.id = o.id;
```

```
/*
OPERATOR_NAME      OPERATOR_DETAILS      EXECUTION_ENGINE TABLE_NAME TABLE_TYPE TABLE_SIZE OUTPUT_SIZE
COLUMN SEARCH      COUNT(*) (LATE MATERIALIZATION, OLTP SEARCH, ENUM_BY: CS_TABLE) COLUMN ? ? ? 1.0
AGGREGATION         AGGREGATION: COUNT(*) COLUMN ? ? ? 1.0
COLUMN TABLE      COLUMN TABLE      10.0 10.0
```

```
-> this works nicely and is safe as no wrong result sets are possible
*/
```

Performance left out(er)

Example 9 – all good... and?

➔ ok, so what's the problem here? both statement runs fairly quick.

➔ data volume!

```
create column table materials
as (select row_number() over () as id
    , 'mat-name_'||row_number() over () as mat_name
from objects cross join objects);
```

```
alter table materials add primary key (id);
```

```
select top 5 *, count(*) over() all_rows from materials ;
/*
```

ID	MAT_NAME	ALL_ROWS
1	mat-name_1	22.657.600

2	mat-name_2	22657600
3	mat-name_3	22657600
4	mat-name_4	22657600
5	mat-name_5	22657600

```
*/
create column table mat_infos (id integer, mat_info
nvarchar(300));
```

```
insert into mat_infos (select id, 'mat_info_'||id from
materials);
```

```
select top 5 *, count(*) over() all_rows from mat_infos ;
/*
```

ID	MAT_INFO	ALL_ROWS
1	mat_info_1	22657600
2	mat_info_2	22657600
3	mat_info_3	22657600
4	mat_info_4	22657600
5	mat_info_5	22657600

```
*/
```

```
select m.id, m.mat_name, mi.mat_info
from materials m
left outer join mat_infos mi
on m.id = mi.id;
```

```
create view full_mat_info as
select m.id, m.mat_name, mi.mat_info
from materials m
left outer join mat_infos mi
on m.id = mi.id;
```

Performance left out(er)

Example 10 – What's the damage

```
select id
from full_mat_info;
```

```
/*
OPERATOR_NAME      OPERATOR_DETAILS
COLUMN SEARCH      FULL_MAT_INFO.ID (LATE MATERIALIZATION, OLTP SEARCH, ENUM_BY: CS_JOIN)
  JOIN              JOIN CONDITION: (LEFT OUTER) M.ID = MI.ID
    COLUMN TABLE
    COLUMN TABLE
*/
```

EXECUTION_ENGINE	TABLE_NAME	TABLE_TYPE	TABLE_SIZE	OUTPUT_SIZE
COLUMN	?	?	?	2.26576E+7
COLUMN	?	?	?	2.26576E+7
COLUMN	MATERIALS	COLUMN TABLE	2.26576E+7	2.26576E+7
COLUMN	MAT_INFOS	COLUMN TABLE	2.26576E+7	2.26576E+7

```
select distinct id
from full_mat_info;
```

```
/*
OPERATOR_NAME      OPERATOR_DETAILS
COLUMN SEARCH      FULL_MAT_INFO.ID (LATE MATERIALIZATION, OLTP SEARCH, ENUM_BY: CS_TABLE)
  COLUMN TABLE
*/
```

EXECUTION_ENGINE	TABLE_NAME	TABLE_TYPE	TABLE_SIZE	OUTPUT_SIZE
COLUMN	?	?	?	2.26576E+7
COLUMN	MATERIALS	COLUMN TABLE	2.26576E+7	2.26576E+7

Performance left out(er)

Example 11 – What's the damage?

Executed [8,571.437 ms]

Overview Executed Plan

Time	
Compilation	N/A
Execution	8.5 s
Dominant Operators	
Name	Execution Time
JECreatenTuple	974.86 ms (11.37%)
JESStep4	355.70 ms (4.15%)
JESStep1	150.93 ms (1.76%)
Distribution	
Number of Nodes	1
Number of Network Transfers	0

Context	
SQL Query	select id from full_mat_info
System	skullbox:30103
System Version	1.00.112.07.1477423516
System Compile Type	rel
Memory Allocated	1.4 GByte(s)
Data Flow	
Number of Tables Used	2
Result Record Count	22657600

No DISTINCT specified

DISTINCT specified

Be aware that DISTINCT only comes 'for free' when used on a column(s) that have a unique/not null or primary key constraint
In all other cases, DISTINCT is rather expensive!

Executed [6,643.987 ms]

Overview Executed Plan

Time	
Compilation	N/A
Execution	6.6 s
Dominant Operators	
Name	Execution Time
Materialize Results	163.17 ms (2.46%)
Search	0.43 ms (0.01%)
Acquire Table Lock	0.01 ms (0.00%)
Distribution	
Number of Nodes	1
Number of Network Transfers	0

Context	
SQL Query	select distinct id from full_mat_info
System	skullbox:30103
System Version	1.00.112.07.1477423516
System Compile Type	rel
Memory Allocated	178.8 MByte(s)
Data Flow	
Number of Tables Used	1
Result Record Count	22657600

in case you wondered how to avoid out of memory issues...

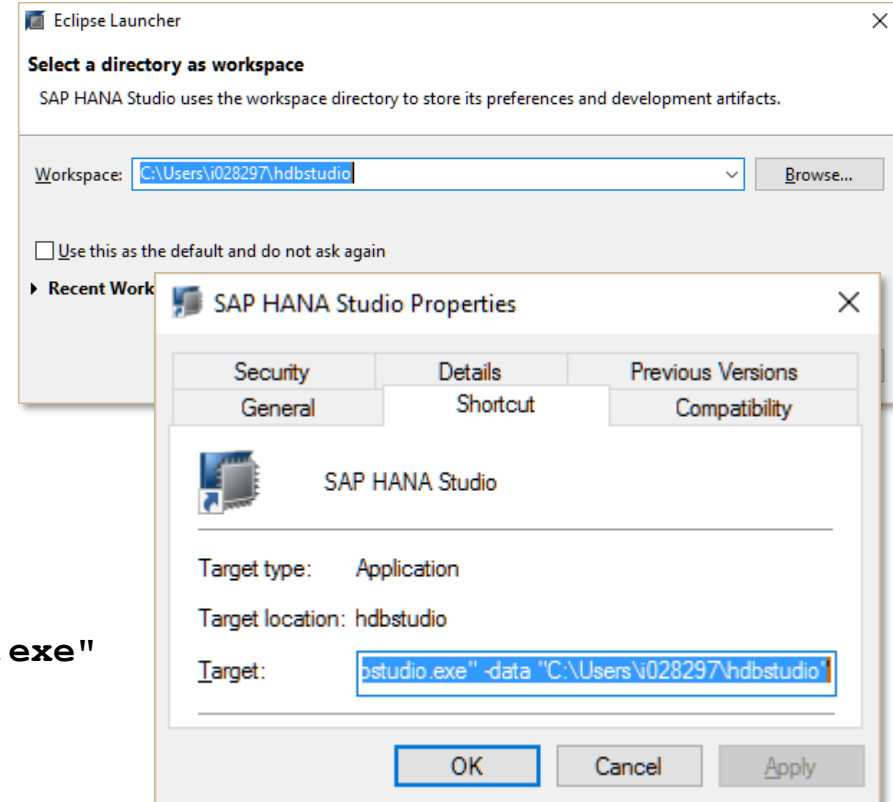
Performance left (outer)

- General SQL know-how and common sense applies to HANA development.
- Pushing computation to the DB layer means understanding what happens there and how to best use it.

One for Everyone

- SAP HANA Studio is still required for certain development tasks and not everyone likes the web-based tools
- When using a WTS server, often a common installation is used – requiring each user to pick the Eclipse Workspace manually. Every. Single. Time. ☹
- Easier: create shortcut to SAP HANA Studio for every user and point to the wanted workspace path:

```
"C:\Program Files\sap\hdbstudio\hdbstudio.exe"  
-data "C:\I028297\hdbstudio"
```



This Presentation Comprises:

- ACT 1: Who am I & what is this about?
 - > Lars Breddemann
 - > SAP HANA development outside the classic use cases
 - > Intel NUC as a SAP HANA development system
- ACT 2: Small systems, small problems, large systems, ...
 - > Finding performance issues
 - > Just not your type
 - > Wrinkles in date and time
 - > That DUMMY has to go...
 - > Performance left out(er)
 - > One for everyone
- ACT 3: Finale
 - > Conclusion

Reference to More Comprehensive Information

- <https://www.sap.com/product/technology-platform/hana.html>
 - SAP HANA product page with links to SCN, documentation, blogs...
- https://help.sap.com/viewer/p/SAP_HANA_PLATFORM
 - New HELP page for SAP HANA Platform
- <https://stackoverflow.com/questions/tagged/hana>
 - stackoverflow topic page SAP HANA
- <https://answers.sap.com/tags/73554900100700000996>
 - SCN Questions & Answers 'SAP HANA' tag
- <https://blogs.sap.com/tags/73554900100700000996/>
 - SCN Blogs 'SAP HANA' tag
- <https://www.youtube.com/user/saphanaacademy>
 - YouTube Channel with free video tutorials
 - GitHub with demo material <https://github.com/saphanaacademy>

Open courses, additional infos

- The Future of Genomics and Precision Medicine
 - > (<https://open.sap.com/courses/asco1-tl>)
- Code of Life - When Computer Science Meets Genetics
 - > (<https://open.hpi.de/courses/ehealth2016/>)
- CancerLinq
 - > www.cancerlinq.org
- <https://news.sap.com/sap-announces-sap-connected-health-platform-and-strategic-relationships-for-transforming-healthcare/>
- <https://news.sap.com/tags/sap-connected-health/>

5 Leading Insights

- SAP HANA is not just for typical “SAP” applications, but a general development platform.
- Pushing computation to the DB layer means understanding what happens there and how to best use it.
- If SQL code looks convoluted and ugly, it’s probably not the best possible code. Look for features/commands, that can help with your task.
- General SQL know-how and common sense applies to HANA development.
- Single DB functions rarely equal application level services.

Questions?

How to contact me:

Lars Breddemann

lars.breddemann@sap.com



Usually I **don't do email Q&A** as this simply doesn't help with knowledge sharing.

Instead, I advise everyone to post the question in one of the **HANA**

related forums (SAP Community

<https://answers.sap.com/questions/metadata/23925/sap-hana.html>, **JAM**

https://jam4.sapjam.com/groups/about_page/6UHzR2Fxxa4quFAbACtxFD or even **stackoverflow**

<http://stackoverflow.com/questions/tagged/hana>

) so that the question and its answers are **search- and findable**.

That way everyone can benefit from this and you even might get faster

and/or better answers than from just writing to me.

I'm happy to answer your question, just send me a link to your question post so that I don't miss it.

Cheers from Melbourne,

Lars