Tableau on SAP HANA:

Performance Tracing and Workload Analysis

Contents

Introduction: Purpose & prerequisites

Performance and stability are critical for the successful implementation of Tableau dashboards on SAP HANA live data sources. To achieve the best possible performance, it is important to analyze the dashboard runtime during the design process and continuously monitor the system for expensive queries thereafter. The purpose of this document is to give an overview of the performance tracing and monitoring capabilities in SAP HANA and Tableau to identify performance bottlenecks. This understanding is a prerequisite for identifying suitable performance optimization measures. Recommendations on how to design efficient Tableau dashboards are not part of the scope—these can be found in separate resources, e.g. Designing Efficient Workbooks (whitepaper), and Best Practices for Dashboard Performance (recorded presentation).

In order to activate the SAP HANA traces mentioned in this document, sufficient authorizations or the help of an **SAP HANA admin** are required. Additionally, a **Tableau Creator** (e.g. analyst or dashboard designer) will be required for collecting the front-end runtimes measured by Tableau. When combined, these Tableau and SAP HANA traces provide the full end-to-end runtime distribution. These traces provide information needed to identify performance bottlenecks which are great candidates for performance optimizations.

While it is recommended that detailed end-to-end traces only be performed for single executions (due to its resource consumption overhead), there are other traces for continuous performance and workload monitoring that can identify expensive queries. To avoid bottlenecks during high-load situations, it is important to be able to analyze and identify the source of expensive (CPU runtime and memory consumption) and dominant (runtime multiplied by execution count) queries. SAP HANA can be configured to collect these statistics and Tableau can connect to them for analysis.

When a long-running or memory intensive SQL statement has been singled out, its processing inside HANA can be further analyzed by using tools such as the SAP HANA Plan Explanation or SAP HANA Plan Visualizer. These tools provide detailed step-by-step information on how the query result is being calculated and which processing steps are costly. This understanding can help identify optimization strategies such as applying filters earlier, aggregating at a different level, etc.

Overview of SAP HANA runtime traces & statistics

Expensive Statements Trace

The Expensive Statements Trace is capturing information about SQL statements whose execution time exceeded a configured threshold. The threshold is entered in microseconds (µs), which means in millionths of a second.

The trace can be used in different ways. It can record practically all queries when the threshold is set to 1, but this setting should only be used for a limited duration where a dedicated performance analysis is required. However, when setting the threshold to a higher value, e.g. 5 seconds (5,000,000 µs), it can remain activated permanently. SAP Note 2180165 (FAQ: SAP HANA Expensive Statements Trace) states, “Due to the significant added value and the small overhead (in case of reasonable thresholds) it is recommended to activate this trace on a permanent basis”.

Another advantage of this trace is that it allows flexible analysis of its records as its results can be retrieved by selecting from a SAP HANA view (‘M_EXPENSIVE_STATEMENTS’ in schema ‘SYS’). This means Tableau’s powerful analytical capabilities can be used to analyze the Expensive Statements recorded in SAP HANA.

Due to the mentioned advantages (can remain permanently active and can be queried using SQL), the Expensive Statements Trace was chosen to be described in more detail in the next chapter.

SQL Trace

While the Expensive Statements Trace can remain permanently activated for performance monitoring purposes, SAP HANA also offers a trace for the purpose of dedicated performance analysis—capturing all SQL queries and their runtime statistics while activated (typically for a dedicated user and short amount of time). Because of this high level of detail, the SQL Trace consumes more resources (storage and CPU) and should only be activated for short-term, explicit performance analysis (instead of long-term monitoring).

For an example of how the trace result looks like and how the SQL Trace Analyzer tool works, see this video from SAP HANA Academy demonstrating how to use the SAP HANA SQL Trace Analyzer.

The SQL Trace is writing the information it collects to a .py text file instead of a database table. While this file type has the advantage of being able to replay the traced database operations, it is difficult to read and analyze traced performance information. To address this issue, SAP offers a python tool, SAP HANA SQL Trace Analyzer, to aggregate the information from the file and to simplify its analysis.

SQL Plan Cache

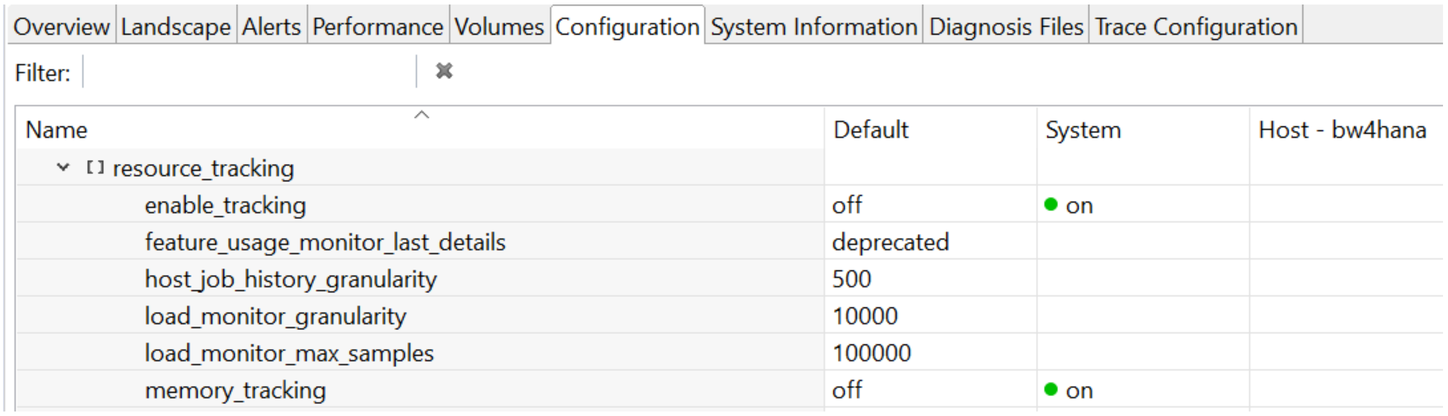
The SQL Plan Cache is a valuable tool for understanding the SQL processing of the SAP HANA database. As it is not a trace, but a cache that is activated per default, the SQL Plan Cache collects valuable statistics without the need to explicitly switch it on.

The SQL Plan Cache provides an overview of the statements that are frequently executed in the system and tracks runtime statistics. These statistics are aggregated so they don't reveal runtime information of specific dashboard executions, but they can be used to identify candidates for optimization—e.g. most frequently executed queries, longest running queries, etc. The SQL Plan cache can be queried as an SAP HANA View so Tableau can be used for its analysis.

Enabling SAP HANA Memory Tracking

Some SAP HANA traces allow capturing the memory consumption of SQL statements in addition to performance KPIs. This can be valuable information when analyzing out of memory situations or trying to identify users that heavily consume the database's memory. To enable the tracking of memory usage, the following two parameters in the global.ini file [resource_tracking] section must be set to 'on':

- enable_tracking
- memory_tracking



Name	Default	System	Host - bw4hana
[] resource_tracking			
enable_tracking	off	● on	
feature_usage_monitor_last_details	deprecated		
host_job_history_granularity	500		
load_monitor_granularity	10000		
load_monitor_max_samples	100000		
memory_tracking	off	● on	

End-to-end single execution trace:

Combining Tableau & SAP HANA performance statistics

End-to-end analysis involves temporarily activating traces on both the Tableau and SAP HANA side and should only be executed for single executions instead of a continuous monitoring.

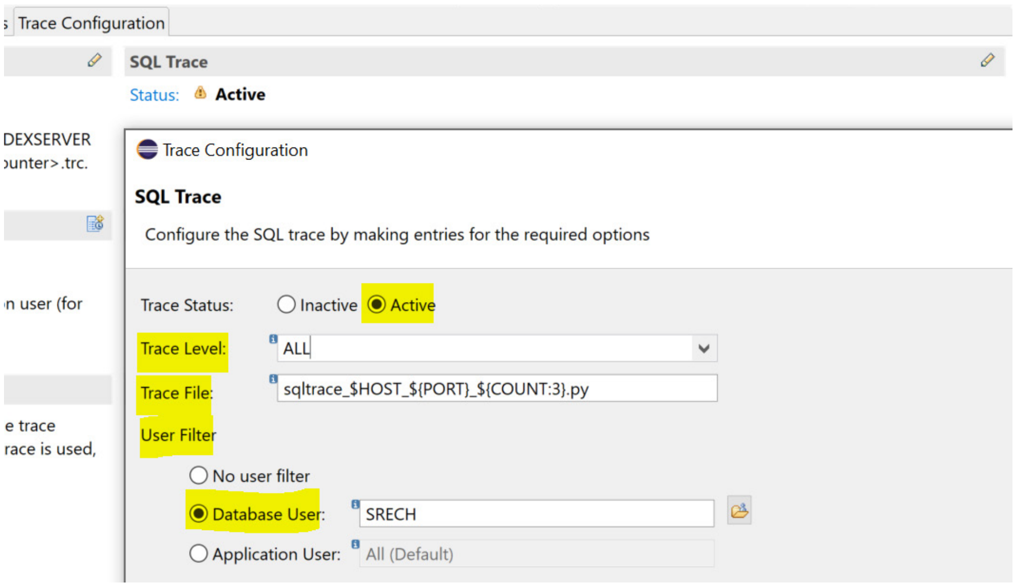
To understand a Tableau workbook's runtime distribution, it is recommended to trace the performance on both the Tableau and the database side—end to end. This will allow for conclusions to be drawn on which layer most of the runtime is spent (Tableau, Network, HANA) and which steps are slowing down the overall execution. This will help identify those steps with the most potential for performance optimization.

Activating SAP HANA's SQL Trace

As a first step, SAP HANA's SQL Trace needs to be activated for the database user that is executing the trace. This ensures that all incoming SQL queries from that user, including their runtime information, are captured. One method to activate this trace is to set it in SAP HANA Studio in the 'Administration' > 'Trace Configuration' section.

To enable the trace, set the Trace Status to 'Active' and use the Trace Level 'ALL' or 'ALL_WITH_RESULTS'. The Trace File field allows you to specify the name of the trace file that gets generated. Additionally, the Database User should be filtered to the user who is executing the Tableau workbook.

For a detailed description of the SQL Trace configuration, see SAP Note 2031647 (How to enable SQL Trace in SAP HANA Studio).



Trace Configuration

SQL Trace

Configure the SQL trace by making entries for the required options

Trace Status: Inactive [✓ Active]

Trace Level: ALL

Trace File: sqltrace_$HOST_${PORT}_${COUNT:3}.py

User Filter

No user filter

[✓ Database User:] SRECH

Application User: All (Default)

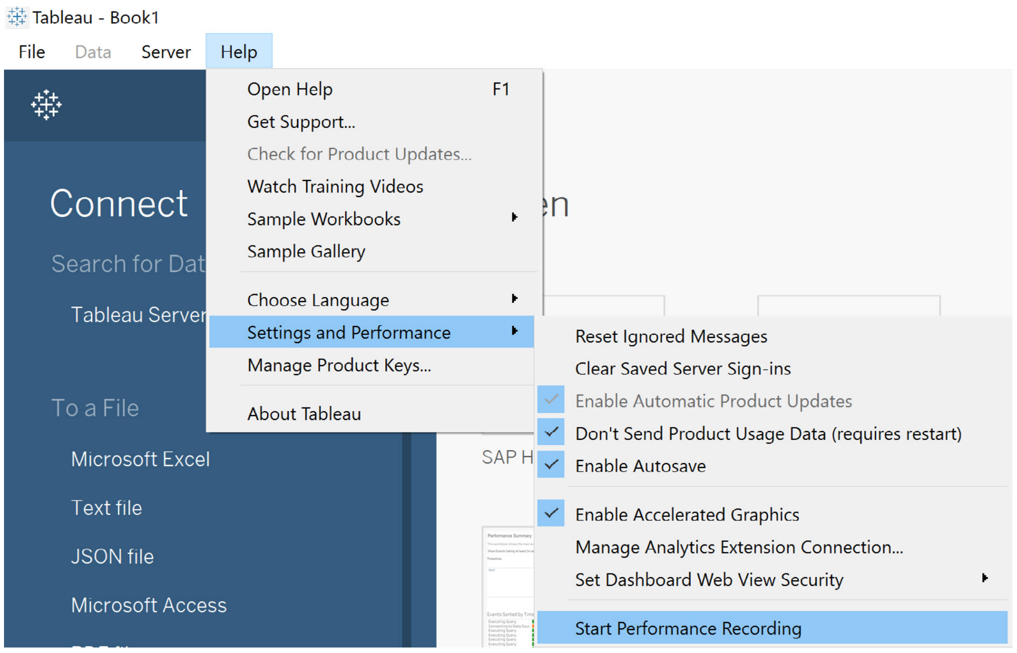
Activating Tableau's Performance Recorder

To record the runtime distribution on the Tableau side, a Tableau performance trace can be used.

Tableau Desktop

When using Tableau Desktop, the performance trace is started by clicking on:

Help > Settings and Performance > Start Performance Recording



Every step performed in Tableau will be recorded as part of the performance trace now.

To stop recording and view a temporary workbook containing results from the recording session, click the same button again (now it's named 'Stop Performance Recording'):

Help > Settings and Performance > Stop Performance Recording

As a result, a Tableau workbook will be generated that contains the traced information. Using 'File' > 'Save as' this workbook can be stored for future reference.

For a detailed description of the Tableau Desktop Trace activation, see Record and Analyze Workbook Performance in Tableau Help.

Tableau Server

When using Tableau Server, the performance trace is started by adding ':record_performance=yes&' at the end of the view URL.

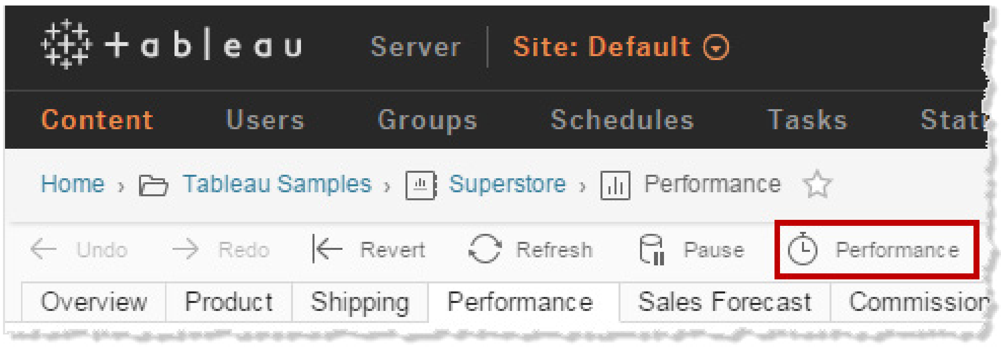
Example: View URL

```
http://10.32.139.22/#/views/Coffee_Sales2013/
USSalesMarginsByAreaCode?:iid=1
```

Example: View URL with enabled performance tracing:

```
http://10.32.139.22/#/views/Coffee_
Sales2013/USSalesMarginsByAreaCode?:record_
performance=yes&:iid=1
```

To view the performance recording from Tableau Server, click the 'Performance' button.



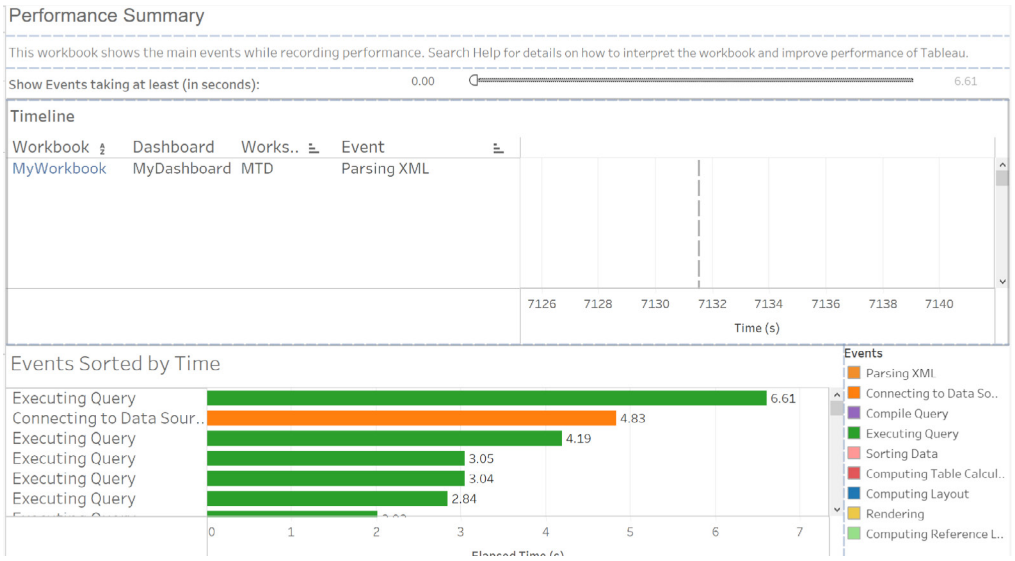
To stop the performance trace, you can click to a different page or remove ':record_performance=yes' from the URL.

For a detailed description of the Tableau Server Trace activation, see Create a Performance Recording in Tableau Help.

Analyzing the information from the Tableau Performance Workbook & HANA SQL Trace

The result of the Tableau performance trace is presented in the form of a Tableau workbook.

The ‘Performance Summary’ dashboard gives an overview of the timeline of the performed steps and the chart at the bottom sorts the events by runtime.



Performance Summary

This workbook shows the main events while recording performance. Search Help for details on how to interpret the workbook and improve performance of Tableau.

Show Events taking at least (in seconds): 0.00 … 6.61

Timeline

Workbook	Dashboard	Works..	Event
MyWorkbook	MyDashboard	MTD	Parsing XML

Events Sorted by Time

Executing Query	6.61
Connecting to Data Sour..	4.83
Executing Query	4.19
Executing Query	3.05
Executing Query	3.04
Executing Query	2.84

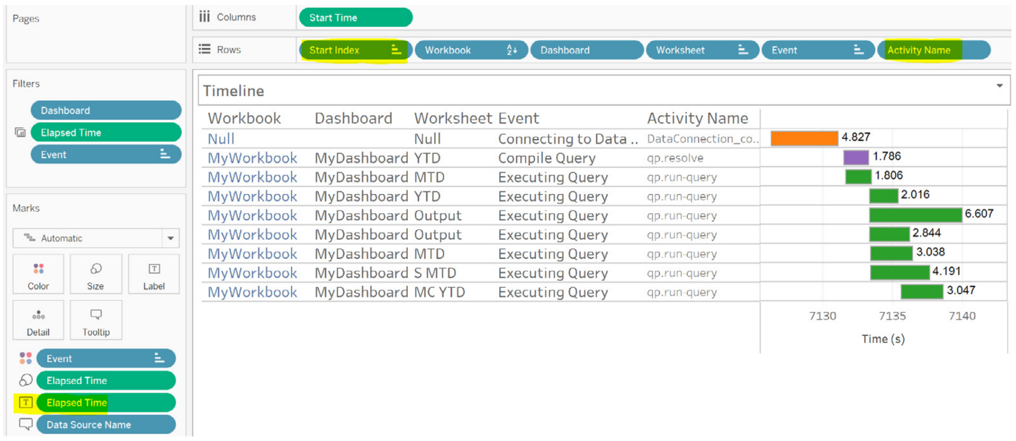
Events
- 🟧 Parsing XML
- 🟧 Connecting to Data So..
- 🟪 Compile Query
- 🟩 Executing Query
- 🟥 Sorting Data
- 🟥 Computing Table Calcul..
- 🟦 Computing Layout
- 🟨 Rendering
- 🟩 Computing Reference L..

In order to focus on the most important events, it is recommended to remove the noise by filtering for events taking at least 0.5 seconds.

Show Events taking at least (in seconds): 0.50

The Timeline sheet can be adjusted to reveal additional information. For example, the ‘Elapsed Time’ can be added as Label, the ‘Start Index’ can be dragged to be the first field in rows (which sorts the actions by start time) and ‘Activity Name’ can be added to the rows.

The result will look similar to this:



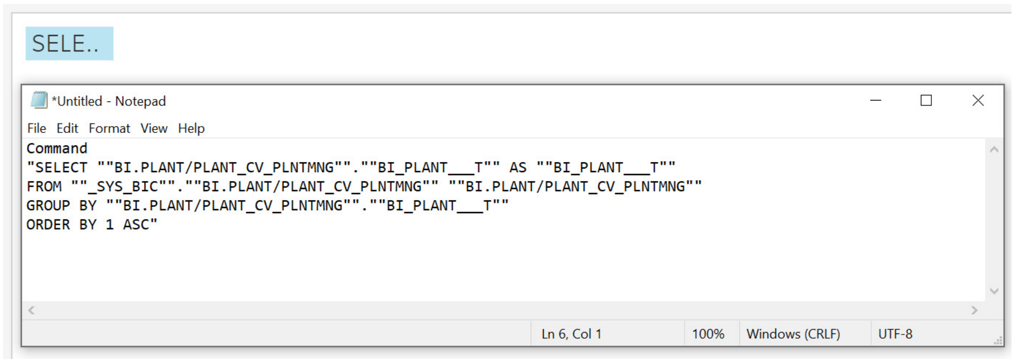
Timeline

Workbook	Dashboard	Worksheet	Event	Activity Name	
Null		Null	Connecting to Data ..	DataConnection_co..	4.827
MyWorkbook	MyDashboard YTD		Compile Query	qp.resolve	1.786
MyWorkbook	MyDashboard MTD		Executing Query	qp.run-query	1.806
MyWorkbook	MyDashboard YTD		Executing Query	qp.run-query	2.016
MyWorkbook	MyDashboard Output		Executing Query	qp.run-query	6.607
MyWorkbook	MyDashboard Output		Executing Query	qp.run-query	2.844
MyWorkbook	MyDashboard MTD		Executing Query	qp.run-query	3.038
MyWorkbook	MyDashboard S MTD		Executing Query	qp.run-query	4.191
MyWorkbook	MyDashboard MC YTD		Executing Query	qp.run-query	3.047

In this case, the majority of the runtime is spent executing queries. 'Executing Query' is the time that Tableau is waiting for a query response from SAP HANA, including the time for transporting the response through the network.

In order to identify the complete SQL statement that corresponds to an 'Executing Query' event, the event can be selected in the Performance Summary. This will filter the Query sheet to display the SQL command from that event.

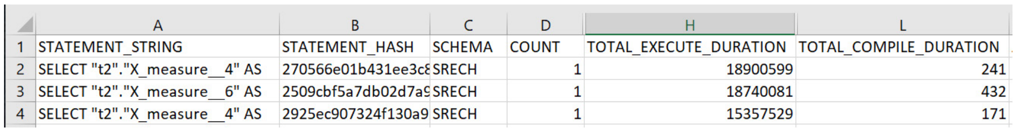
The Tableau workbook sheet will have too little space to display the statement, but it can be selected and copied into a text editor for full display.



```
Command
"SELECT ""BI.PLANT/PLANT_CV_PLNTMNG"".""BI_PLANT___T"" AS ""BI_PLANT___T""
FROM ""_SYS_BIC"".""BI.PLANT/PLANT_CV_PLNTMNG"" ""BI.PLANT/PLANT_CV_PLNTMNG""
GROUP BY ""BI.PLANT/PLANT_CV_PLNTMNG"".""BI_PLANT___T""
ORDER BY 1 ASC"
```

After having identified which work sheets and queries consume most of the runtime, this information can be combined with the HANA SQL Trace.

After executing SAP's SQL Trace Analyzer (see SAP Note 2412519 FAQ: SAP HANA SQL Trace Analyzer) on the trace file, the output will reveal runtime information for each SQL query in HANA—including the execution, compilation, cursor and fetch runtimes. This information is displayed in microseconds so a division by 1,000,000 will convert it to seconds.



	A	B	C	D	H	L
1	STATEMENT_STRING	STATEMENT_HASH	SCHEMA	COUNT	TOTAL_EXECUTE_DURATION	TOTAL_COMPILE_DURATION
2	SELECT "t2"."X_measure__4" AS	270566e01b431ee3c	SRECH	1	18900599	241
3	SELECT "t2"."X_measure__6" AS	2509cbf5a7db02d7a	SRECH	1	18740081	432
4	SELECT "t2"."X_measure__4" AS	2925ec907324f130a9	SRECH	1	15357529	171

A comparison between the SQL statement runtimes in HANA and the corresponding 'Executing Query' runtimes in Tableau can provide valuable insight. If there is a large gap, it might be related to the network speed or a firewall between the HANA server and the Tableau front-end.

If the runtime of a SQL query in HANA is large, there are multiple ways to address this. For example:

- Check if the query can be simplified by redesigning the Tableau worksheet.
- Check if the HANA data model can be optimized for the query (see the HANA Plan Explanation and HANA Plan Visualization section of this document).

If the SQL execution plan compilation time in SAP HANA is large or even larger than the query execution, it is a hint that Tableau's bind variable feature should be activated. This capability can increase the hit rate of the SQL plan cache and consequently reduces the need for compiling SQL execution plans.

If there is a long running SQL query that ends with 'HAVING (COUNT(1) > 0)', there is a TDC setting ('CAP_QUERY_HAVING_REQUIRES_GROUP_BY') which replaces this condition with a 'GROUP BY' statement when enabled. This setting has dramatically reduced the SQL query runtime in certain use cases.

These are some examples of findings that can be made by analyzing the Tableau traces combined with SAP HANA traces.

Monitoring:

Identify long-running & memory-intensive SQL queries

SAP HANA provides various traces, logs, and statistics that contain valuable information for performance analysis and monitoring. For example, when high load resource bottlenecks occur, they can be used to identify performance optimization candidates and/or perform root cause analysis.

Unfortunately, it can be difficult to identify which information from these logs are related to Tableau queries, or even a particular Tableau workbook or sheet. To track this information for Tableau queries, Tableau's Initial SQL feature can be very helpful. It extends the possibilities to analyze Tableau workload and performance in SAP HANA by making use of SAP HANA session variables. Together with information from SAP HANA logs, e.g. the Expensive Statements Trace or the SQL Plan Cache, this creates new possibilities for SAP HANA admins and Tableau workbook designers.

Enhancing SAP HANA's session information through Tableau's Initial SQL

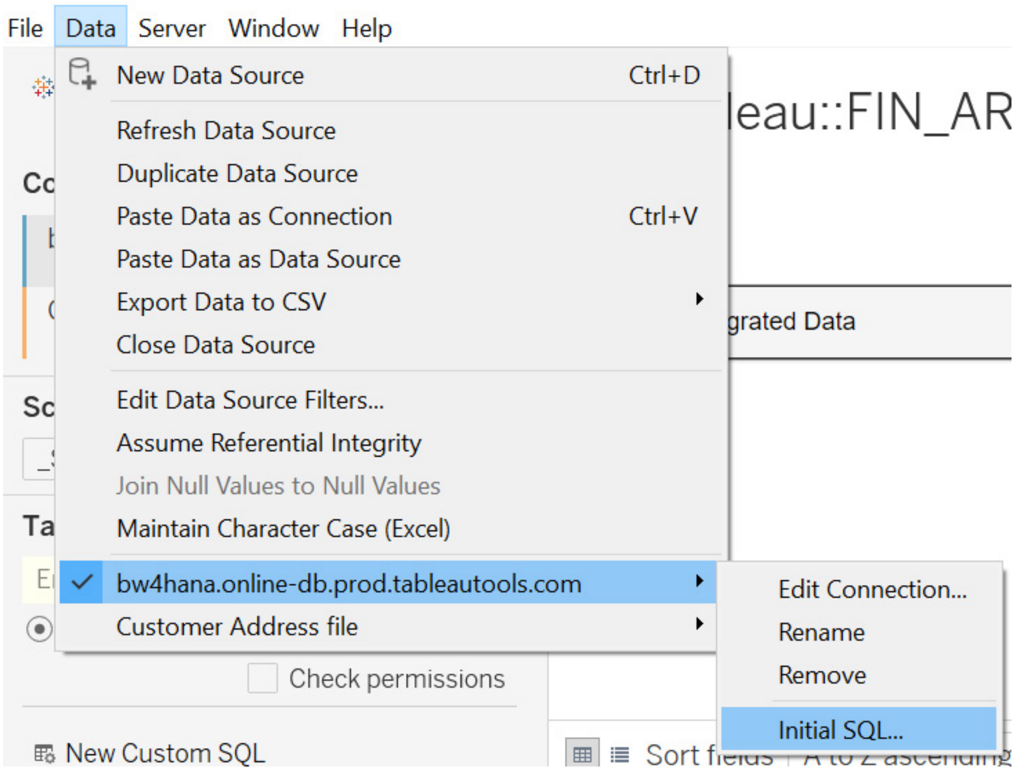
When a new connection to SAP HANA is made, a new session is established in the SAP HANA Session Management. It maintains information regarding the source of the session and technical information which are used by SAP HANA for traces and statistical data.

Examples for pre-defined session variables are APPLICATION, APPLICATIONVERSION, APPLICATIONUSER, and APPLICATIONSOURCE. A complete list of the predefined session variables and their uses can be found in the SAP HANA SQL and System Views Reference.

When Tableau connects to SAP HANA, some session variables (e.g. APPLICATION and APPLICATIONVERSION) are automatically assigned by Tableau. This capability helps separate Tableau queries and/or workload from other applications/tools. Tableau's Initial SQL feature can be used to assign additional session variables to store information; such as the Tableau Workbook Name or Tableau Server User. Initial SQL is a set of SQL commands defined on the data source level that will be executed when a database connection is made, e.g. when a workbook is opened or data is being refreshed.

In this example, we will use APPLICATIONSOURCE to submit the name of the Tableau workbook that established the connection and APPLICATIONUSER to identify the Tableau User who is running the query.

In order to set the Initial SQL, the Data Source tab needs to be opened and in the Data menu, the SAP HANA data source needs to be selected:



The Syntax for setting or overwriting session variables is as follows:

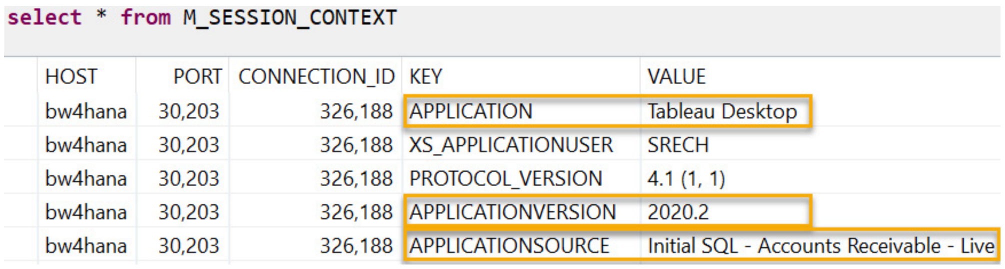
SET [SESSION] <variable_string_literal> = <value_string_literal>

As an example, the following commands can be used in the Initial SQL to store the Tableau application, version, and workbook name in the SAP HANA session variables:

SET SESSION ‘APPLICATIONSOURCE’ = [WorkbookName];

SET SESSION ‘APPLICATIONUSER’ = [TableauServerUser];

To check whether the Initial SQL providing the information you expect, the session variables can be retrieved by querying the M_SESSION_CONTEXT system view (e.g. 'select * from M_SESSION_CONTEXT').

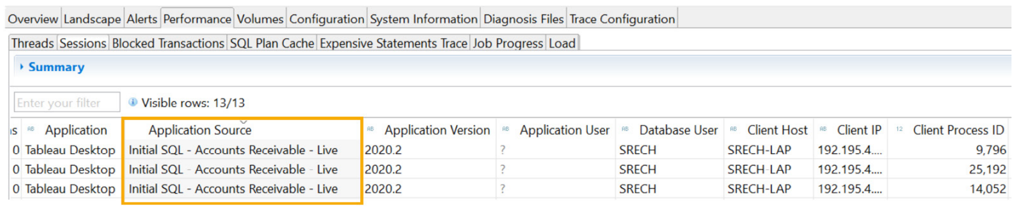


```
select * from M_SESSION_CONTEXT
```

HOST	PORT	CONNECTION_ID	KEY	VALUE
bw4hana	30,203	326,188	APPLICATION	Tableau Desktop
bw4hana	30,203	326,188	XS_APPLICATIONUSER	SRECH
bw4hana	30,203	326,188	PROTOCOL_VERSION	4.1 (1, 1)
bw4hana	30,203	326,188	APPLICATIONVERSION	2020.2
bw4hana	30,203	326,188	APPLICATIONSOURCE	Initial SQL - Accounts Receivable - Live

As a result, SAP HANA admin tools like the Session Monitoring or the Expensive Statements Trace will display information about the source of the session that previously was not available.

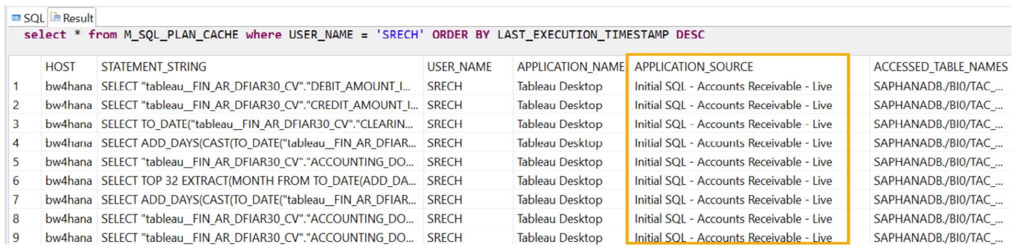
a) Session Monitoring:



Visible rows: 13/13

s	Application	Application Source	Application Version	Application User	Database User	Client Host	Client IP	Client Process ID
0	Tableau Desktop	Initial SQL - Accounts Receivable - Live	2020.2	?	SRECH	SRECH-LAP	192.195.4....	9,796
0	Tableau Desktop	Initial SQL - Accounts Receivable - Live	2020.2	?	SRECH	SRECH-LAP	192.195.4....	25,192
0	Tableau Desktop	Initial SQL - Accounts Receivable - Live	2020.2	?	SRECH	SRECH-LAP	192.195.4....	14,052

b) Logs and traces:


SQL Result

```
select * from M_SQL_PLAN_CACHE where USER_NAME = 'SRECH' ORDER BY LAST_EXECUTION_TIMESTAMP DESC
```

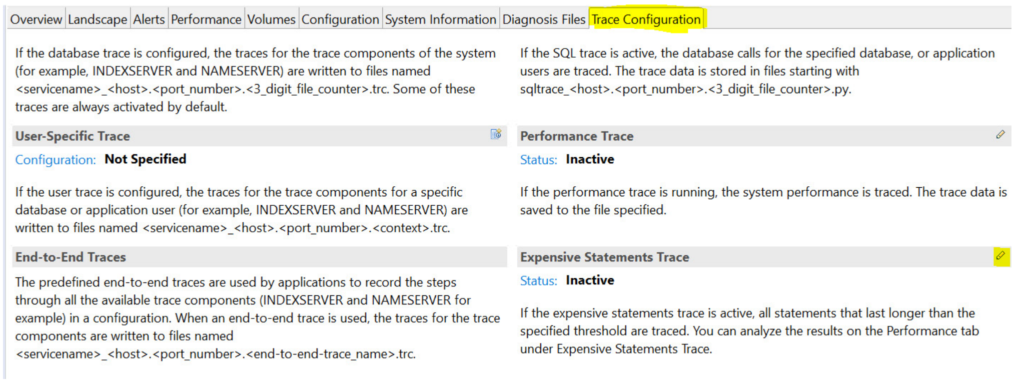
	HOST	STATEMENT_STRING	USER_NAME	APPLICATION_NAME	APPLICATION_SOURCE	ACCESSED_TABLE_NAMES
1	bw4hana	SELECT "tableau__FIN_AR_DFIAR30_CV"."DEBIT_AMOUNT_I...	SRECH	Tableau Desktop	Initial SQL - Accounts Receivable - Live	SAPHANADB./BI0/TAC_...
2	bw4hana	SELECT "tableau__FIN_AR_DFIAR30_CV"."CREDIT_AMOUNT_I...	SRECH	Tableau Desktop	Initial SQL - Accounts Receivable - Live	SAPHANADB./BI0/TAC_...
3	bw4hana	SELECT TO_DATE("tableau__FIN_AR_DFIAR30_CV"."CLEARIN...	SRECH	Tableau Desktop	Initial SQL - Accounts Receivable - Live	SAPHANADB./BI0/TAC_...
4	bw4hana	SELECT ADD_DAYS(CAST(TO_DATE("tableau__FIN_AR_DFIAR...	SRECH	Tableau Desktop	Initial SQL - Accounts Receivable - Live	SAPHANADB./BI0/TAC_...
5	bw4hana	SELECT "tableau__FIN_AR_DFIAR30_CV"."ACCOUNTING_DO...	SRECH	Tableau Desktop	Initial SQL - Accounts Receivable - Live	SAPHANADB./BI0/TAC_...
6	bw4hana	SELECT TOP 32 EXTRACT(MONTH FROM TO_DATE(ADD_DA...	SRECH	Tableau Desktop	Initial SQL - Accounts Receivable - Live	SAPHANADB./BI0/TAC_...
7	bw4hana	SELECT ADD_DAYS(CAST(TO_DATE("tableau__FIN_AR_DFIAR...	SRECH	Tableau Desktop	Initial SQL - Accounts Receivable - Live	SAPHANADB./BI0/TAC_...
8	bw4hana	SELECT "tableau__FIN_AR_DFIAR30_CV"."ACCOUNTING_DO...	SRECH	Tableau Desktop	Initial SQL - Accounts Receivable - Live	SAPHANADB./BI0/TAC_...
9	bw4hana	SELECT "tableau__FIN_AR_DFIAR30_CV"."ACCOUNTING_DO...	SRECH	Tableau Desktop	Initial SQL - Accounts Receivable - Live	SAPHANADB./BI0/TAC_...

Expensive Statements Trace: Monitoring long-running & memory-intensive queries

SAP HANA offers the possibility to trace SQL statements whose execution time exceeds a configured threshold. This widely used feature is called Expensive Statement Trace and is typically used to identify queries that need performance optimization.

If the SAP HANA session variables are not assigned in Tableau's Initial SQL, it can be difficult to find out which of the long-running SQL statements come from Tableau (depends on the Tableau version used), and more importantly, which Tableau workbook caused the query.

After assigning the session variables in Tableau's Initial SQL, this gets much easier. When we configure the trace, it is possible to restrict it not only on the user level but also on the application level. It is important to pay attention to the Threshold Duration (in microseconds: 1 µs = one millionth of a second). If only long running queries should be traced, this value should be high (default is 1,000,000). But if all or most Tableau queries should be traced, it can be set to a much lower value (e.g. 1,000 for 1 ms).



If the database trace is configured, the traces for the trace components of the system (for example, INDEXSERVER and NAMESERVER) are written to files named <servicename>_<host>.<port_number>.<3_digit_file_counter>.trc. Some of these traces are always activated by default.

If the SQL trace is active, the database calls for the specified database, or application users are traced. The trace data is stored in files starting with sqltrace_<host>.<port_number>.<3_digit_file_counter>.py.

User-Specific Trace

Configuration: **Not Specified**

If the user trace is configured, the traces for the trace components for a specific database or application user (for example, INDEXSERVER and NAMESERVER) are written to files named <servicename>_<host>.<port_number>.<context>.trc.

Performance Trace

Status: **Inactive**

If the performance trace is running, the system performance is traced. The trace data is saved to the file specified.

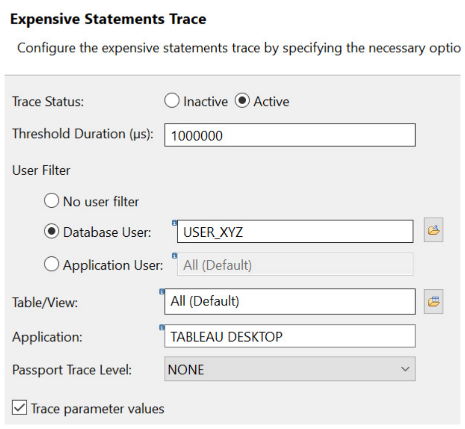
End-to-End Traces

The predefined end-to-end traces are used by applications to record the steps through all the available trace components (INDEXSERVER and NAMESERVER for example) in a configuration. When an end-to-end trace is used, the traces for the trace components are written to files named <servicename>_<host>.<port_number>.<end-to-end-trace_name>.trc.

Expensive Statements Trace

Status: **Inactive**

If the expensive statements trace is active, all statements that last longer than the specified threshold are traced. You can analyze the results on the Performance tab under Expensive Statements Trace.


Expensive Statements Trace

Configure the expensive statements trace by specifying the necessary optio

Trace Status: ( ) Inactive [✓ Active]

Threshold Duration (µs): 1000000

User Filter

( ) No user filter

[✓ Database User:] USER_XYZ

( ) Application User: All (Default)

Table/View: All (Default)

Application: TABLEAU DESKTOP

Passport Trace Level: NONE

☑ Trace parameter values

After the trace has finished, a select on the Expensive Statements view, including a filter on the APPLICATION_NAME column, will return those statements that originated in Tableau.

select * from M_EXPENSIVE_STATEMENTS where APPLICATION_NAME like '%Tableau%'

The result set contains valuable information for each database operation—among others: User name, Workbook name, SQL statement, Start time, Duration (in microseconds), affected DB tables, and number of records. This query can be further restricted to certain users, workbooks, start time, etc. in the 'where' clause.

The following information may be particularly useful:

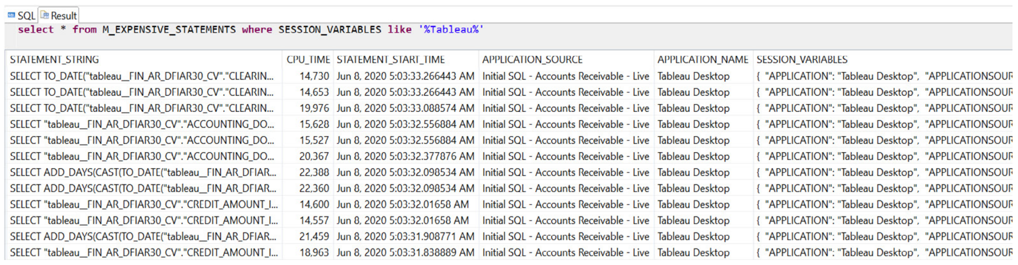
- The type of operation during the statement execution (OPERATION)
- When the query started (START_TIME)
- How long the query took (DURATION_MICROSEC)
- The CPU time (in microseconds) to compute the statement (CPU_TIME)
- Name(s) of the objects accessed (OBJECT_NAME)
- The SQL statement (STATEMENT_STRING)
- Peak memory usage (in bytes) during the execution of the statement (MEMORY_SIZE)

A few important operation types that are recorded include:

Operation	Description
AGGREGATED_EXECUTION	Overall execution time of an individual database request
CALL	Execution time of procedure calls
COMPILE	Preparation/parse time
CURSOR_CLOSE	Cursor closing time
FETCH	Fetch time

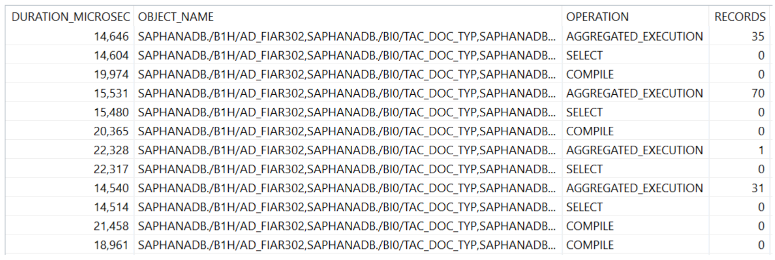
Operation	Description
SELECT, INSERT, UPDATE, DELETE	Execution time of corresponding operation

An example of an Expensive Statements Trace result:



```
select * from M_EXPENSIVE_STATEMENTS where SESSION_VARIABLES like '%Tableau%'
```

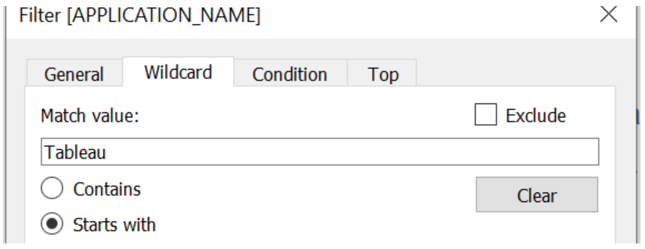
STATEMENT_STRING	CPU_TIME	STATEMENT_START_TIME	APPLICATION_SOURCE	APPLICATION_NAME	SESSION_VARIABLES
SELECT TO_DATE("tableau__FIN_AR_DFIAR30_CV"."CLEARIN...	14,730	Jun 8, 2020 5:03:33.266443 AM	Initial SQL - Accounts Receivable - Live	Tableau Desktop	{ "APPLICATION": "Tableau Desktop", "APPLICATIONSOUR
SELECT TO_DATE("tableau__FIN_AR_DFIAR30_CV"."CLEARIN...	14,653	Jun 8, 2020 5:03:33.266443 AM	Initial SQL - Accounts Receivable - Live	Tableau Desktop	{ "APPLICATION": "Tableau Desktop", "APPLICATIONSOUR
SELECT TO_DATE("tableau__FIN_AR_DFIAR30_CV"."CLEARIN...	19,976	Jun 8, 2020 5:03:33.088574 AM	Initial SQL - Accounts Receivable - Live	Tableau Desktop	{ "APPLICATION": "Tableau Desktop", "APPLICATIONSOUR
SELECT "tableau__FIN_AR_DFIAR30_CV"."ACCOUNTING_DO...	15,628	Jun 8, 2020 5:03:32.556884 AM	Initial SQL - Accounts Receivable - Live	Tableau Desktop	{ "APPLICATION": "Tableau Desktop", "APPLICATIONSOUR
SELECT "tableau__FIN_AR_DFIAR30_CV"."ACCOUNTING_DO...	15,527	Jun 8, 2020 5:03:32.556884 AM	Initial SQL - Accounts Receivable - Live	Tableau Desktop	{ "APPLICATION": "Tableau Desktop", "APPLICATIONSOUR
SELECT "tableau__FIN_AR_DFIAR30_CV"."ACCOUNTING_DO...	20,367	Jun 8, 2020 5:03:32.377876 AM	Initial SQL - Accounts Receivable - Live	Tableau Desktop	{ "APPLICATION": "Tableau Desktop", "APPLICATIONSOUR
SELECT ADD_DAYS(CAST(TO_DATE("tableau__FIN_AR_DFIAR...	22,388	Jun 8, 2020 5:03:32.098534 AM	Initial SQL - Accounts Receivable - Live	Tableau Desktop	{ "APPLICATION": "Tableau Desktop", "APPLICATIONSOUR
SELECT ADD_DAYS(CAST(TO_DATE("tableau__FIN_AR_DFIAR...	22,360	Jun 8, 2020 5:03:32.098534 AM	Initial SQL - Accounts Receivable - Live	Tableau Desktop	{ "APPLICATION": "Tableau Desktop", "APPLICATIONSOUR
SELECT "tableau__FIN_AR_DFIAR30_CV"."CREDIT_AMOUNT_I...	14,600	Jun 8, 2020 5:03:32.01658 AM	Initial SQL - Accounts Receivable - Live	Tableau Desktop	{ "APPLICATION": "Tableau Desktop", "APPLICATIONSOUR
SELECT "tableau__FIN_AR_DFIAR30_CV"."CREDIT_AMOUNT_I...	14,557	Jun 8, 2020 5:03:32.01658 AM	Initial SQL - Accounts Receivable - Live	Tableau Desktop	{ "APPLICATION": "Tableau Desktop", "APPLICATIONSOUR
SELECT ADD_DAYS(CAST(TO_DATE("tableau__FIN_AR_DFIAR...	21,459	Jun 8, 2020 5:03:31.908771 AM	Initial SQL - Accounts Receivable - Live	Tableau Desktop	{ "APPLICATION": "Tableau Desktop", "APPLICATIONSOUR
SELECT "tableau__FIN_AR_DFIAR30_CV"."CREDIT_AMOUNT_I...	18,963	Jun 8, 2020 5:03:31.838889 AM	Initial SQL - Accounts Receivable - Live	Tableau Desktop	{ "APPLICATION": "Tableau Desktop", "APPLICATIONSOUR



DURATION_MICROSEC	OBJECT_NAME	OPERATION	RECORDS
14,646	SAPHANADB./B1H/AD_FIAR302,SAPHANADB./BI0/TAC_DOC_TYP,SAPHANADB...	AGGREGATED_EXECUTION	35
14,604	SAPHANADB./B1H/AD_FIAR302,SAPHANADB./BI0/TAC_DOC_TYP,SAPHANADB...	SELECT	0
19,974	SAPHANADB./B1H/AD_FIAR302,SAPHANADB./BI0/TAC_DOC_TYP,SAPHANADB...	COMPILE	0
15,531	SAPHANADB./B1H/AD_FIAR302,SAPHANADB./BI0/TAC_DOC_TYP,SAPHANADB...	AGGREGATED_EXECUTION	70
15,480	SAPHANADB./B1H/AD_FIAR302,SAPHANADB./BI0/TAC_DOC_TYP,SAPHANADB...	SELECT	0
20,365	SAPHANADB./B1H/AD_FIAR302,SAPHANADB./BI0/TAC_DOC_TYP,SAPHANADB...	COMPILE	0
22,328	SAPHANADB./B1H/AD_FIAR302,SAPHANADB./BI0/TAC_DOC_TYP,SAPHANADB...	AGGREGATED_EXECUTION	1
22,317	SAPHANADB./B1H/AD_FIAR302,SAPHANADB./BI0/TAC_DOC_TYP,SAPHANADB...	SELECT	0
14,540	SAPHANADB./B1H/AD_FIAR302,SAPHANADB./BI0/TAC_DOC_TYP,SAPHANADB...	AGGREGATED_EXECUTION	31
14,514	SAPHANADB./B1H/AD_FIAR302,SAPHANADB./BI0/TAC_DOC_TYP,SAPHANADB...	SELECT	0
21,458	SAPHANADB./B1H/AD_FIAR302,SAPHANADB./BI0/TAC_DOC_TYP,SAPHANADB...	COMPILE	0
18,961	SAPHANADB./B1H/AD_FIAR302,SAPHANADB./BI0/TAC_DOC_TYP,SAPHANADB...	COMPILE	0

Analyzing SAP HANA Expensive Statement Trace results in Tableau

Tableau can be used for analyzing the results of the Expensive Statements Trace. To achieve this, a connection to the 'M__EXPENSIVE__STATEMENTS' view in schema 'SYS' needs to be established. In order to filter for only those statements that are triggered by Tableau, a data source filter can be used (e.g. a Wildcard filter for Tableau on the APPLICATION_NAME Dimension).


Filter [APPLICATION_NAME]

General | Wildcard | Condition | Top

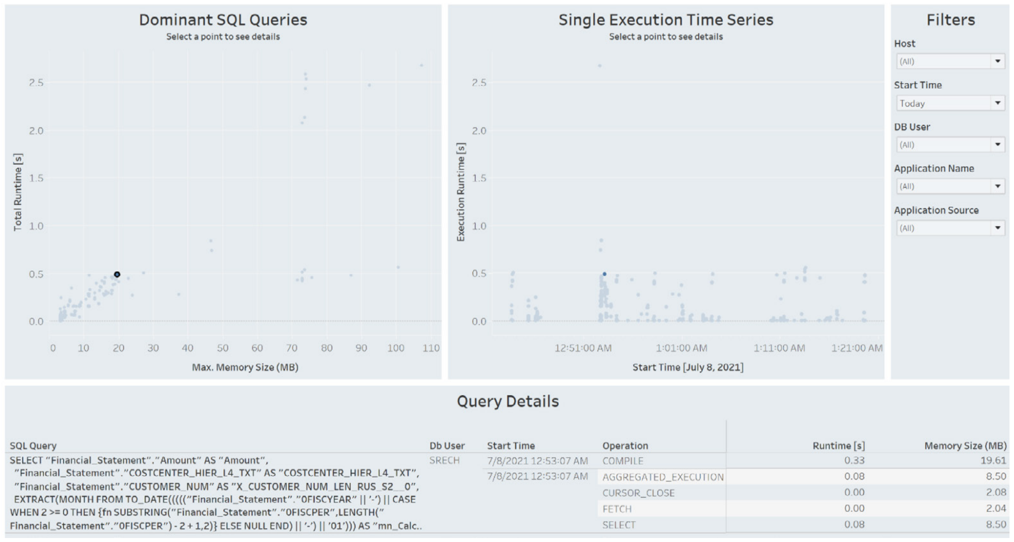
Match value: Tableau

☐ Exclude

○ Contains

● Starts with

An example dashboard for analyzing expensive queries:



Query Details

SQL Query	Db User	Start Time	Operation	Runtime [s]	Memory Size (MB)
SELECT "Financial_Statement"."Amount" AS "Amount", "Financial_Statement"."COSTCENTER_HIER_L4_TXT" AS "COSTCENTER_HIER_L4_TXT", "Financial_Statement"."CUSTOMER_NUM" AS "X_CUSTOMER_NUM_LEN_RUS_S2__0", EXTRACT(MONTH FROM TO_DATE((((("Financial_Statement"."0FISCYEAR" || '-') || CASE WHEN 2 >= 0 THEN {fn SUBSTRING("Financial_Statement"."0FISCPER",LENGTH(" Financial_Statement"."0FISCPER") - 2 + 1,2)} ELSE NULL END) || '-') || '01'))) AS "mn_Calc..	SRECH	7/8/2021 12:53:07 AM	COMPILE	0.33	19.61
		7/8/2021 12:53:07 AM	AGGREGATED_EXECUTION	0.08	8.50
			CURSOR_CLOSE	0.00	2.08
			FETCH	0.00	2.04
			SELECT	0.08	8.50

SQL SQL Plan Cache statistics: Identify optimization candidates (long-running & frequently executed)

The SQL Plan Cache is a valuable tool for understanding the SQL processing of the SAP HANA database. It gives an overview of the statements that are executed in the system and tracks statistics like execution runtimes. As it offers an insight into frequently executed queries and slow queries, it can be used to identify candidates for optimization—without the need to activate a dedicated trace. The SQL Plan Cache can be queried as an SAP HANA View resulting in the ability to use Tableau for the analysis of this information.

Before a SQL statement is executed in SAP HANA, it is compiled to a plan. Once a plan has been compiled, it is better to re-use the plan the next time the same statement is executed rather than compiling a new plan every time. In SAP HANA, the SQL Plan Cache stores plans generated from previous executions. Additionally, it keeps statistics about each plan for monitoring purposes. This allows for you to analyze the number of executions, min/max/total/average runtime, lock/wait statistics, and more.

The following information may be particularly useful:

- Dominant statements (TOTAL_EXECUTION_TIME)
- Long-running statements (AVG_EXECUTION_TIME)
- Frequently executed plans (EXECUTION_COUNT)
- Number of records returned (TOTAL_RESULT_RECORD_COUNT)

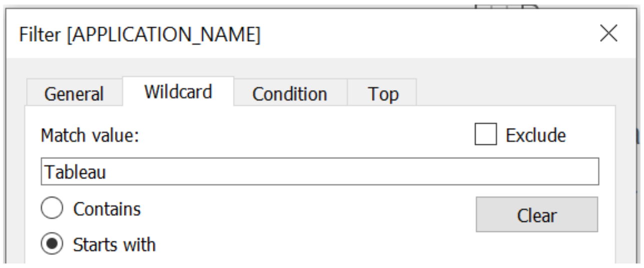
SAP Note 2000002 (FAQ: SAP HANA SQL Optimization) gives additional information how the runtime statistics in the SQL Plan Cache can be interpreted. For example, the operations are broken down into the following actions:

Operation	Description
CURSOR	Contains the overall cursor time including SAP HANA server time and client time; If the client performs other tasks between fetches of data, the cursor time can be much higher than the SAP HANA server time.
EXECUTION	Contains the execution time (open + fetch + lock wait + close) on SAP HANA server side, does not include table load and preparation time.
EXECUTION_OPEN	Contains the open time on SAP HANA server side; Includes the actual retrieval of data in case of column store accesses with early materialization.
EXECUTION_FETCH	Contains the fetch time on SAP HANA server side; Includes the actual retrieval of data in case of row store accesses or late materialization.
EXECUTION_CLOSE	Contains the close time on SAP HANA server side.
TABLE_LOAD	Contains the table load time during preparation, is part of the preparation time.
REPARATION	Contains the preparation time.
LOCK_WAIT	Contains the transaction lock wait time, internal locks are not included.

For a guide on how to read the SQL Plan Cache, see SAP Help's page Example: Reading the SQL Plan Cache.
Recommendations concerning the analysis of the SQL Plan Cache can be found at SAP Help's page SQL Plan Cache Analysis.

Analyzing the SAP HANA SQL Plan Cache in Tableau

Tableau can be used for analyzing the results of the SQL Plan Cache. To achieve this, an SAP HANA connection to the 'M_SQL_PLAN_CACHE' view in schema 'SYS' needs to be established. In order to filter for only those statements that are triggered by Tableau, a data source filter can be used (e.g. a Wildcard filter for Tableau on the APPLICATION_NAME Dimension).

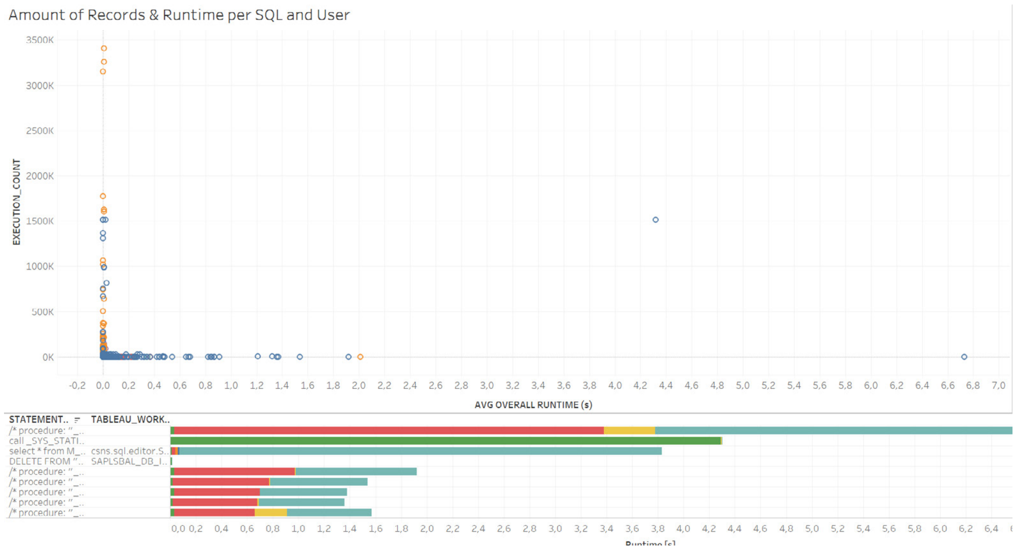

Filter [APPLICATION_NAME]

Match value: Tableau

☐ Exclude

Contains [✓ Starts with]

An example visualization could include the execution count and the average overall runtime with the average distribution of the runtime:

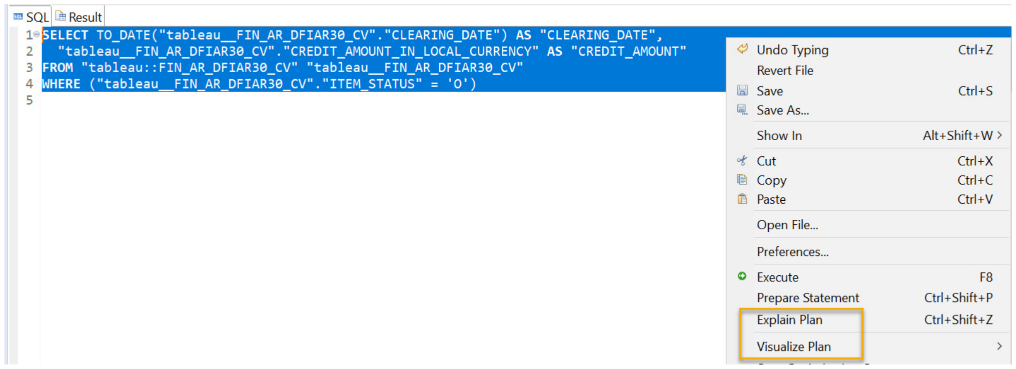


Analyze single SQL queries in detail:
SAP HANA Explain Plan & Visualize Plan

After having identified long-running queries using the Expensive Statements Trace or the SQL Plan Cache statistics, the next step is finding out why the execution takes so much time.

To identify the root cause of the runtime, it is beneficial to understand how SAP HANA handles the SQL Statement execution. The 'Explain Plan' and 'Visualize Plan' features in SAP HANA are two ways to investigate this.

The easiest way to execute either of these is to copy the statement into the SQL Console and choose either 'Explain Plan' or 'Visualize Plan' in the context menu.

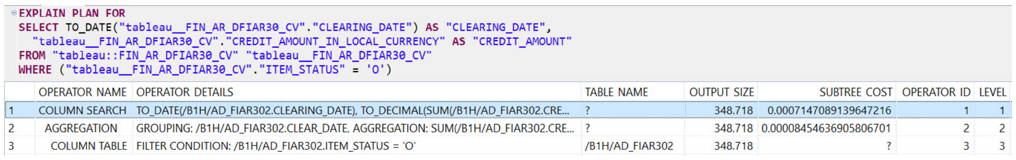


```
SELECT TO_DATE("tableau__FIN_AR_DFIAR30_CV"."CLEARING_DATE") AS "CLEARING_DATE",
  "tableau__FIN_AR_DFIAR30_CV"."CREDIT_AMOUNT_IN_LOCAL_CURRENCY" AS "CREDIT_AMOUNT"
FROM "tableau::FIN_AR_DFIAR30_CV" "tableau__FIN_AR_DFIAR30_CV"
WHERE ("tableau__FIN_AR_DFIAR30_CV"."ITEM_STATUS" = 'O')
```

More detail on each feature is explained in the next two sections.

SAP HANA Plan Explanation

After generating a plan explanation for a SQL statement, the result will show detailed information on the query execution and the database operations involved during the processing.



```
EXPLAIN PLAN FOR
SELECT TO_DATE("tableau__FIN_AR_DFIAR30_CV"."CLEARING_DATE") AS "CLEARING_DATE",
  "tableau__FIN_AR_DFIAR30_CV"."CREDIT_AMOUNT_IN_LOCAL_CURRENCY" AS "CREDIT_AMOUNT"
FROM "tableau::FIN_AR_DFIAR30_CV" "tableau__FIN_AR_DFIAR30_CV"
WHERE ("tableau__FIN_AR_DFIAR30_CV"."ITEM_STATUS" = 'O')
```

	OPERATOR NAME	OPERATOR DETAILS	TABLE NAME	OUTPUT SIZE	SUBTREE COST	OPERATOR ID	LEVEL
1	COLUMN SEARCH	TO_DATE(/B1H/AD_FIAR302.CLEARING_DATE), TO_DECIMAL(SUM(/B1H/AD_FIAR302.CRE...	?	348.718	0.0007147089139647216	1	1
2	AGGREGATION	GROUPING: /B1H/AD_FIAR302.CLEAR_DATE, AGGREGATION: SUM(/B1H/AD_FIAR302.CRE...	?	348.718	0.00008454636905806701	2	2
3	COLUMN TABLE	FILTER CONDITION: /B1H/AD_FIAR302.ITEM_STATUS = 'O'	/B1H/AD_FIAR302	348.718	?	3	3

Some of the key values are described briefly here and in the examples which follow, refer to the EXPLAIN_PLAN_TABLE system view in the SAP HANA SQL and System Views Reference for full details.

Area	Detail
Operation details	The OPERATOR_NAME value shows the type of operation which was executed, such as joins, unions, aggregations and so on. Operations depend on the engine used—essentially row engine or column engine. Dependencies are shown by indentation—see examples below.
Engine	The type of engine where an operator is executed is shown in the EXECUTION_ENGINE column: ROW, COLUMN, OLAP, HEX, ESX.
Table details	Table details include table name, type, size, tables, or objects which were accessed.
Estimated cost	Cost values include the estimated output row count (OUTPUT_SIZE) and the estimated time in seconds (SUBTREE_COST).

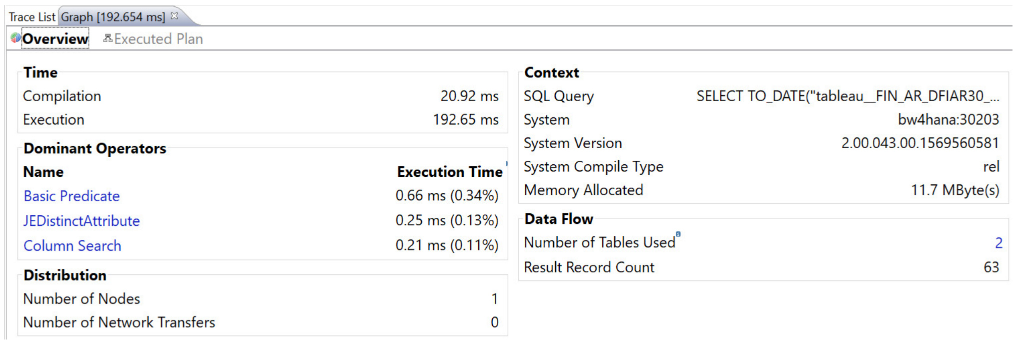
Examples of how the Explain Plan results can be interpreted are described on the following pages:

- Analyzing SQL Execution with the Plan Explanation
- SAP HANA SQL and System Views Reference: Explain Plan Statement

SAP HANA Plan Visualizer

The SAP HANA Plan Visualizer allows for graphically analyzing the SQL execution plan. This makes it easier to understand the steps involved in the processing and how the amount of records and runtime develops over time.

The runtime is given as “Exclusive” (the execution time of the node) and “Inclusive” (the execution time including the descendent nodes) values.



Time

Compilation	20.92 ms
Execution	192.65 ms

Dominant Operators

Name	Execution Time
Basic Predicate	0.66 ms (0.34%)
JEDistinctAttribute	0.25 ms (0.13%)
Column Search	0.21 ms (0.11%)

Distribution

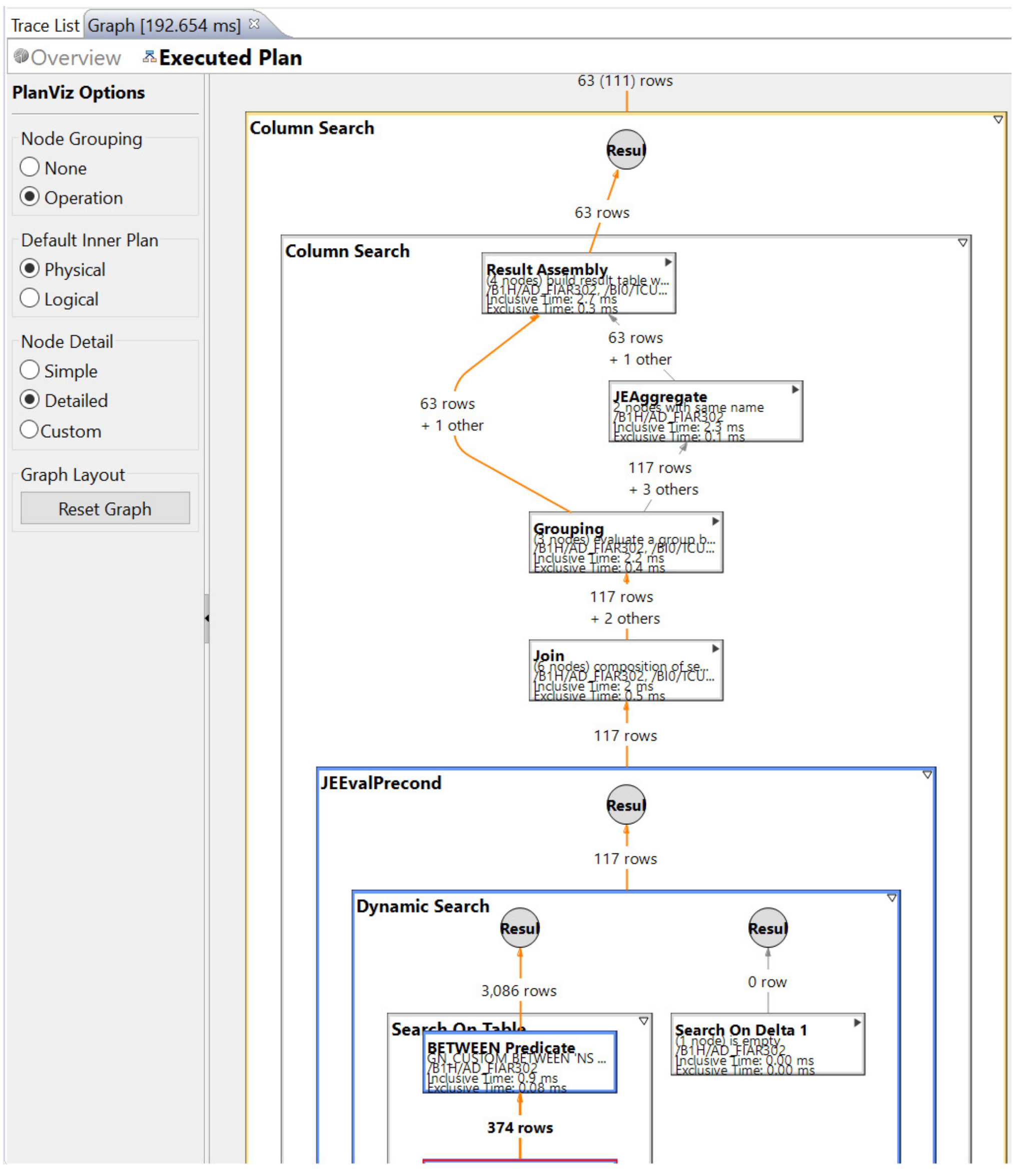
Number of Nodes	1
Number of Network Transfers	0

Context

SQL Query	SELECT TO_DATE("tableau__FIN_AR_DFIAR30_...
System	bw4hana:30203
System Version	2.00.043.00.1569560581
System Compile Type	rel
Memory Allocated	11.7 MByte(s)

Data Flow

Number of Tables Used	2
Result Record Count	63



PlanViz Options

Node Grouping
- None
- [✓ Operation]

Default Inner Plan
- [✓ Physical]
- Logical

Node Detail
- Simple
- [✓ Detailed]
- Custom

Graph Layout
- Reset Graph

The Plan Visualizer offers additional views, e.g. a Timeline View and a Network View for the analysis. More information on those can be found in the SAP HANA Troubleshooting and Performance Analysis Guide in the chapter Analyzing SQL Execution with the Plan Visualizer.

Additionally, SAP released several blogs with instructions on how to use the PlanViz:

- The HANA PlanVisualizer (PlanViz) – Quick and Easy
- Analyzing SQL Execution with the Plan Visualizer (PlanViz)

Once the bottlenecks in the SAP HANA query processing have been identified, they should be addressed. For more information about performance optimization possibilities in SAP HANA Calculation Views, refer to the Optimization Features in Calculation Views chapter of the SAP HANA Performance Guide for Developers and SAP's knowledge base article 2000002 (FAQ: SAP HANA SQL Optimization).
In order to change the SQL query that Tableau sends to SAP HANA, a re-modeling of the dashboard in Tableau needs to be considered. Best practices for dashboard performance can be found in these resources: Designing Efficient Workbooks and Best practices for dashboard performance.

Conclusion

In order to understand the performance of a Tableau on SAP dashboard, use several traces on the Tableau side and SAP HANA side.

For optimizing a single dashboard, an end-to-end execution trace combining Tableau's Performance Recorder and SAP HANA's SQL Trace is recommended. This will reveal how the runtime is distributed across the execution steps and processing layers and help uncover bottle necks.

For long-term performance monitoring, the SAP HANA Expensive Statements Trace is most suitable as it is generating minimal overhead and can remain permanently activated.

Once long-running or memory intensive SQL queries have been identified, they can be further analyzed using SAP HANA's Plan Explanation or Plan Visualizer for understanding the DB execution steps and their runtime impacts.

About Tableau

+ a b | e a u®