



PUBLIC

SAP Datasphere

First Guidance: Development Guidelines and Naming Conventions

Version: 3.0
September 2023



Contents

1 Audience	4
2 Motivation	4
3 General Aspects	4
3.1 Software Versions	4
3.2 General Remarks	4
4 Development Guidelines.....	5
4.1 Landscape	5
4.2 Spaces.....	5
4.2.1 General Remarks	5
4.2.2 Organization of Spaces.....	6
4.2.3 Space Concept with SAP BW Bridge	9
4.2.4 Cross-space Sharing.....	10
4.3 Connections.....	11
4.3.1 Remote Connections Federation Support	11
4.3.2 Remote Connections Architecture	12
4.3.3 Limitation in Remote Source Capabilities	13
4.3.4 Connection Specific Users	13
4.4 Data Modelling.....	14
4.4.1 General Remarks	14
4.4.2 Layered Modeling Approach: Overview	15
4.4.3 Inbound Layer Modelling Guideline	17
4.4.4 Harmonization / Propagation Layer Modelling Guideline	20
4.4.5 Reporting Layer Modelling Guideline.....	23
4.4.6 Corporate Memory Layer (Optional).....	23
4.4.7 Outbound Data Layer (Optional).....	24
4.4.8 Modelling KPIs, Restricted and Calculated Key Figures.....	24
4.4.9 Master Data and Hierarchies	24
4.4.10 Federation vs. Persistency	25
5 Naming Conventions.....	30
5.1 General Remarks	30
5.2 Space Names	30
5.3 Connections.....	31
5.4 Database Users and Open SQL Schema	32
5.5 Folders	32
5.6 Modeling Objects	32
6 Performance Best Practices	35
6.1 General Considerations	35
6.2 Performance for Virtual Access	35
6.2.1 Transfer of Large Amounts of Data	35

- 6.2.2 Option 1: ABAP Layer OLAP Access 38
- 6.2.3 Option 2: Database Layer Access 40
- 6.3 View Modelling 42
 - 6.3.1 Monolithic Modelling 42
 - 6.3.2 Filters on Formulas 43
 - 6.3.3 Data Transposition 44
- 7 Root cause Analysis for performance problems 46
 - 7.1 Performance Root Cause Analysis for View Persistency 46
 - 7.1.1 Memory Analysis 46
 - 7.1.2 Analyzing HANA Execution Plans for Datasphere View Persistency (with examples) 49
 - 7.1.3 Optimization Possibilities for Datasphere View Persistency 57
 - 7.2 Performance Root Cause Analysis for Data Flows 60
 - 7.2.1 Data Integration Monitor 60
 - 7.2.2 Checking Network Activity 63
 - 7.2.3 Potential Performance Improvements 65
- 8 Additional Information 67

1 Audience

This document serves as a guideline for everyone creating (reusable) data models for various source system connections.

The guideline is public because it should serve as a first guidance to customers and partners. It should give guidelines based on best practices and experience from existing projects about development as well as naming conventions.

2 Motivation

SAP Datasphere offers a rich set of features to model data. It also targets an audience that may not be used to modelling in this type of development surrounding. This guide shares best practices that have proven to work in various real-life projects and therefore can help others choose the right modelling approach. SAP Datasphere continues to evolve, so more and more options become available over time. Not every feature has been actively used in this guide, but that does not mean that they should not be used.

SAP Datasphere is an open and versatile solution, which offers a toolset for various modeling approaches and this document should not restrict the ingenuity of an application architect. It should rather hint into the direction of what to think about when starting modeling SAP Datasphere.

3 General Aspects

3.1 Software Versions

This guide is based on SAP Datasphere Version: 2023.18

3.2 General Remarks

The focus of this guide includes space management and data modeling in the data builder. Other areas might be included in future versions.

4 Development Guidelines

4.1 Landscape

The system landscape in the context of SAP Datasphere is a logical group of provisioned SAP Datasphere tenants, linked together by transport routes. Each tenant provides an isolated environment with independent resources and allows to separate development process from the production system usage. Tenants may be also used to physically separate departments, LOB's, data domains, though it is recommended to use Spaces within a tenant for logical workstreams separation instead, see the next section.

Many customer projects have shown that it is beneficial to have more than one tenant of SAP Datasphere, especially for larger implementations. One tenant is typically used for production and another as a development environment. Additional tenants may be provisioned, for instance, to create a separate test environment, or to align with the landscape of remote source systems or connected Analytics systems.

However, one-tenant landscapes are valid options to begin with for proof of concepts, initial projects, or smaller implementations.

There are a few considerations to keep in mind when planning the system landscape:

- Isolation of data and compute resources
- Roles and user management
- Contents lifecycle management
- Integration with external systems

With multiple tenants in the landscape, you may allocate different resources to each tenant. Physical separation of tenants also allows to protect the production environment from the development and tests environment workload, which may be not optimized from performance perspective. In addition, each tenant may have own set of roles and users, which allows you to flexibly manage user access to production data and contents.

The SAP Datasphere Transport System supports cross-tenant transport via Content Network, which simplifies the contents lifecycle management. Using Export and Import features, you can easily move the contents between the tenants, providing that space and connection names are the same.

Integration with remote systems should be also considered from the data separation and lifecycle management perspective. The remote system transports may have their own limitation in re-pointing data models to another source, which may be a factor in SAP Datasphere system landscape decision. For instance, a live connection to SAP Datasphere from SAP Analytics Cloud (SAC) cannot be re-pointed to another SAP Datasphere tenant easily, therefore, to benefit from the SAP Analytics Cloud Transport system, it is recommended to align the environments in Datasphere and SAP Analytics Cloud. Same considerations should be applied to the remote source systems SAP Datasphere connects to.

4.2 Spaces

4.2.1 General Remarks

Spaces provide a way to partition your SAP Datasphere tenant into independent virtual work environments for departments, LOB's, data domains, project teams, and individuals. By default, spaces are strictly separated from each other in terms of user access, data access, (data model) namespace, storage resources, etc. so that multiple teams inside an organization can work independently from each other in a secured environment.

Cross-space sharing allows you to make objects of a particular space available for consumption in other spaces for other teams. Objects in the Data Builder can be shared with one or several other spaces in the

same SAP Datasphere tenant. This works either one by one on an object level or multiple objects at a time using the landing page in the Data Builder for one space. Alternatively, you can also share multiple objects in the repository explorer.

4.2.2 Organization of Spaces

Spaces are virtual work environments and are required before you start to model your data. For more information about space settings and space management feature please check the [SAP documentation](#).

It is recommended to use **more than one space** within your tenant, even if you want to keep all your data within one space. The other space would be used for administration, monitoring, security, etc. as explained below. Hence, authorizations are one criterion to whether you want to create a separate space or not. Other criteria could be source system access and the load on the source system. For instance, you want to avoid accessing the same core tables of your SAP S/4HANA system like ACDOCA from multiple spaces using different access methods (remote, replication) and that way create heavy load on that system.

There are a few areas of sensitivity to which not every user of your tenant may get access to:

1. Permission tables/views for Data Access Controls (DAC)¹
2. Access to Audit Logs and Activities²
3. Access to Monitoring Views³

You can also setup multiple dedicated spaces for these areas to further segregate access to particularly sensitive data sets.

4.2.2.1 Option 1 – Minimum Setup

Create a dedicated space for the sensitive topics, which is called Administration in Figure 1.

The Administration space is used for:

- selected people only have access, like
 - Administrators responsible for authorizations and (performance) monitoring
 - Auditors to access Audit Logs and Activities
- performance monitoring
- analyze logs, activities and audit information
- central management of tables with authorization values to be protected against unauthorized maintenance
- exposing authorization tables/views as read-only to the Target Spaces

¹ Find more information in the [Data Access Control documentation](#) and [Row Level Security document](#).

² Find more about [audit logging](#) and [repository activities](#) in the documentation.

³ Find more information about [Monitoring Views](#) in the documentation.

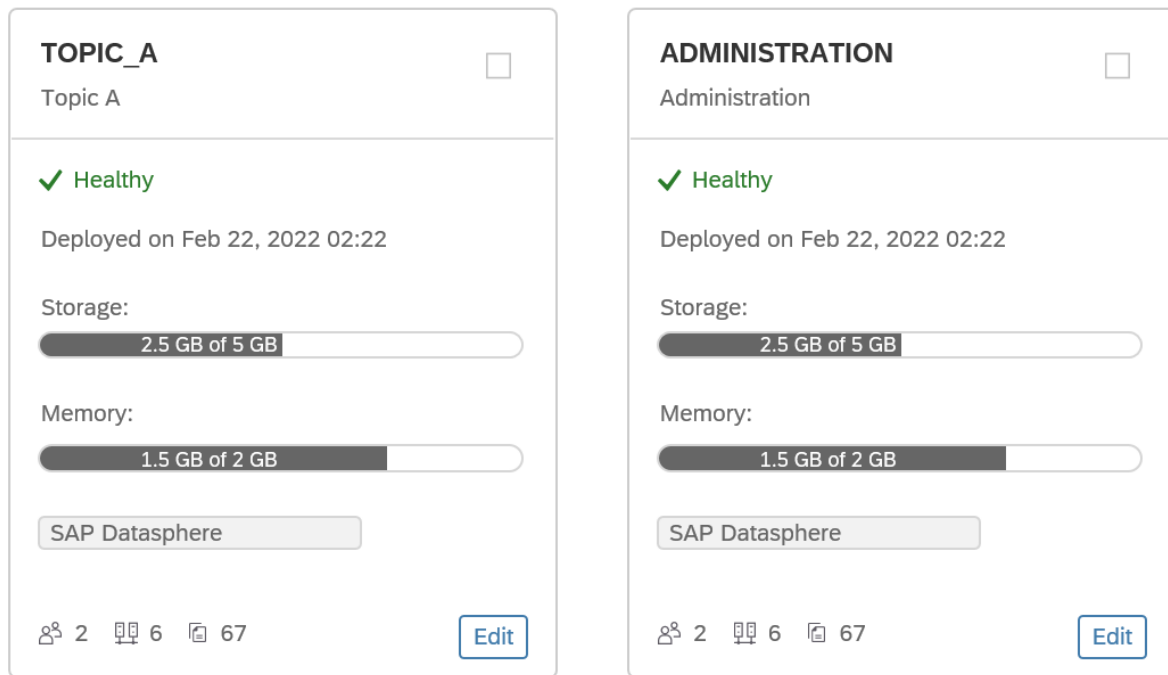


Figure 1: Minimum space setup

Recommendation, if Data Access Controls (DAC) are used in the Target Space (see Figure 2):

- only (Security) Administrators should have authorizations to create Data Access Controls in the Target Spaces
- Any user in Target Space can use the locks provided by the (Security) Administrator to lock their views
- Members of the Target Spaces cannot change the authorized values in the Administration Space, **but could potentially modify** the permission entity (a view consuming the shared artifact) which is used by the Data Access Control or even completely remove the Data Access Control from data models

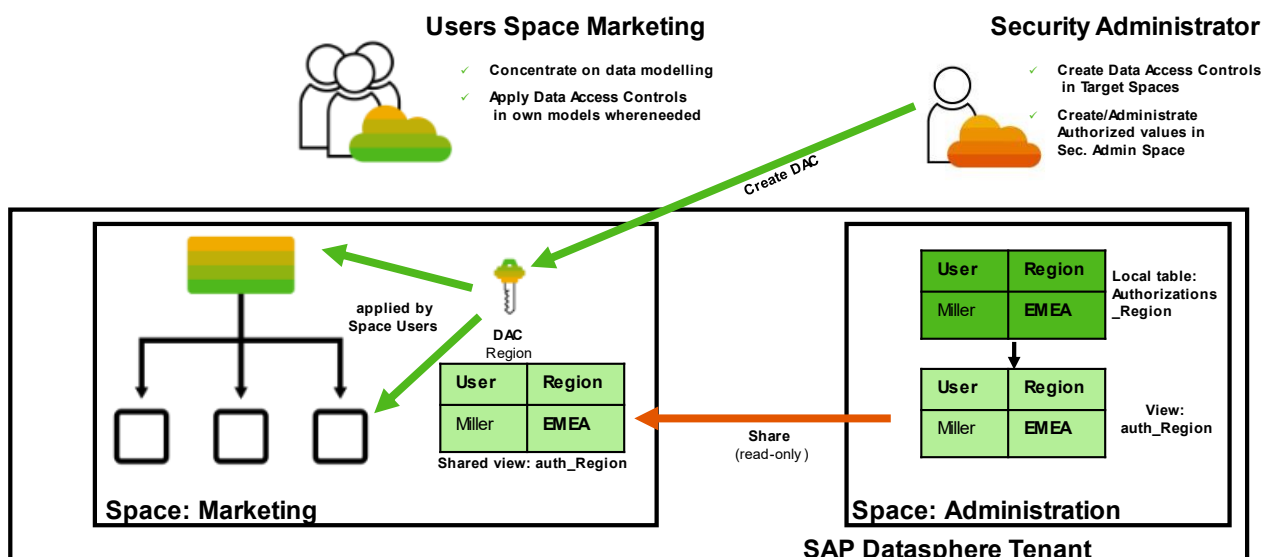


Figure 2: Data Access Controls and spaces

4.2.2.2 Option 2 – Central Governance Setup

Create multiple dedicated spaces for the sensitive topics like permission tables, audit and activity log access, (performance) monitoring or other administrative topics. These spaces might be called Administration and Permissions like in Figure 3.

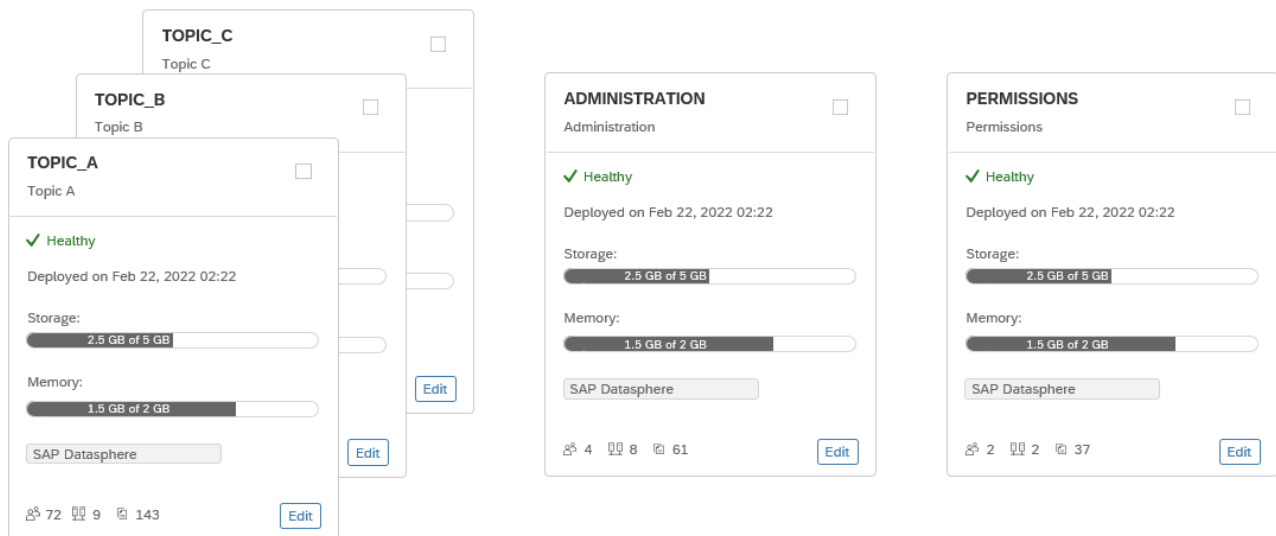


Figure 3: Central governance space setup

The Administration space is used for:

- selected people only have access, like
 - Administrators responsible for authorizations and (performance) monitoring
 - Auditors to access Audit Logs and Activities
- (performance) monitoring
- analyze logs, activities and audit information
- Developers or (Security) Administrators create DACs and assign them to views on transactional data tables to protect access to data
- Protected views are shared to Target Spaces (e.g. Marketing Space). That way the authorized values are completely separated from Business Department.
- Benefit: Strict separation of authorized values to a small group of selectively named users (e.g. in Business departments or Security Admins) to maintain authorizations by providing flexibility for developers and secure access for pure consumption users to the relevant data only

The Permissions space is used for:

- central management of tables with authorization values to be protected against unauthorized maintenance
- only Administrators responsible for authorizations should have access to this space
- exposing authorization tables/views as read-only to the Administration space

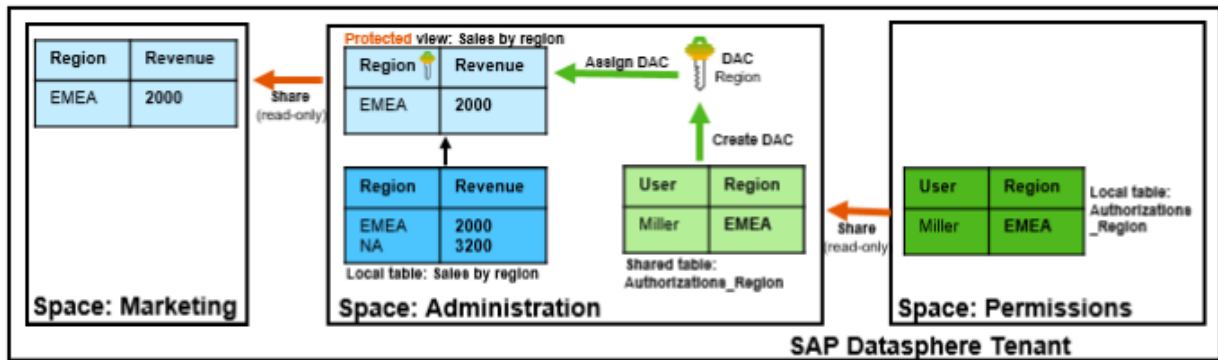


Figure 4: Data Access Controls in a central governance space setup

4.2.3 Space Concept with SAP BW Bridge

Customers who have licensed the corresponding SAP BW bridge on top of SAP Datasphere automatically receive a corresponding SAP BW bridge space in their SAP Datasphere tenant. This serves as an inbound space and reflects all remote tables based on entity imports in SAP Datasphere.

In order to continue to have strict governance as in classic SAP BW, there is a special space for the model import from SAP BW bridge recommended.

In the so-called “[IT] Governed SAP BW bridge”, all queries and InfoProviders are imported via the entity import (see Figure 5). The corresponding artifact can then be shared with the target space via cross-space sharing. Thus, the original always remains in a central space and can be clearly managed by IT.

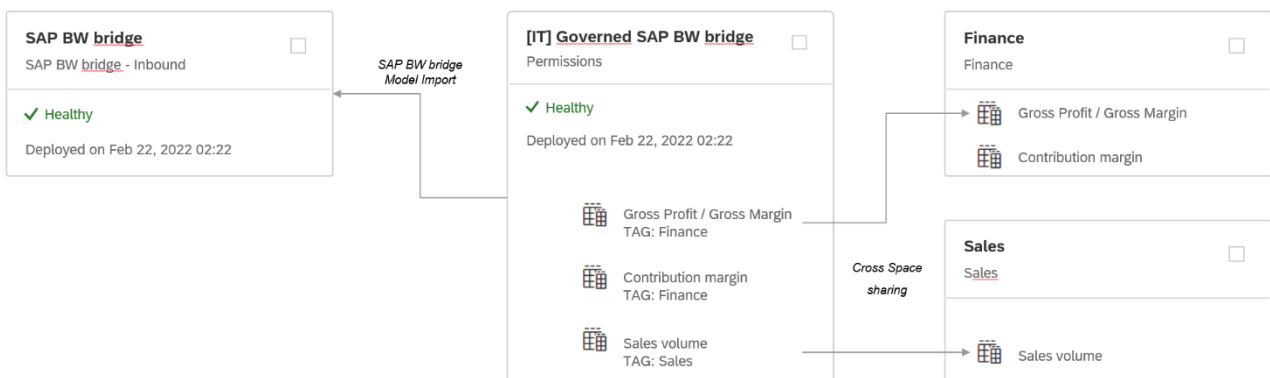


Figure 5: Spaces with SAP BW bridge

4.2.4 Cross-space Sharing

Sharing across spaces follows the concept to store data once and use it in many different contexts. There is no limitation as to what type of model you can share. This brings up a few thoughts to consider.

As an example, there are spaces A and B and a user of space A wants to share models with users of space B (see Figure 6). In space B the shared model will not be part of the model list, but rather it will appear as a repository object of type table or view in the data modeling editors of the data builder. This means the model loses a bit of its context, which directly affects user decisions when modelling with the shared object. A situation that could occur is that the model from space A has a calculated column and relies on restricting data, for fine granular reporting. This model is shared with space B and the application designer from space B uses the restricted column in a selection criterion. The calculation needs to happen before the restriction. Therefore, the filter value is never used in the source. This can lead to a severe performance impact and undesirable memory consumption on the system.

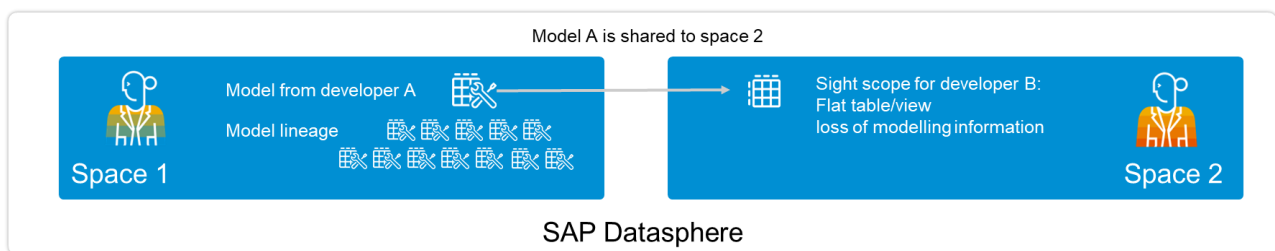


Figure 6: Cross-space sharing

To avoid situations like this, system wide naming conventions and design rules for shared objects should be followed. Naming conventions (see chapter 5 Naming Conventions) make search and classification of models easier and if these are known system wide, it even allows to pass on certain information about a model to developers outside of a space.

Columns with calculations could be named with “CM_” to signal to another developer, that it is calculated measure or “CD_” in case it is a calculated dimension. A model named to fit a specific application layer can signal to another developer the purpose of the model. A model from the reporting layer shared with another space should be indicated to signal, there may be complex logic behind. Models from the inbound layer shared to other spaces, on the other hand, supply raw data, eventually from a specific source or even virtually via the network. You find more information on the Layered Modeling Approach in section 4.4.2 and Naming Conventions in chapter 5 of this document.

Especially in a central governance setup a set of rules for cross-space management can be setup by the central administration. These rules can also be publicized by building a small model in the administration space stating the rules. This model can be shared with all other spaces for the purpose of looking up the central guidelines.

4.3 Connections

4.3.1 Remote Connections Federation Support

SAP Datasphere supports remote connections to various source systems. The full details about the [Supported Connection Type Features](#) are described in the SAP Help documentation.

Looking only at the support for remote table federation Figure 7 gives an overview about SAP systems and Figure 8 about non-SAP systems supporting federation. These figures also show if a connection requires a middleware component like the DP Agent or a Cloud Connector (CC), or if a direct connection can be established.

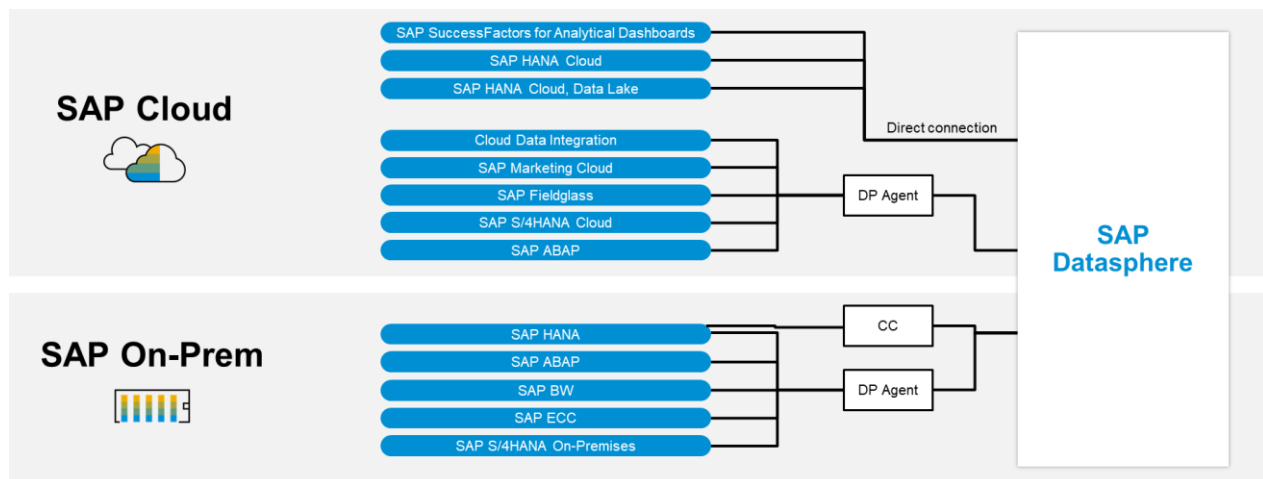


Figure 7: Connection Types Supporting Remote Table Federation (1)

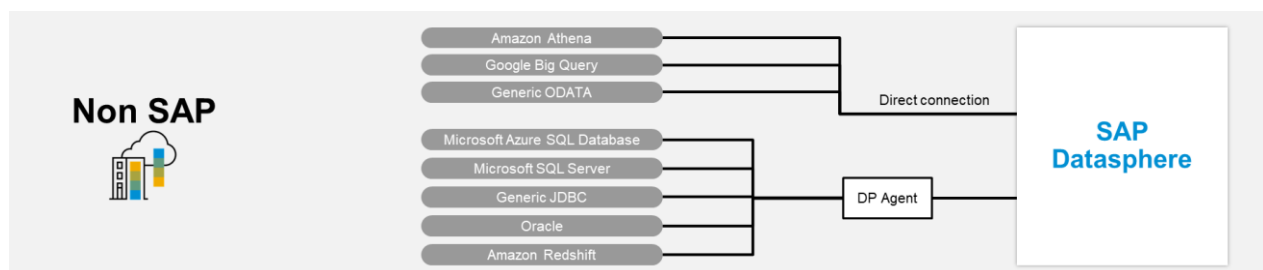


Figure 8: Connection Types Supporting Remote Table Federation (2)

Figure 9 lists the source system connections with no support for federation.

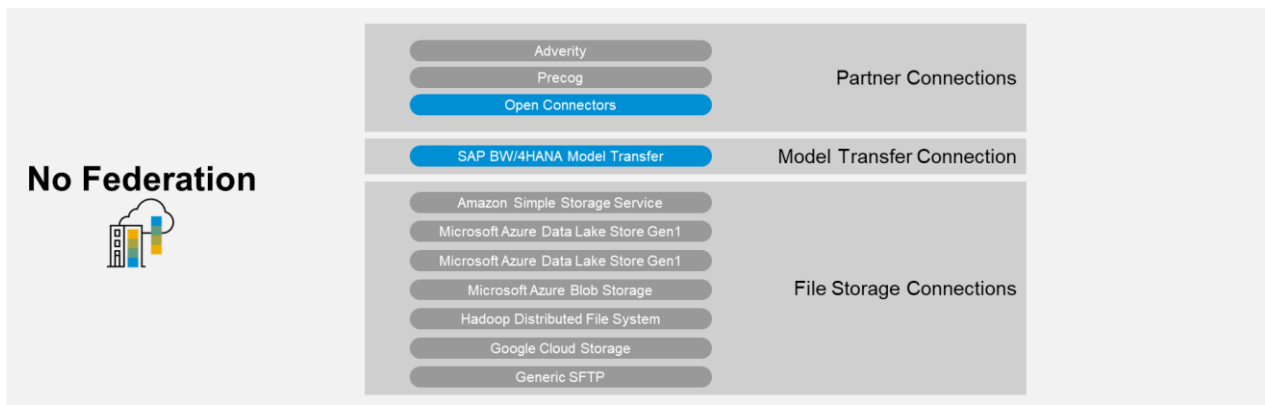


Figure 9: Connection Types Without Remote Table Federation Support

4.3.2 Remote Connections Architecture

SAP Datasphere leverages various services behind the scenes. Therefore, it is worth to look at the architecture (Figure 10). SAP HANA Cloud is used for storage, with Smart Data Integration (SDI) and Smart Data Access (SDA) for connections to remote tables. This includes the dpServer and dpAgents for SDI. For the Data Flows the DI Embedded Engine uses their Adapter/connectors to connect to the source directly or via Cloud Connector.

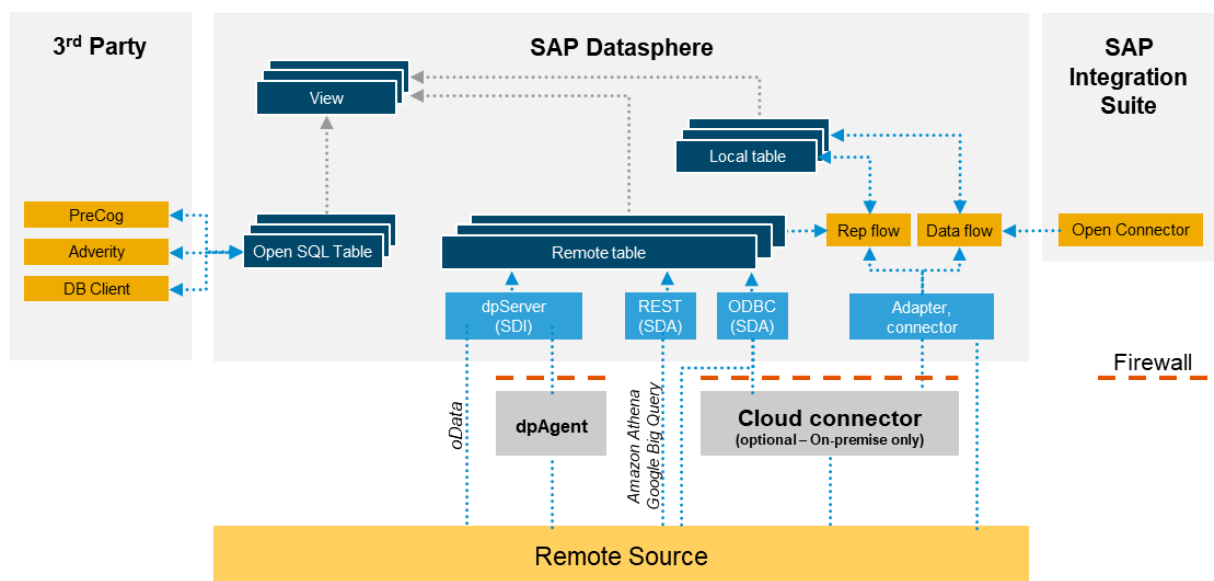


Figure 10: Federation Architecture

4.3.3 Limitation in Remote Source Capabilities

The connections of SAP Datasphere support various features (details see [Supported Connection Type Features](#)). Differences in supported features can have several reasons:



Figure 11: Limitation in Remote Source Capabilities

- **Source**
the remote entity can have limits in SQL capabilities they support. E.g. an ABAP Extractor is an ABAP Function Module, that cannot Aggregate, accept JOIN Push Downs, Sort, TOP, etc.
- **Adapter**
A Connector/Adapter can also have limited capabilities. E.g. Camel JDBC does not support LIMIT, even if the underlying MySQL does.
- **Model**
SQL Models can have logical constraints that prevent a push down of a SQL artefact. E.g. INNER JOINs act as filter, so a TOP in the statement can only be applied once the join result is known.

4.3.4 Connection Specific Users

Typically, one connection per source system would be used to avoid redundancies for remote tables.

But there are scenarios, where the connection only exposes a limited or specific data set e. g. cost center controlling. For these specific scenarios it makes sense to restrict the authorizations of the (technical) user that is used for the connection in the source system. In such a setup, the number of tables and views accessible from the source are restricted. This also limits the data transfer as well and can add another level of security.

Another reason to limit the access of the technical user to the source system may be out of performance reasons. If the source offers analytic privileges or another form of row-level security, the data extracted can be steered by the source system to limit resource consumption while accessing models and objects.

Example: Datasphere is accessing HANA views generated by a BW/4HANA system. The system contains a history of the last 15 years of financial controlling data, most of which is archived to a Near-Line Storage on an SAP IQ. Nearly all of the ad-hoc analysis of the Datasphere models from a certain space need only YTD figures. The rest of the historical reporting can be done with BW-live connections. In this case the technical user from the financial controlling space could be limited using analytic privileges on the current calendar year. In this case an end user cannot request by mistake all 15 years.

Another example: According to the Best Practices and Sizing Guide for Smart Data Integration⁴, it is recommended to use multiple connections (remote sources) connecting to the same remote source to distribute the Remote Table load coming from real-time replication with larger data volumes across the connections. You may also use multiple connections to separate real-time replication for critical tables because any replication exception causes the replication to stop for all subscriptions (tables replicated in real-time) based on the same connection because SDI tries to preserve transactional integrity.

⁴ Find more information in the [Best Practices and Sizing Guide for Smart Data Integration when used in SAP Data Warehouse Cloud](#) document, section Multiple Connections / Remote Sources.

4.4 Data Modelling

4.4.1 General Remarks

We recommend considering the modeling of reusable views instead of creating large, monolithic views. Divide your views in the data layers to basic views and reuse these basic views in more complex views.

Datasphere is powered by HANA Cloud, therefore the best practices in modelling of HANA apply to Datasphere as well. HANA has different engines that influence the execution of a query. The engines optimize the query to execute in the best possible way. The purpose behind this is to make artifacts reusable. In theory, when building a consumable model, to optimally execute it would be necessary to build a model for each execution purpose. When using modular views and combining them with each other, this may lead to sub-optimal overall runtime, if executed exactly the way designed. Therefore, the HANA optimizer will rearrange execution sequences. General rules should be followed (see Figure 12), such as:

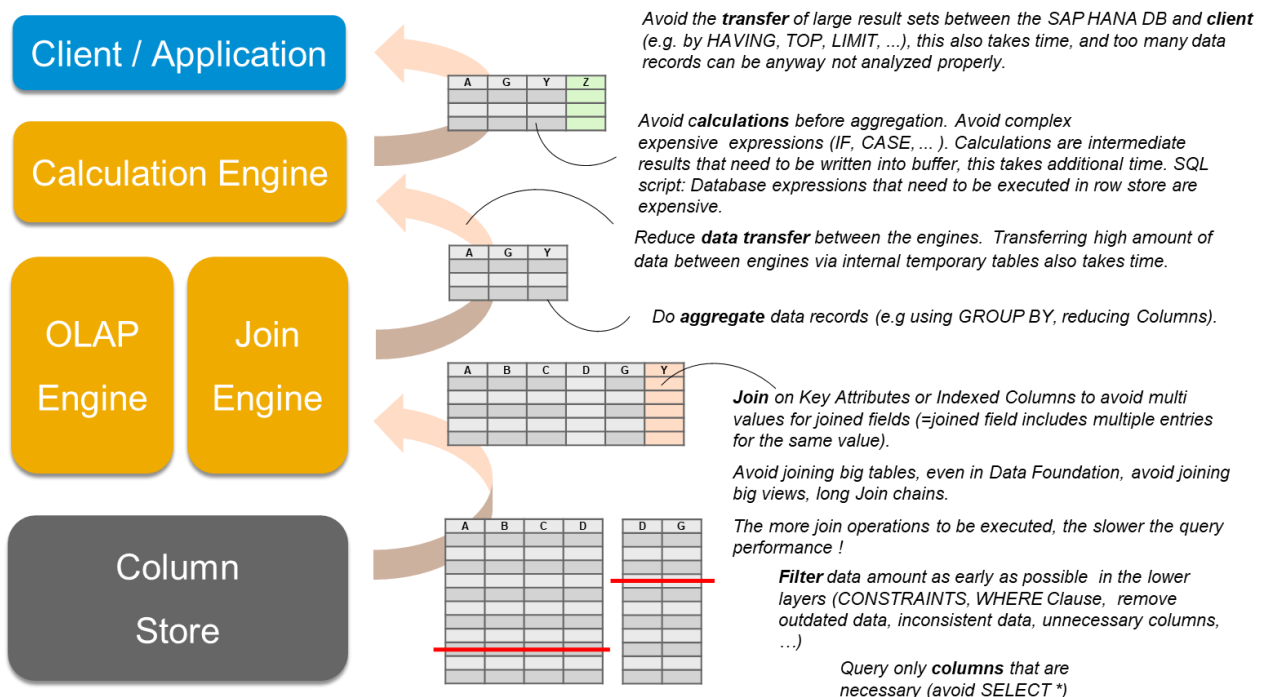


Figure 12: SAP HANA execution sequence

The [SAP HANA performance guide for developers](#), is a central best practice document for HANA development, especially SQL Scripts and other SQL based modelling guidelines can be viewed in more detail.

Considerations of joins and associations:

- For performance reasons, it is better to associate the master data to transaction data instead of creating joins. Language dependent texts should also be associated using the text association for the master data (semantic type "Dimension"). The time dimension should also be associated.
- If master data, text or time objects are required for analytics, then they should be associated. Joins should be used only when the objects are required for the modeling.

Considerations of data types:

- For performance reasons, it is recommended to use String and Binary data types over LargeString and LargeBinary. Not only because they are stored separately in the LOB store, but also because some operations (like GROUP BY) are not allowed on LOB types.

4.4.2 Layered Modeling Approach: Overview

Concepts of modeling in data warehouses in general will often introduce so-called layers. Layers are logical groups of entities (tables, views, data access controls, etc.).

A well-established layering approach follows the flow of the data as shown in Figure 13:

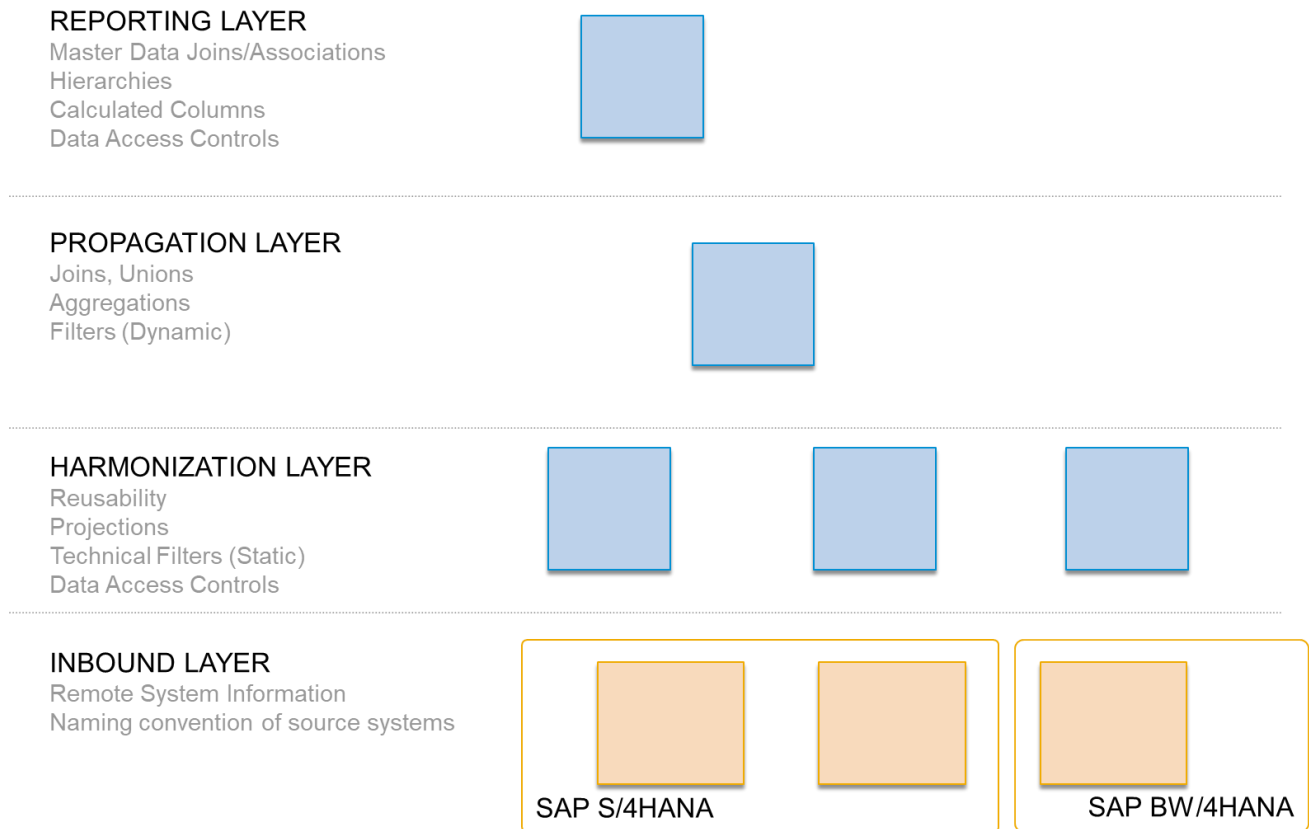


Figure 13: Layered modeling approach

- **INBOUND LAYER**
 - data close to the source systems or data sources
 - can be the first view with virtualized access using remote tables, a data persistency using real-time replication, Data Flow or an external ETL-tool
 - inbound layer view with 1:1 mapping from source; filters and projection only where necessary; without any data manipulation.
- **HARMONIZATION LAYER**
 - objects on top of the inbound layer with a strong focus on reusability
 - in this layer the individual data sets get harmonized in terms of standardized semantics from different systems e.g. RBUKRS from SAP ECC and 0COMP_CODE in SAP BW. Unifying column names early helps a lot during modelling later, since automated mapping goes via the column name.
 - the harmonization layer is often portrayed explicitly as the preparation layer for the objects in the propagation layer

- use projections (column selection, column renaming, only SUM) to harmonize field names from different systems (e.g. RBUKRS from SAP ECC and 0COMP_CODE in SAP BW) and to apply static technical filters
 - use Data Access Controls to restrict users from accessing all data
- PROPAGATION LAYER
 - objects on top of the inbound or harmonized layer providing semantic and value standardization of data from various sources, thus offering data in a highly harmonized form
 - in this layer the individual data sets get combined with joins and unions, data is aggregated if required and dynamic filters are applied
 - master data objects and their semantics are modeled in this layer including text and hierarchy associations
 - data in this layer is consistent, complete and generally free of redundance (except for performance reasons or complicated modelling tasks)
 - the propagation layer can also be used for persistent views to serve the persistent architected data marts and is a basis for the reporting layer
 - Create facts and associate them with dimensions for further use in reporting layer. One thing to note here – Analytic dataset semantic use is deprecated and replaced by Facts. This means Analytic dataset can still be used but going forward the use of Fact is recommended. Although, Analytic dataset was viewed as reporting object, but Fact will be used as an object in propagation layer to create associations with Dimensions. More on this can be found at - <https://blogs.sap.com/2023/06/29/why-are-analytical-datasets-deprecated-and-what-does-it-mean-for-your-project/>.
- REPORTING LAYER
 - typically objects that are accessed by reporting tools such as SAP Analytics Cloud
 - this is where master data joins or associations will be applied
 - hierarchies, exception aggregation, restricted and calculated columns are modeled here
 - Analytical data sets are deprecated and no longer recommended for new modelling. Please use new modelling object Analytic Models for multi-dimensional and semantically rich analytical modelling. More on Analytic Models can be found at - <https://blogs.sap.com/2023/03/13/introducing-the-analytic-model-in-sap-datasphere/> .
 - use Data Access Controls for further reporting relevant restrictions on data sets

For using Analytic Model within SAP Analytics Cloud, optimized design story mode needs to be used instead of classical story mode. In new modelling paradigm, semantic object Facts is recommended, with Analytic Model on top of it for exposing the data for consumption in SAP Analytics Cloud. Looking at the layers from an analytics perspective, access to all layers is allowed from SAP Analytics Cloud (see Figure 14). For simple scenarios, Semantic object Fact can be created in Inbound layer directly on the table but as per best practices it is recommended to create it in propagation layer following the layered modelling approach mentioned above. In such a simple setup this Analytic model created on top of table with semantic usage as Fact would represent the reporting layer. You can add further layers based your requirements and modelling preferences.

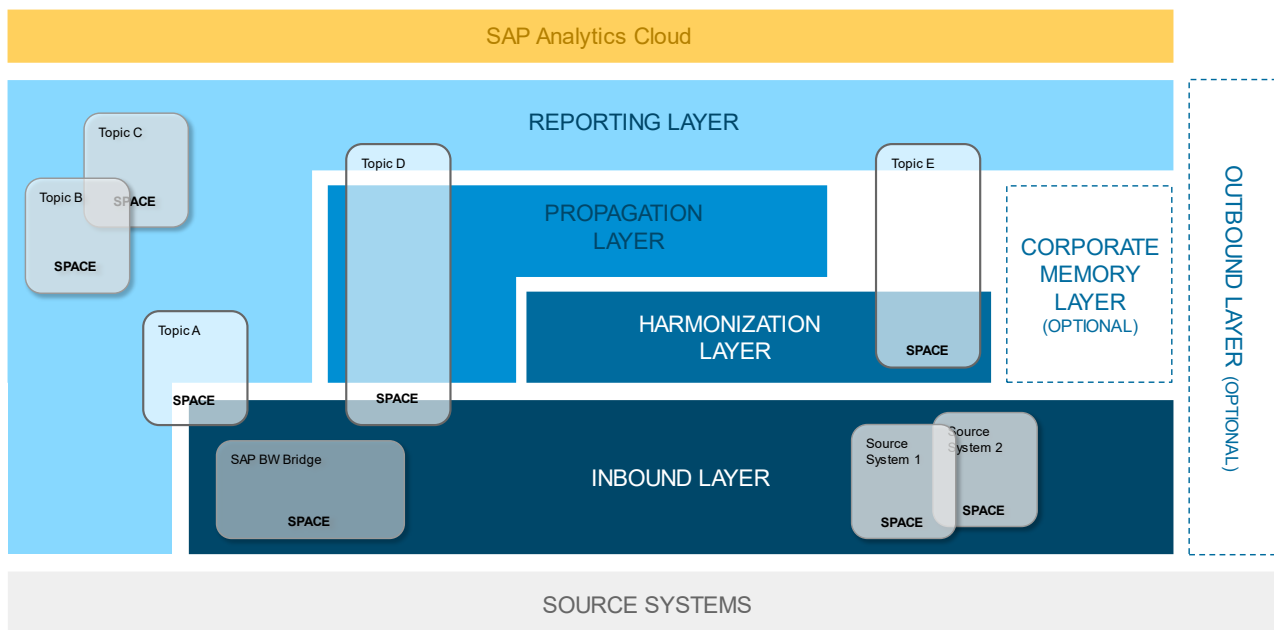


Figure 14: Access of layers from SAP Analytics Cloud

4.4.3 Inbound Layer Modelling Guideline

The inbound layer has some very specific challenges as it is the point where you communicate with ‘the outside world’ which is by nature diverse. In this chapter we focus on views and the SDI (Smart Data Integration) technology for communication. Some adapters will handle federation quite easily, while others struggle to push down something to the remote side, notably the ABAP adapter.

As a rule of thumb, if you are thinking about using federated data, use a database connector, not an application connector like ABAP. You will want to make it possible for SDI to push as much as possible to the remote database and move as little data as possible over the DP agent to Datasphere.

A general overview of Adapter capabilities and supported functions can be found here: [SDI Data Provisioning Adapters](#). You may get more details about the different levels of adapter capabilities and about capabilities of specific Adapters in [Data Provisioning Adapter SDK Guide](#), check section Capabilities.

This falls under ‘*Filter data as early as possible*’.

When you think about persisting, from a data traffic point of view, the best option is real time replication (subscription in SDI terms). This will have an initial load that moves a lot of data, but once it’s there, only the changes will have to be ‘replayed’ in Datasphere. Making it the option with the least load on both the source system and Datasphere.

Obviously if data is persisted in the inbound layer, every layer above will benefit from that.

- In all cases (both federation and persisting) reduce amount of data as early as possible using where clauses/filters on (remote) table level. Remember however, that filtering / where-clause execution takes additional time, so make sure that the data filtered out is significant (> 10% of the rows) in case of federation.
- HANA works with a standard expected number of rows for remote tables, obviously the tables on the remote side are not one size fits all. Use statistics on the remote table to allow the HANA optimizer to find the best execution plan.

You can create these statistics in the Data Integration Monitor > Remote Query Monitor > Remote Table Statistics, select a table then in the next screen select *Create Statistics* (see Figure 15).

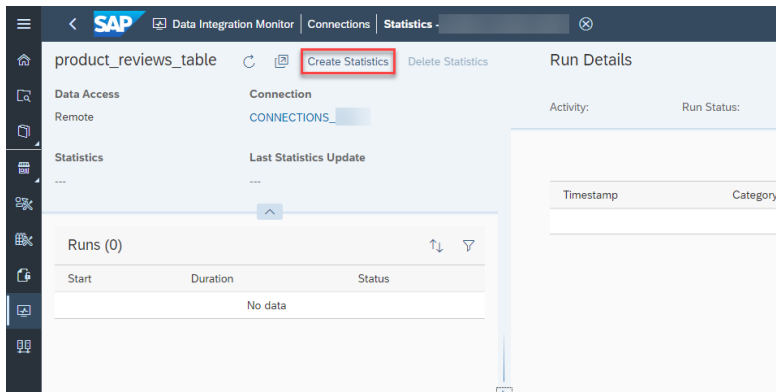


Figure 15: Remote table statistics

The statistics are available in different versions (see Figure 16):

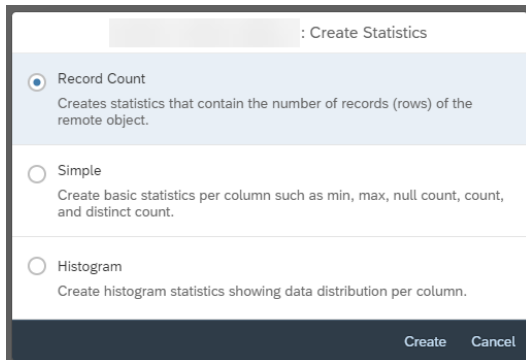


Figure 16: Options for remote table statistics

Note that not all statistic forms work with all source tables, some data types will not be usable for statistics. As we cannot direct Datasphere to specific columns, record count is the only option left in these cases.

Some specific performance killers when data is used federated.

- **Derived columns**
The direction is to push complex calculations/expressions to the ETL (derived column, calculate once and persist). In a federated approach this is not possible, however always make note of these candidates as this might change.
- **Data type transformation**
While it can be tempting to transform a data type at the source, so it's harmonized with the other sources already, this makes the push down of filters or joins on those transformed columns difficult for most adapters and it also hide the best format for the consumers of this layer, so resist the urge to do so. Consider the following real customer example

We have an ABAP table in an SAP source system, out of which we want to pick up yesterday's records;
We also have a transformation from the ABAP date (string of 8 characters) to a SQL date which is the date format we want in transformation layer. Now we notice that selecting where the date is put in as a fixed string, this is pushed down to the remote side, retrieving only 3 records, but when using a construct like 'ADD_DAYS(CURRENT_DATE,-1)' for the exact same date, it retrieves all records

and filters in Datasphere.

CONNECTION_ID	TRANSACTION_ID	STATEMENT_ID	REMOTE_CONNECTIO	REMOTE_SOURCE_SC	REMOTE_SOURCE_NW	START_TIME	END_TIME	FETCHED_RECORD_C	FETCHED_SIZE	REMOTE_STATEMENT
262709	1254	112832602483106	0	NULL	DWC_M0G5.P11	2021-11-01 14:33:08...	2021-11-01 14:37:05...	188826	5367652395	CLOSED
262521	1343	1127519148630570	0	NULL	DWC_M0G5.P11	2021-11-01 14:31:37...	2021-11-01 14:31:38...	3	1221	CLOSED
262923	4945	1129245725478367	0	NULL	DWC_M0G5.P11	2021-11-01 14:22:18...	2021-11-01 14:26:06...	188823	4784418920	CLOSED
262826	7250	1128829113646884	0	NULL	DWC_M0G5.P11	2021-11-01 14:14:44...	2021-11-01 14:21:41...	188821	5367637875	CLOSED
262723	752	1128386732011834	0	NULL	DWC_M0G5.P11	2021-11-01 14:07:48...	2021-11-01 14:12:13...	188820	4784411550	CLOSED
262709	1254	112832602483102	0	NULL	DWC_M0G5.P11	2021-11-01 14:07:12...	2021-11-01 14:07:14...	3	1118	CLOSED

has to be done on Datasphere side. An implicit conversion from date to string (the remote format) does not happen as a date data type is preferred by HANA over a string data type.

- Selective (inner) join with data not in the same database (or adapter incapable to push this)
The only way a join like this could be executed on the remote database and bring back only those rows we are interested in, would be to create a temporary table on the remote side and put that data in there, then join it. Obviously, it cannot do that in most databases, as it doesn't have the necessary rights. So Datasphere will do what it can, bring the full data and do the join and selection in Datasphere. Be prepared to move away from federation if the performance is unacceptable.
- Selective WHERE-conditions with sub-select in a SQL View
Sub-select in the WHERE-condition is implicitly converted into an inner join by SAP HANA optimizer. Consider the SQL view below:

```
SELECT "COLUMN3" FROM "TABLE1"
WHERE "COLUMN1" = (SELECT "COLUMN2" FROM "TABLE2" LIMIT 1)
```

After the optimization, it will be converted into a statement similar to

```
SELECT "COLUMN3" FROM "TABLE1" T1 INNER JOIN
(SELECT "COLUMN2" FROM "TABLE2" LIMIT 1) T2 ON T1."COLUMN1" = T2."COLUMN2"
```

If the Adapter of the "TABLE1" connection does not support join relocation, this statement cannot be pushed into the remote database. Therefore, Datasphere will have to create a temporary table, pull all the data from the remote table "TABLE1", and perform join operation in Datasphere. Pulling large data set from "TABLE1" may have performance implications.

You may improve the performance by replacing the SQL View with a Table Function:

```
DECLARE lv_filter <COLUMN2_data_type>;
SELECT "COLUMN2" INTO lv_filter FROM "TABLE2" LIMIT 1;
RETURN SELECT "COLUMN3" from "TABLE1" where "COLUMN1" = :lv_filter;
```

In this case, the sub-select expression will be calculated first, and the statement sent to the remote database to pull TABLE1 data will have the filter applied:

```
SELECT "COLUMN3" FROM "TABLE1" WHERE "COLUMN1" = ?;
```

If the filter is selective, it may significantly reduce the amount of data fetched from the remote system for "TABLE1".

- Joins/Unions in this layer
This is not so much a performance problem as a potential one, we expect the shared model of this layer to be directly translatable to a single source table, either federated or persisted. Which means that your consumer assumes a filter or join condition on a column is not expected to have a problem to push down. However, joins and unions might prevent this.

4.4.4 Harmonization / Propagation Layer Modelling Guideline

The harmonization layer is where you have the most flexibility to get the end result you want to achieve, so this is also where modeling keeping performance in mind has the most impact.

When modeling this layer, always measure every step as you slowly go from simple to complex. When you find out performance is not good enough in a final model and you have to backtrack which join/union/where condition is causing the problem it takes a lot more effort.

Here are some small things that will help performance.

- Use a where condition on a joined result set instead of an inner join to achieve filtering. This performs better.
- Avoid joining on a calculated column. Intermediate results from calculations that are not natively supported by the column engine are materialized, which may lead to higher memory consumption. Join on materialized columns if possible.
- Avoid filtering on a calculated column. Filter on a calculated column cannot be pushed down to the table as calculations have to be performed first, materialized, and only then the filter may be applied.
- Avoid implicit type casting in queries. If there is a comparison between two values or columns of different types, the system performs an implicit type-casting operation on one of the columns depending on the precedence⁵. Depending on the column selectivity, it may be beneficial to explicitly convert another column data type to process less records.

For instance, expression `column_date_string < CURRENT_DATE` is implicitly converted into `TO_DATE(column_date_string) < CURRENT_DATE` and leads to type casing for all the values in the column `column_date_string`. On the other hand, explicit conversion `column_date_string < TO_NVARCHAR(CURRENT_DATE)` will have to perform type casing for one value only.

- Use where conditions in the join ON-clause to filter data in a joined in table (also works for outer joins) instead of an inline select. This gives the query optimizer a better chance to find the optimal model as the SQL gives more details.
- Rewrite functions that use columns from different tables.
For instance, if a where-condition `TABLE_A.A + TABLE_B.B = constant` is used, it requires calculation of the expression `TABLE_A.A + TABLE_B.B`, which mandates to perform join between the tables `TABLE_A` and `TABLE_B`. From performance perspective, it is recommended to use a where-condition `TABLE_A.A = constant - TABLE_B.B`, where both parts of the predicate may be calculated on each table independently before the join.
- Use NOT EXISTS operator instead of NOT IN due to performance reasons.
- Embed business logic in helper views once you've found the best way of calculation. It makes sense to reuse both for performance and maintenance reasons.
- Minimize complex expressions, especially avoid SQL that requires the row engine like Window functions.

An example on how you can achieve things that are typically done by using correlated subquery or window functions; Consider the table KONV in SAP this holds the payment conditions but there is no indicator on what the currently active condition applicable is.

So you will see SQL using window functions (PARTITION BY the GROUP BY and ORDERING the list descending and taking the first value) or SQL doing a correlated subquery to find the maximum key per GROUP BY in KONV and then selecting that.

Both are not optimal for HANA and for the Window functions you need a SQLscript type view which makes maintenance more difficult.

See below a way to achieve this in normal SQL however so you have a single pass of the table and use normal aggregation (which is fast in HANA).

```
SELECT "KNUMV",  
       "KPOSN",
```

⁵ More details about the Data Type conversion precedence is available in [SAP HANA Cloud, SAP HANA Database SQL Reference Guide](#)

```

-- pivot action to get the amounts for each condition type in its separate column
MAX(CASE WHEN "KSCHL" = 'ZPAP' THEN "KBETR" END) AS "ZPAP_KBETR",
MAX(CASE WHEN "KSCHL" = 'Z001' THEN "KBETR" END) AS "Z001_KBETR",
MAX(CASE WHEN "KSCHL" = 'Z002' THEN "KBETR" END) AS "Z002_KBETR",
MAX(CASE WHEN "KSCHL" = 'ZPOR' THEN "KBETR" END) AS "ZPOR_KBETR",
MAX(CASE WHEN "KSCHL" = 'ZPOR' THEN "KFAKTOR1" END) AS "ZPOR_KFAKTOR1"
FROM (
    SELECT "KNUMV",
           "KPOSN",
           "KSCHL",
           TO_DECIMAL(SUBSTRING(MAX("ZAEHK" || TO_CHAR("KBETR")),
                                LENGTH(MAX("ZAEHK")) + 1), 11, 2) AS "KBETR", -- KBETR belonging to last counter
           TO_DECIMAL(SUBSTRING(MAX("ZAEHK" || TO_CHAR("KBETR") ||
                                TO_CHAR("KFAKTOR1")), LENGTH(MAX("ZAEHK" || TO_CHAR("KBETR")) + 1), 11, 2) AS
           "KFAKTOR1" -- and belonging KFAKTOR1
    FROM "KONV_View"
    WHERE "MANDT" = '100'
    AND "KSCHL" IN ('ZPAP', 'ZPOR') OR ("KSCHL" IN ('Z001', 'Z002') AND "KINAK" = '')
    GROUP BY "KNUMV",
           "KPOSN",
           "KSCHL" -- I need the latest for each type, so KSCHL goes in the GROUP BY
)
GROUP BY "KNUMV",
"KPOSN"

```

- Minimize joins on multiple columns.

As HANA is a column store database, as long a condition can be executed on that one column, it can be fast. However, when two or more columns are needed this is not as simple as it is in a row store database, where for every row you have the two columns side by side. It needs to evaluate one part of the condition after the other, to be sure the combination is needed or not. Typically, HANA will create an internal table (ITAB) to solve it.

- Minimize cyclic joins.

The column engine does not natively support join cyclic outer joins. If there is such a cycle, the result from a child of the join that completes the cycle is materialized to break the cycle. Cyclic inner join is natively supported, but it is still recommended to use acyclic inner join due to better performance.

If possible, push this type of logic to the ETL layer (obviously only possible when persisting data).

Cyclic join (table B joins to A, C to A and B)

ETL: Look up the column of B needed to join C to A and persist it in A

- Minimize theta joins

Theta join is a join that links tables based on a relationship other than equality between two columns, it is also known as non-equi join. When a non-equality predicate is used in a join, the row engine executes the join operation instead of column engine, after materializing the intermediate results from both join children, which may lead to performance implications.

A mix of equality and non-equality predicates connected by AND is also processed in the same way as non-equi join predicate. Therefore, it is recommended to minimize number of non-equi joins.

- Temporal joins (where column between column A and column B)

One of the frequently used examples of a theta join is a temporal join. It allows you to join the dimensions with the transactional data based on the time validity, for instance where a temporal column from transactional data is between column A and column B of the dimension.

ETL: Generate a new single ID column for each dimension 'version' that will give you the exact dimension data valid on a certain date. Lookup this column (normally transaction date) and persist it in the facts

Workaround: Join only one of the two columns, choose the one that gives you the least matches. This will create a semi-cartesian result. Use a where condition on that result set to keep the one you want

(testing the other column) or a calculated column if the number of columns is small (preventing a query re-write you don't want).

- Period-based calculations

All these different period summations are something you will not want to persist because of the exploding data footprint; at the same time, you want fast performance for the end user. Luckily you can use the cartesian trick to get back the facts records exactly as many times as you need by implementing a new period table that tells you this based on a single column (equi join) and using case when logic based on its columns.

Consider a table format like this:

PERIOD_13	PERIOD_PER	PERIOD_YTD	DIFF
2004001	2003001	N	12
2004001	2003002	N	11
2004001	2003003	N	10
...			
2004001	2003012	N	1
2004001	2004001	2004001	0
2004002	2003002	N	12
2004002	2003003	N	11
2004002	2003004	N	10
....			
2004012	2004011	2004011	1
2004012	2004012	2004012	0

In this table PERIOD_13 is the base period, the one you will select. For each period in PERIOD_13 there are 13 rows, 13 'child' periods (PERIOD_PER). The PERIOD_YTD column will help to easily identify whether a period should be summed for a YTD total. The DIFF column will give you the offset of the 'child' period the period in PERIOD_13. If the DIFF column holds a 0, we are looking at the period itself, and if its value is 12, it is the same period 12 periods, aka a year, ago.

When we now join the fact table period column to the PERIOD_PER column, we can use case logic to determine if a period value needs to be counted in a certain type of rolling sum or not.

Period this year	CASE WHEN DIFF = 0 THEN measure column ELSE 0 END
Same period last year	CASE WHEN DIFF = 12 THEN measure column ELSE 0 END
Rolling x periods	CASE WHEN DIFF < x THEN measure column ELSE 0 END
YTD	CASE WHEN PERIOD_YTD = 'N' THEN 0 ELSE measure column END

In all cases the PERIOD_13 column is used to group the results by.

```
Select PERIOD_13 AS PERIOD,
sum(case when DIFF=0 then VV010 else 0 end) AS VV010_PER
sum(case when DIFF=12 then VV010 else 0 end) AS VV010_PER_LY
sum(case when DIFF=0 then VV010 else 0 end) AS VV010_PER,
sum(case when DIFF=1 then VV010 else 0 end) AS VV010_PER_PREV
sum(case when PERIOD_YTD='N' then 0 else VV010 end) AS VV010_PER,
from "CE1IDEA" c INNER JOIN "PERIODS_MP" p ON (c.PERIO=p.PERIOD_PER)
group by PERIOD_13
```

And even faster, but maybe less readable option is to calculate indicators (1 or 0) which you then can multiply with the period amounts, depending on the performance needed it may be an option. The example for YTD;

```
select PERIOD_13, sum(PER_YTD_IND*VV010) as VV010_YTD, sum(PER_IND*VV010) as VV010_PER
from
(select PERIOD_13, case when PERIOD_YTD='N' then 0 else 1 end as PER_YTD_IND, case when DIFF=0 then 1 else 0
end as PER_IND, sum(VV010) as VV010)
from "CE1IDEA" c INNER JOIN "PERIODS_MP" p ON (c.PERIO=p.PERIOD_PER)
group by PERIOD_13, case when PERIOD_YTD='N' then 0 else 1 end, case when DIFF=0 then 1 else 0 end
```

4.4.5 Reporting Layer Modelling Guideline

The main performance driver here is the volume of data, so key is to minimize the data transfer between Datasphere (database) and the frontend (SAP Analytics Cloud or 3rd party). The most common best practices here are:

- Enforce data aggregation by reducing the dimensions (group by attributes) and using less granular ones
- Use data filters (where conditions, prompts and data access controls), make this (business user) filtering possible to push down.
Think about what makes sense, if a report will normally be used for any other data than the current year, built in this filter.
- Especially when the consuming party is OLAP based (SAP Analytics Cloud), use associations for dimensions, to improve speed of navigation and filtering dimension members

4.4.6 Corporate Memory Layer (Optional)

The basic idea of a corporate memory layer is to preserve the complete history of the loaded data for a long-term persistent storage. This is an optional layer (see Figure 14).

The main purpose for such a layer is for untransformed historization of the source data. That way it can be used as a source for reconstructions, without the need to access the original sources again. This is especially helpful when source system data is deleted or archived, or in cases when source systems are retired. The data of a corporate memory layer is often kept on cheaper storage systems like SAP HANA Data Lake and the likes.

4.4.7 Outbound Data Layer (Optional)

The optional outbound data layer (see Figure 14) provides a temporary storage area for data which can be accessed directly using both internal and 3rd party tools or being pushed to an external destination. It would usually be placed above the reporting layer but could sit anywhere based on the requirements.

Key Principles:

- Used as the data storage layer to provision data from Datasphere to requesting target systems
- Data can be deleted from this layer as required when no longer needed to supply the target system
- You can timestamp records that were replicated to external systems, as this makes data reconciliation a whole lot easier

4.4.8 Modelling KPIs, Restricted and Calculated Key Figures

The calculation of key performance indicators (KPIs) and other derived measures should be done in SAP Datasphere wherever possible and not in SAP Analytics Cloud.

This allows for common definitions in one place (single source of truth) that may be re-used in multiple stories.

With Analytic model, Restricted measures, Calculated Measures now becomes part of Data Builder. It is no longer required to create Restricted key figures to be modeled in the Business Builder only. Alternatively, you could model restricted key figures in SAP Analytics Cloud.

Besides, Analytic Model provides additional capability of using Exception Aggregation inside Restricted and Calculated measures. Also, currency conversion can be done on aggregated data, variables, filters can be applied as well. Constant selection can also be enabled for the Restricted Measure, where a measure with constant selection will not be impacted by filters or drill-downs on the constant dimensions. More on this can be found at -

https://help.sap.com/docs/SAP_DATASPHERE/c8a54ee704e94e15926551293243fd1d/e5fbe9e2cb93484da b8b1963145e565f.html

4.4.9 Master Data and Hierarchies

The tables and views for transactional data should not redundantly hold master data or hierarchy information, especially when consumed in an OLAP fashion by SAP Analytics Cloud. There will be performance improvements both for the core model, as well as the navigation through the hierarchies and dimensions, as these can be done on a much smaller dataset.

Instead, master data (including language dependent texts) and hierarchies should be modelled independently and only once. To allow the use of them in an Analytic Model, they should be of the type dimension (also a pre-requisite for defining hierarchies).

Use associations to add them star-schema-like to the Fact view or analytical dataset view. For a new model, always use the new Analytic Model and a Fact type view as source. The (deprecated) Analytical Dataset can only expose one level of dimensions, the ones directly associated in the view itself. In the modeler for the Analytical Model, you can navigate dimensions associated with this first level and add them as well. You expose only dimensions you need for the set of stories for which you build the Analytic Model.

4.4.10 Federation vs. Persistency

Persistency in SAP Datasphere is possible on various layers. For remote tables, the inbound layer offers the option for real-time replication or (scheduled) single snapshots. Data acquired via the data flow, replication flow or other ETL-tools is always persisted. In these cases, you can build up data historization or also a corporate memory layer.

Additionally, the Data Builder offers view persistence for target views. These can make sense in the propagation layer top provide reporting persistence, especially where heavy joins, sequential calculations or formula filters are applied.

Since there are quite a few options to retrieve data, the best way to get an overview is by listing the possibilities and for what they are good.

Method	Virtual access or persisted	Target table	Apply filter	General limitation
Remote table	Virtual	Internal table during runtime of query	Yes, via table editor	Internal tables in HANA are generally limited by memory and/or 2 billion (4 byte integer) maximum number of rows
Remote table	Snapshot	Generates table of full width or with selected columns only	Yes, via table editor	The loading process can consume more memory than expected before HANA executes first compression. Partitioning settings in loading process can limit resource consumption for large source tables.
Remote table	Real-time replication	Generates table of full width or with selected columns only	Yes, via table editor	The initial load has the same restrictions as the snapshot. Subsequent delta loads are real-time and trigger based, depending on the number of tables to be replicated, the number of connections to the agent and the source system should be properly configured.
Views (analytic and relational)	Virtual, if no underlying objects are persisted	Internal table during runtime of query	Yes, via underlying table editor or in view editing	A view created on virtual models underlies the memory quotas and internal table limitations as all internal tables. The result set size during specific operation steps can supersede the end result, depending on the model.
Views (analytic and relational)	Snapshot	Target table with view column width	Yes, via underlying table editor or in view editing	

When you decide for federated access, be aware on the impact on the source system(s) as well as your SAP Datasphere tenant. Figure 17 shows different scenarios.

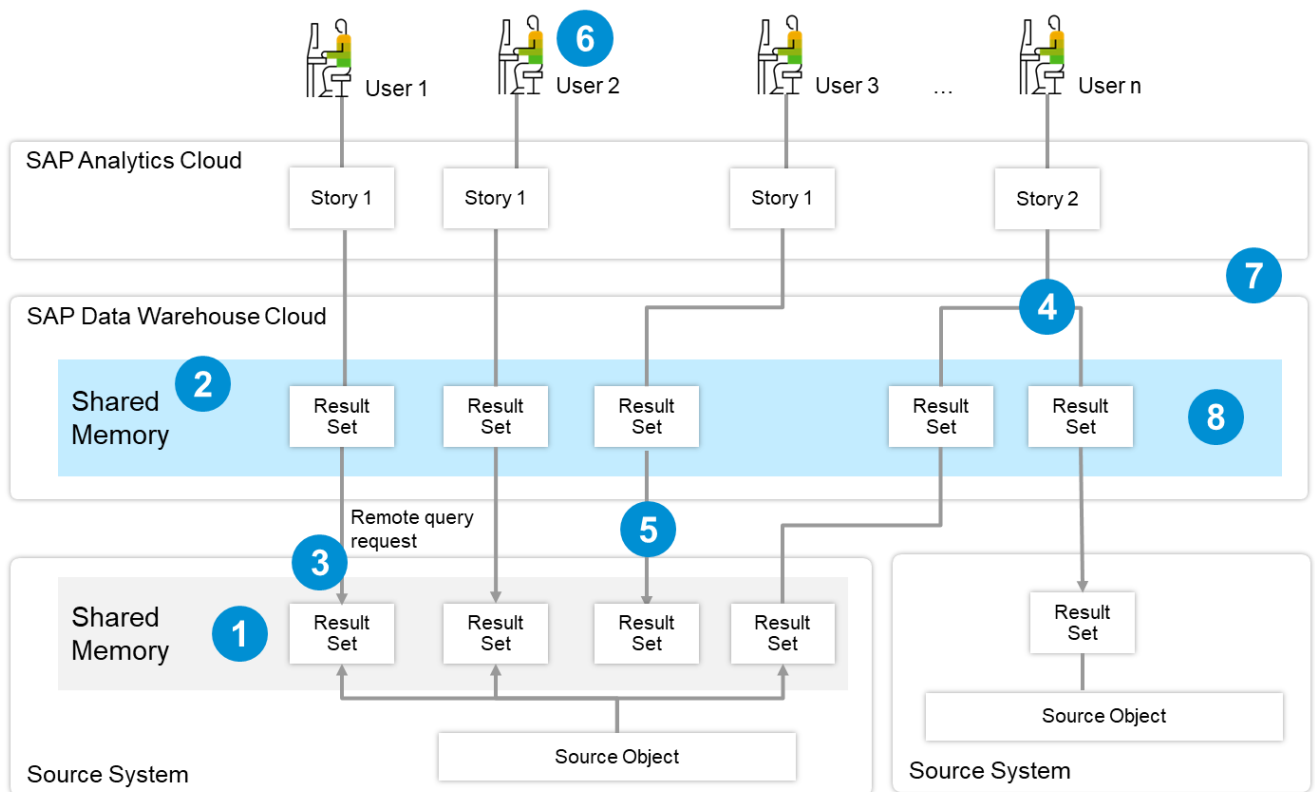


Figure 17: Query execution on virtual sources

1. Federated access will pass through all requests to the source and stress the source system
2. Datasphere needs to hold the result sets in federation separately. A persisted result set can act as cache, but would break federation
3. Connections have defined capabilities, not available capabilities will be taken over by Datasphere, but can lead to more data being selected (see also 0

4. Limitation in Remote Source Capabilities)
5. Cross system models have additional challenges, that logic can require to load full data every time
6. Federation will lead to higher network load, that needs to be considered and can create cost
7. Consider parallel loads from multiple user and stories consuming multiple data sets
8. In federation you always need to consider the E2E flow of data from source object, through the connector, the network, the processing in Datasphere (Views, etc.), through the MDS, into the SAC story and the navigation and aggregation from the client
9. Additional Datasphere featured like data access control or Intelligent Lookup have to be considered. Data access controls can require loading more granular data

Persisting some data sets in SAP Datasphere can add value, specifically when it comes to load on your source system(s) as well as Datasphere.

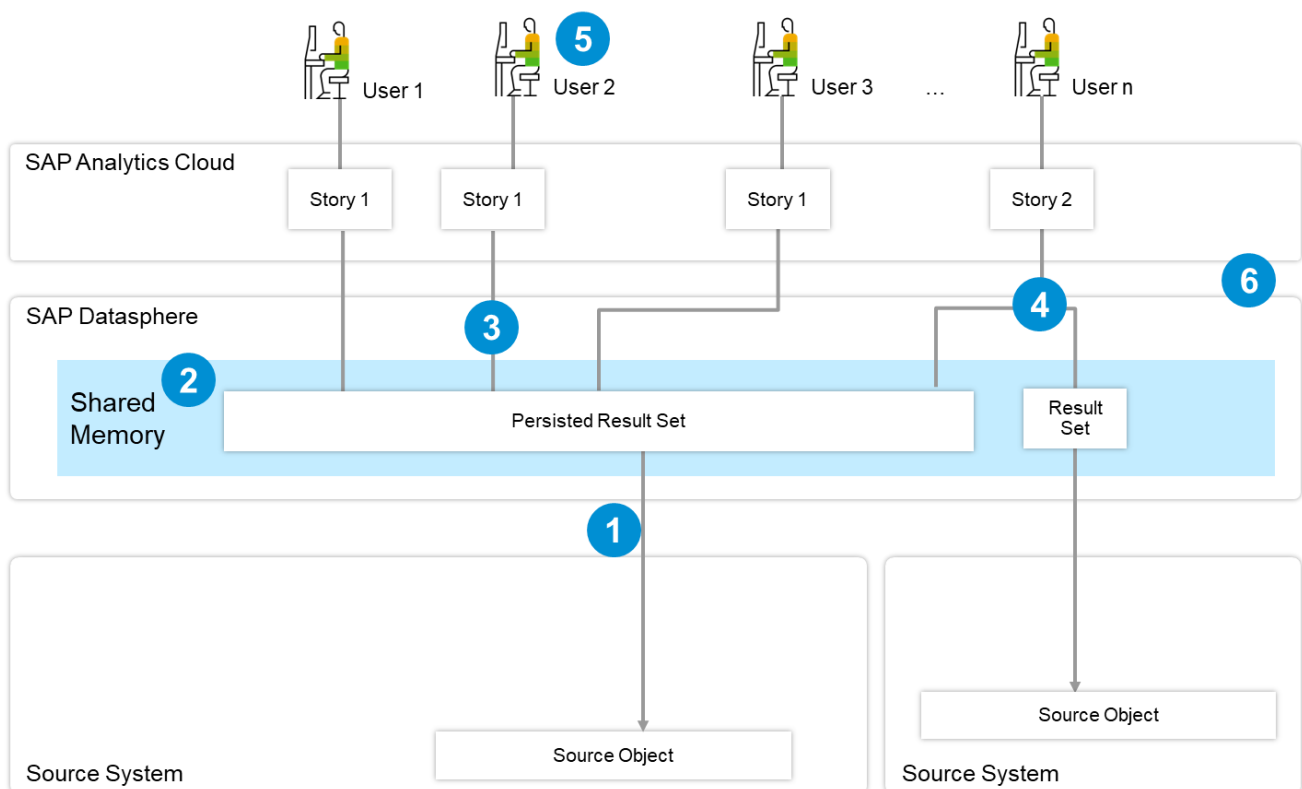


Figure 18: Query execution with persisted and virtual sources

1. Data is only transferred once
2. Result sets can be cached
3. HANA can provide all SQL and MDS capabilities to the data
4. Cross system modeling is replaced by local processing, therefore everything can be executed locally
5. Parallel loads will only impact Datasphere
6. The E2E flow of Data is a lot shorter

SAP Datasphere offers multiple options for data replication with remote table replication, data flows and replication flows. The comparison in Figure 19 and the following characteristics table helps to better

understand the differences and similarities with regards to Datasphere artifacts, the connectivity components and the characteristics of each option.

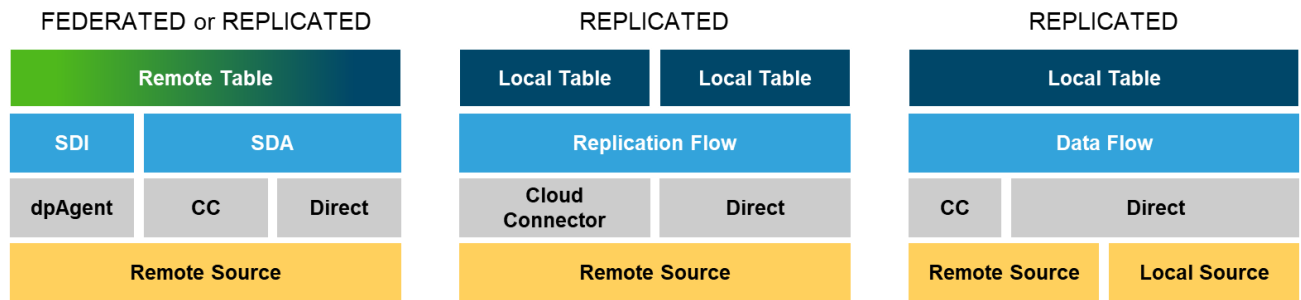


Figure 19: Federation and Replication options

Characteristics	Remote Table	Replication Flow	Data Flow
Source Entities	Single source entity	Multiple entities from one source	Multiple sources to single target entity
Transformation	Row Filter, Only column filter	Row Filter, Projection	Row Filter, Full Transformation capabilities
Partitioning	Partitioning	No Partitioning	No Partitioning
Replication Type	Real-Time replication, snapshot replication, transparent switch between replication and federation	Batch Delta replication, Snapshot replication, replication only	Pseudo Delta, substitution parameter, replication only
Others		Resilient replication, pick up replication after interruption	

5 Naming Conventions

5.1 General Remarks

A naming convention can help to identify and separate the objects in SAP Datasphere or describe their purpose. The technical names are technically unique per space. In case it is required to have a harmonized naming across spaces, this will require some governance next to the system capabilities.

This chapter will give some guidance, how to organize and thus name the objects. This should also help all users to find the specific entity they search with in the tenant and within the spaces they are assigned to.

Another main aspect of naming conventions is an evolution of the model over time. People and responsibilities may change over time. Having a naming convention and adhering to that convention helps developers and users in the system, through having a purpose, an area, an origin or even the priority of a model as part of a naming convention. Anyone would as such be able to quickly understand an object and its purpose, if it can be reused or if it is allowed to be altered.



Avoid the Prefix “SAP” for any object, as SAP delivered business content is using this prefix.

SAP Datasphere does not provide the functionality to define namespaces.
Check the [Rules for Technical Names](#) on the product documentation.

General recommendations for naming:

- Unambiguous: Use clear, well-defined descriptions, if in doubt, talk to your project's information developer and check SAP terminology.
Goal: Users understand immediately what the entity is used for
- For descriptions and business names use blanks. Do not use underscores or CamelCase words.
- Avoid abbreviations as much as possible

5.2 Space Names

Spaces provide an independent virtual work environment for departments, LOB's, data domains, project teams, and individuals.

When creating spaces, please be aware that the **Space ID** can contain a maximum of 20 characters and cannot be changed later. The valid characters for the Space ID are: A - Z, 0 - 9, and _

The **Space Name** contains a max. of 30 characters and can be changed any time.



Reserved keywords, such as SYS, PUBLIC, CREATE, SYSTEM, DBADMIN, PAL_STEM_TFIDF, SAP_PA_APL, Datasphere_USER_OWNER, Datasphere_TENANT_OWNER, Datasphere_AUDIT_READER, Datasphere_GLOBAL, and Datasphere_GLOBAL_LOG, must not be used. Unless advised to do so, the ID must not contain prefix _SYS and should not contain prefixes: Datasphere_, SAP_. Also, the keywords that are reserved for the SAP HANA database cannot be used in a space ID (see [Reserved Words](#) in the *SAP HANA SQL Reference Guide for SAP HANA Platform*).

5.3 Connections

We recommend using a similar name also for connections.

Please be aware that technical names of connections are case-sensitive, allow a maximum of 40 characters and cannot be changed later (valid characters are: A - Z, a - z, 0 - 9, and _).

For example:

- All SAP source system connections to start with “SAP_”
- Specify the backend type or application type using abbreviations:
e.g. S4H for S/4HANA systems or B4H for BW/4HANA systems
- This part may also contain underscores if necessary.
- Optional: Underscore and then free text in camel case to specify more details, e.g. indicate the type of data that can be accessed or the business area. The free text may contain additional underscores.

Hence, connection names may have the following structure:

SAP_<backend system type or application>[_<free text in camel case>]

Examples:

SAP_B4H

SAP_S4H

SAP_S4H_CostCenter

In case you have multiple systems of the same type, e.g. multiple SAP S/4HANA systems, we recommend you add the unique SAP system identifier (SID) to the connection name: SAP_S4H_P07. Similar for cases where you might have multiple clients in one source system to use the client number as additional identifier.

Please be aware that the SAP delivered content for Datasphere (which will import into the space SAP_CONTENT) contains the following source system connections:

Technical Name	Business Description	Connection Type
SAP_AUT	SAP Automotive S/4HANA	SAP ABAP
SAP_CP	SAP Consumer Products S/4HANA	SAP ABAP
SAP ISU	SAP Utilities S/4HANA	SAP ABAP
SAP_OBBI_Insights	SAP OBBI Insights	oDATA
SAP_OBBI_S4H	SAP S/4HANA On Premise (OBBI)	SAP S/4HANA On-Premise
SAP_S4H	SAP S/4HANA On Premise	SAP S/4HANA On-Premise
SAP_S4HC	SAP S/4HANA Cloud	Cloud Data Integration
SAP_SFSF	SuccessFactors	SAP SuccessFactors for Analytical Dashboards

These names may be used, as you most probably would want to use one connection per source system. In case you want to use the SAP delivered content, or parts of it, next to your own content, you would have less efforts to combine your content with SAP content, as the connection(s) would already be “aligned”.

SAP Datasphere also provides a functionality to easily change from one system to another in a view, in case an alignment to one connection is required, the above table should help you understand what is currently being used by SAP within their content packages.

5.4 Database Users and Open SQL Schema

We recommend that you name the database users for Open SQL schemas in a similar way as the connections, especially if they are used for write access and SQL modelling. For read only access this is less relevant, even though understandable names are always recommended.

- All SAP related open SQL schemas to start with “SAP_” or the application name (e.g ARIBA)
- Specify the backend type or application type using abbreviations:
e.g. S4H for S/4HANA systems or B4H for BW/4HANA systems
- This part may also contain underscores if necessary.
- Optional: Underscore and then free text in camel case to specify more details, e.g. indicate the type of data that can be accessed or the business area. The free text may contain additional underscores.

As the Space Name is always part of the Schema Name, “<SPACE_NAME>#” is already given and the schema name follows the #. So, all open SQL schema names have the following structure:

<SPACE_NAME>#<SAP_<application>[_<free text in camel case>]

5.5 Folders

Folders can be used to group and organize objects. They are currently only available in the Business Builder. We recommend to group artefacts in a meaningful way e. g. master data or subject oriented in case you have a finance space with accounting and controlling data for instance.

5.6 Modeling Objects

The technical names for modelling objects in the data builder are case-sensitive, limited to 50 characters and cannot be changed later (valid characters are: A - Z, a - z, 0 - 9, and _).

It may make sense to use the technical names to apply naming conventions, especially in a space with strong central governance and heavy reuse of objects by other spaces.

In the data builder **tags** may be used to group and find objects. The business purpose section of the objects (see Figure 20) allows you to apply tags to an object. Subsequently, the repository explorer and the data builder landing page allow you to filter on these tags.

Business Purpose

Description:

Sales Orders View for consumption in analytics tools. It contains all sales order attributes from header and item tables.

Purpose:

Sales Orders Consumption View

Business Contact Person:

John Smith

Responsible Team:

Sales

Tags:

Consumption X Sales X Sales Orders X

Figure 20: Use tags in modeling objects

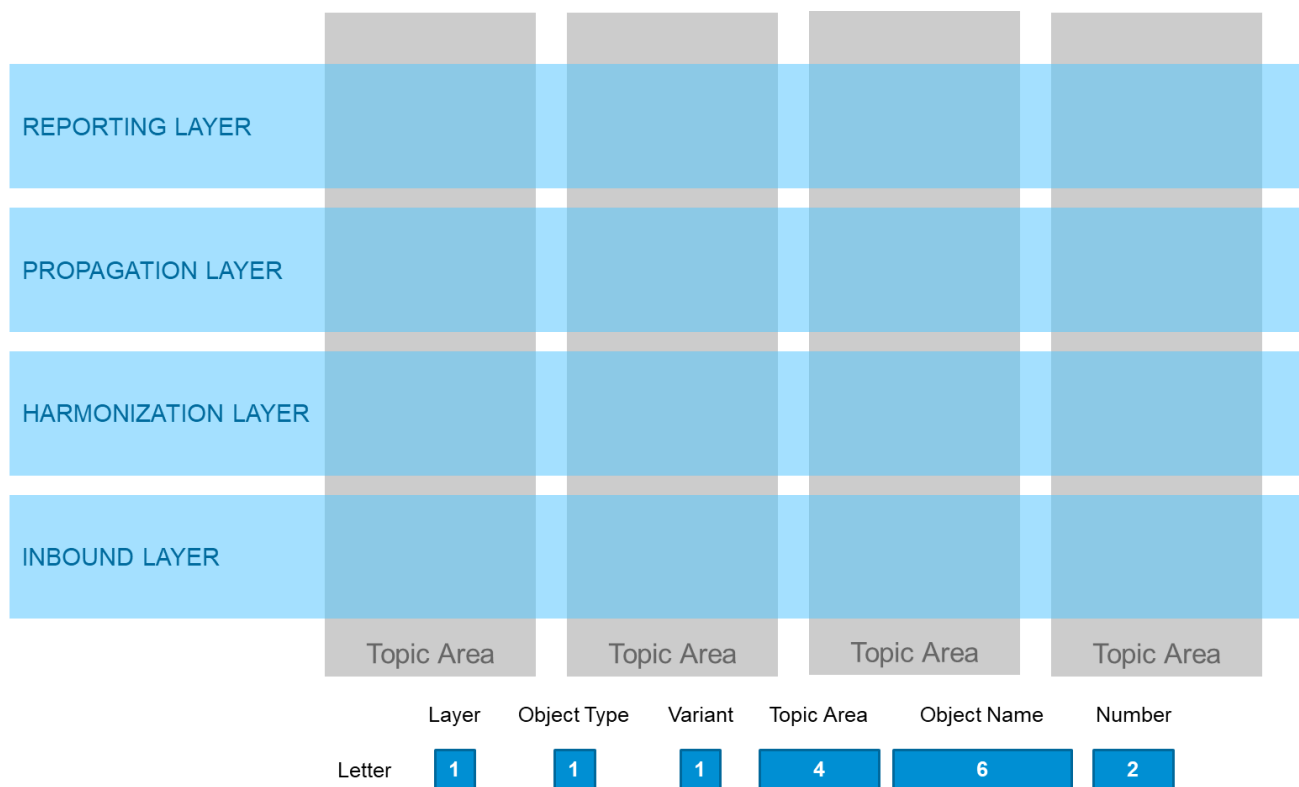


Figure 21: Naming convention overview

A common modelling approach is using stacked models with various layers (see 4.4.2 Layered Modeling Approach). Hence, it would make sense to identify the layer the object belongs to already from the name of the artifact. The same applies for the object type and as each table or view may have different variants, it might make sense in some cases to also reflect the variant in the name.

Please consider that these table or view types can be changed at a later point in time. The technical names cannot be changed later, only a replacement of a table/view in a model would be possible.

In case you have multiple topic areas in one space e. g. cost center or purchasing orders it may be worth including the topic area in the name. If you use individual spaces for your topic areas, then this might only be useful if you share lots of artefacts via the cross-space sharing feature.

The object name should describe the object itself. You could also spend more characters for object names.

At the end a two-digit number makes sense, as there might be multiple versions or similar artefacts for different purposes. You could give an explanation about the versions in the description or the business purpose description optionally.

Examples:

1TR_FINA_COSTCT_01 - for a relational table for finance cost centers

3VA_PURC_PURORD_01 - for an analytical view for purchasing orders



At the end it is your project decision to introduce naming conventions for all or may be just centrally governed space(s). You might only pick the parts which fit your use case like layer, object name and number, also using different lengths in another sequence. It is more important that you stick to your naming conventions in your tenant (or the relevant spaces), as the example described above is just one way to apply naming conventions to modeling objects.

6 Performance Best Practices

6.1 General Considerations

In the following chapter should provide an overview of use cases, which have lead to bad performance or unwanted results. It should also provide an overview of situations to avoid and how to avoid them.

6.2 Performance for Virtual Access

6.2.1 Transfer of Large Amounts of Data

The following model joins a local table with a remote source using an ABAP connection

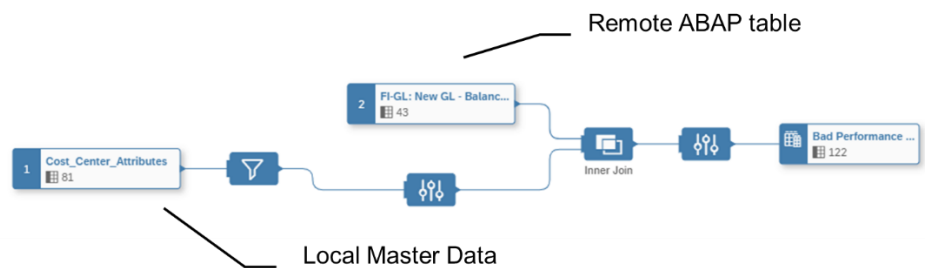


Figure 22 Model Example for ABAP Data Transfer

Simply executing a preview takes a long time and does not reflect selecting top 1000 from this result set. A long runtime against a model including a remote source leaves to believe, the query against the remote table is taking too long. For a top down analysis, a developer can open the monitor and see remote queries. As in this case:

The screenshot shows the SAP Data Integration Monitor interface with the 'Remote Query Monitor' tab selected. It displays a table of remote queries.

Connection	Start Time	Statement Runtime	Rows	Status	Remote SQL Statement
BW4 Connection - CoE ...	Jul 26, 2022 17:31		1,369,088	EXECUTING	SELECT "IMOD_CCA10\$VT"."CO_AREA", "IMOD_CCA10\$VT"."CO_DOC_NO", "IMOD_CCA10\$VT"."CO_ITEM_NNR", "IMOD_CCA10\$VT":... More

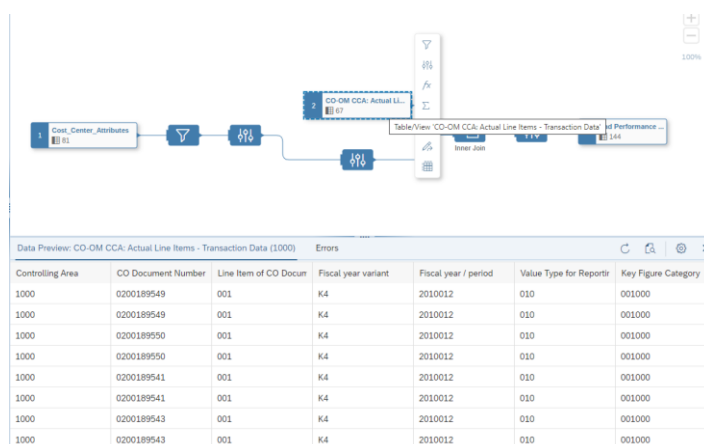
Figure 23: Remote Query Monitor ABAP Connection

The result set fetched is already ~1.3 mio rows, not reflecting the top 1000 selection. In the details of the remote SQL statement, the impression solidified:

```
SELECT
    "IMOD_CCA10$VT"."CO_AREA",
    "IMOD_CCA10$VT"."CO_DOC_NO",
    "IMOD_CCA10$VT"."CO_ITEM_Nr",
    "IMOD_CCA10$VT"."FISCVARNT",
    "IMOD_CCA10$VT"."FISCPER",
    "IMOD_CCA10$VT"."VTYPE",
    "IMOD_CCA10$VT"."METYPE",
    "IMOD_CCA10$VT"."CURTYPE",
    "IMOD_CCA10$VT"."RECORDMODE",
    "IMOD_CCA10$VT"."COSTCENTER",
    "IMOD_CCA10$VT"."ACTTYPE",
    "IMOD_CCA10$VT"."VTDETAIL",
    "IMOD_CCA10$VT"."VTSTAT",
    "IMOD_CCA10$VT"."VERSION",
    "IMOD_CCA10$VT"."VALUATION",
    "IMOD_CCA10$VT"."CORRTYPE",
    "IMOD_CCA10$VT"."COSTELMNT",
    "IMOD_CCA10$VT"."DB_CR_IND",
    "IMOD_CCA10$VT"."PIOBSV",
    "IMOD_CCA10$VT"."PIOVALUE",
    "IMOD_CCA10$VT"."PART_CCTR"
FROM "BW./IMO/D_CCA10$F" "IMOD_CCA10$VT"
```

Figure 24: Remote table statement without restriction

The query is getting all records. In comparison, just previewing on top of the remote source runs in 3 seconds and the statement reflects a top 1000 selection.



The screenshot shows the SAP BW Data Preview tool interface. At the top, there is a query execution plan diagram with nodes for 'Cost_Center_Attributes', 'CO-OM CCA: Actual Li...', and 'Table View: CO-OM CCA: Actual Line Items - Transaction Data'. Below the diagram, a 'Data Preview' window displays a table with 1000 rows. The table has the following columns: Controlling Area, CO Document Number, Line Item of CO Docum, Fiscal year variant, Fiscal year / period, Value Type for Reportir, and Key Figure Category.

Controlling Area	CO Document Number	Line Item of CO Docum	Fiscal year variant	Fiscal year / period	Value Type for Reportir	Key Figure Category
1000	0200189549	001	K4	2010012	010	001000
1000	0200189549	001	K4	2010012	010	001000
1000	0200189550	001	K4	2010012	010	001000
1000	0200189550	001	K4	2010012	010	001000
1000	0200189541	001	K4	2010012	010	001000
1000	0200189541	001	K4	2010012	010	001000
1000	0200189543	001	K4	2010012	010	001000

Figure 25: Data preview of sub-query

Connection	Start Time	Statement Runtime	Rows	Status	Remote SQL Statement
BW4 Connection - CoE ...	Jul 26, 2022 17:42	3 Seconds	1,000	CLOSED	SELECT "IMOD_CCA10\$VT"."CO_AREA", "IMOD_CCA10\$VT"."CO_DOC_NO", "IMOD_CCA10\$VT"."CO_ITEM_N...", "IMOD_CCA10\$VT"... More

Figure 26: Remote table monitor

```

SELECT

    "IMOD_CCA10$VT"."CO_AREA",
    "IMOD_CCA10$VT"."CO_DOC_NO",
    "IMOD_CCA10$VT"."CO_ITEM_N...",
    "IMOD_CCA10$VT"."FISCVARNT",
    "IMOD_CCA10$VT"."FISCPER",
    "IMOD_CCA10$VT"."VTYPE",
    "IMOD_CCA10$VT"."METYPE",
    "IMOD_CCA10$VT"."CURTYPE",
    "IMOD_CCA10$VT"."RECORDMODE",
    "IMOD_CCA10$VT"."COSTCENTER",
    "IMOD_CCA10$VT"."ACTTYPE",
    "IMOD_CCA10$VT"."VTDETAIL",
    "IMOD_CCA10$VT"."VTSTAT",
    "IMOD_CCA10$VT"."VERSION",
    "IMOD_CCA10$VT"."VALUATION",
    "IMOD_CCA10$VT"."CORRTYPE",
    "IMOD_CCA10$VT"."COSTELMNT",
    "IMOD_CCA10$VT"."DB_CR_IND",
    "IMOD_CCA10$VT"."PIOBJSV",
    "IMOD_CCA10$VT"."PIOVALUE"

FROM "BW./IMO/D_CCA10$F" "IMOD_CCA10$VT" LIMIT 1000

```

Figure 27: Remote table statement (with limit 1000)

The limit 1000 is applied to the view on the very top and the HANA Cloud will try to push down this limit to the relevant sources. In this case it will not happen. HANA neither knows any cardinality of the source object and if the limit 1000 would reduce the result set significantly. Nor does HANA know if the join could alter the sequence of the result and therefore alter the limit 1000 result considerably. This means HANA will not change the sequence of execution (now being join first, limit 1000 second). Last but not least the join cannot be pushed to the source system, since the ABAP connection does not support join push-down. Therefore, HANA will get all data, then join, then apply the limit 1000.

This is a very simple scenario, but it shows small models can pose a great deal of problems. Knowing this, there are several ways to make the result better.

6.2.2 Option 1: ABAP Layer OLAP Access

If the ABAP Layer is the only way to access data, this is a boundary condition to work with. It limits the options but could be a necessity.

The main driver for the huge result set is the virtual model. The source object is called “CCA: Actuals Line-Items”. This should already hint to the conclusion: very granular data, very bad when not compressed.

To look at the sizes, it is possible in HANA based systems to get sizing statistics of the tables involved. In this case the accessed table of the remote BW system is mainly the /B1H/AD_CCA102, the active table of Cost Center Accounting business content ADSO from a BW/4HANA system. In the monitoring view of the source M_CS_COLUMNS, it is possible to find every cardinality and size of the columns that are selected.

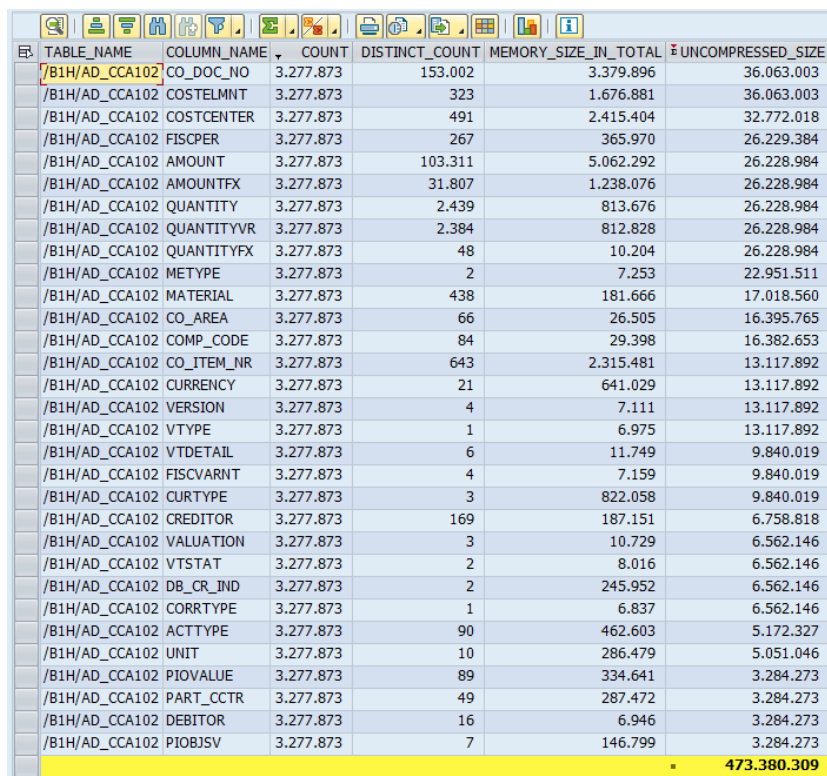
Memory Size in Total = Amount of memory used by the column with HANA compression

Uncompressed Size = Amount of memory used during data transfer (virtual data transfer decompresses the result set)

Count = number of records in total of the table

Distinct_Count = the distinct number of entries

These figures give an idea of the data transferred during the selection. In our example this amounts to:



TABLE_NAME	COLUMN_NAME	COUNT	DISTINCT_COUNT	MEMORY_SIZE_IN_TOTAL	UNCOMPRESSED_SIZE
/B1H/AD_CCA102	CO_DOC_NO	3.277.873	153.002	3.379.896	36.063.003
/B1H/AD_CCA102	COSTELMNT	3.277.873	323	1.676.881	36.063.003
/B1H/AD_CCA102	COSTCENTER	3.277.873	491	2.415.404	32.772.018
/B1H/AD_CCA102	FISCPER	3.277.873	267	365.970	26.229.384
/B1H/AD_CCA102	AMOUNT	3.277.873	103.311	5.062.292	26.228.984
/B1H/AD_CCA102	AMOUNTFX	3.277.873	31.807	1.238.076	26.228.984
/B1H/AD_CCA102	QUANTITY	3.277.873	2.439	813.676	26.228.984
/B1H/AD_CCA102	QUANTITYVR	3.277.873	2.384	812.828	26.228.984
/B1H/AD_CCA102	QUANTITYFX	3.277.873	48	10.204	26.228.984
/B1H/AD_CCA102	METYPE	3.277.873	2	7.253	22.951.511
/B1H/AD_CCA102	MATERIAL	3.277.873	438	181.666	17.018.560
/B1H/AD_CCA102	CO_AREA	3.277.873	66	26.505	16.395.765
/B1H/AD_CCA102	COMP_CODE	3.277.873	84	29.398	16.382.653
/B1H/AD_CCA102	CO_ITEM_NR	3.277.873	643	2.315.481	13.117.892
/B1H/AD_CCA102	CURRENCY	3.277.873	21	641.029	13.117.892
/B1H/AD_CCA102	VERSION	3.277.873	4	7.111	13.117.892
/B1H/AD_CCA102	VTTYPE	3.277.873	1	6.975	13.117.892
/B1H/AD_CCA102	VTDETAIL	3.277.873	6	11.749	9.840.019
/B1H/AD_CCA102	FISCVARNT	3.277.873	4	7.159	9.840.019
/B1H/AD_CCA102	CURTYPE	3.277.873	3	822.058	9.840.019
/B1H/AD_CCA102	CREDITOR	3.277.873	169	187.151	6.758.818
/B1H/AD_CCA102	VALUATION	3.277.873	3	10.729	6.562.146
/B1H/AD_CCA102	VTSTAT	3.277.873	2	8.016	6.562.146
/B1H/AD_CCA102	DB_CR_IND	3.277.873	2	245.952	6.562.146
/B1H/AD_CCA102	CORRTYPE	3.277.873	1	6.837	6.562.146
/B1H/AD_CCA102	ACTTYPE	3.277.873	90	462.603	5.172.327
/B1H/AD_CCA102	UNIT	3.277.873	10	286.479	5.051.046
/B1H/AD_CCA102	PIOVALUE	3.277.873	89	334.641	3.284.273
/B1H/AD_CCA102	PART_CCTR	3.277.873	49	287.472	3.284.273
/B1H/AD_CCA102	DEBITOR	3.277.873	16	6.946	3.284.273
/B1H/AD_CCA102	PIOBSV	3.277.873	7	146.799	3.284.273
					473.380.309

Figure 28: Source column granularity and memory consumption

The select statement getting all data will in the end transfer ~450MB of data via the network, which should be avoided. The main space drivers are the columns with the highest distinct count, and column width. In this case document number, cost element and cost center. Since the join happens on Cost Center in the model, this column cannot be excluded, but the other two are not necessary. The measures will necessarily need to be aggregated to reduce the size.

Additionally, the ABAP layer still makes a strong distinction between processing for reporting and extraction. The federation case mainly needs reporting processing. Accessing a Provider like an ADSO will trigger extraction processing. This will want to guarantee the granularity of the data and the sequence (the operation in the target could be different to simple aggregation). When targeting a query an additional possibility is to aggregate data in the source. To compress the amount of data sent, a query aggregates data before transferring.

Side note: using a query instead of an ADSO will limit the options in replication. There is no delta, therefore replication is restricted to snapshots or view persistency.

In the source system, a query is used instead of the InfoProvider. The query is flat and has all characteristics as free characteristics and is flagged as “Query is used as InfoProvider”.

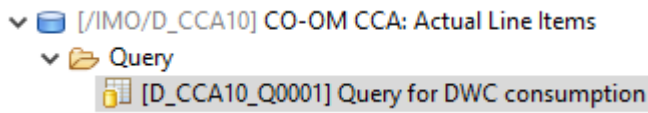


Figure 29: Defined Query-Provider for better ABAP consumption

In Datasphere the model is built and only the relevant columns are used, therefore a projection reduces all columns like document and line items:

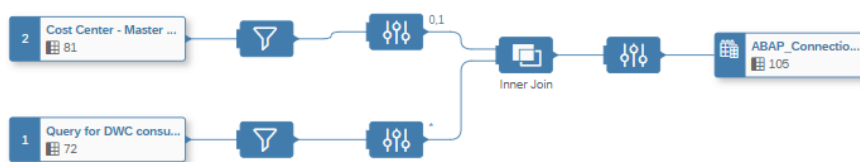


Figure 30: Model consuming Query-Provider

Instead of a long runtime in the remote query monitor, because all data is extracted in full granularity, only the necessary data is retrieved and is aggregated before transfer:

Remote Table Monitor View Persistency Monitor Data Flow Monitor Remote Query Monitor Task Chain Monitor					
Remote Queries (16)					
Connection	Start Time	Statement Runtime	Rows	Status	Remote SQL Statement
BW4 Connection - CoE ...	Aug 4, 2022 11:14	2 Seconds	1,412	CLOSED	SELECT "D_CCA10_Q0001\$VT"."BUS_AREA", "D_CCA10_Q0001\$VT"."COMP_CODE", "D_CCA10_Q0001\$VT"."CORRTYPE", "D_CCA10... More

Figure 31: Remote statement on Query-Provider

Statement also shows, that Top 1000 is not pushed down, because of the join:

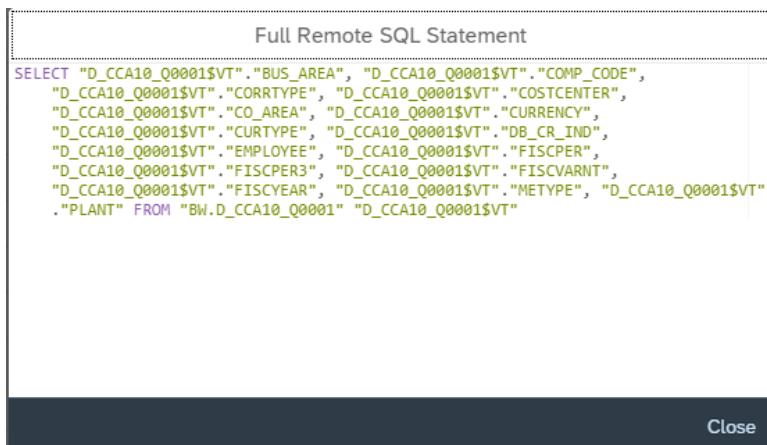


Figure 32: Select statement

Difference of the extraction process in BW ODQMON:

Query with projection:

Composite Request	C.	S.	Subscription	RT	Units	Rows	Original Size in Bytes	Compressed Size in Bytes	Comp. %	Lower Limit for TSN	Upper Limit for TSN	Extractions Request	Ext Extraction Mode	Background Job	Selection
20220804110154	✓	HANA_SDI	SAPDS001	1	1,412	1,412	1,753,704	11,671	99,3			(2022-08-04 11:01:...	Data Snapshot	ODQR_20220804_090154_000000_F	
20220804110233	✓	HANA_SDI	SAPDS001	1	1,425	1,425	1,769,850	16,730	99,1			(2022-08-04 11:02:...	Data Snapshot	ODQR_20220804_090233_000000_F	
20220804111135	✓	HANA_SDI	SAPDS001	1	1,412	1,412	1,753,704	11,671	99,3			(2022-08-04 11:11:...	Data Snapshot	ODQR_20220804_091135_000000_F	
20220804111412	✓	HANA_SDI	SAPDS001	1	1,412	1,412	1,753,704	11,671	99,3			(2022-08-04 11:14:...	Data Snapshot	ODQR_20220804_091412_000000_F	
					4	5,661	7,030,962	51,743							

Figure 33: Source system transfer of Query-Provider with data volume

InfoProvider:

Composite Request	C.	S.	Subscription	RT	Units	Rows	Original Size in Bytes	Compressed Size in Bytes	Comp. %	Lower Limit for TSN	Upper Limit for TSN	Extractions Request	Ext Extraction Mode	Background Job	Selection
20220726173101	✓	SAPDS001	SAPDS001	3,721	3,277,873	3,277,873	3,900,668,870	21,699,055	99,4			(2022-07-26 17:31:...	Data Snapshot	ODQR_20220726_153101_000001_F	
20220726173407	✓	SAPDS001	SAPDS001	3,721	3,277,873	3,277,873	3,900,668,870	21,699,055	99,4			(2022-07-26 17:34:...	Data Snapshot	ODQR_20220726_153407_000101_F	
20220726173754	✓	SAPDS001	SAPDS001	3,721	3,277,873	3,277,873	3,900,668,870	24,864,999	99,4			(2022-07-26 17:37:...	Data Snapshot	ODQR_20220726_153754_000182_F	
20220726174238	✓	SAPDS001	SAPDS001	3,721	3,277,873	3,277,873	3,900,668,870	24,864,722	99,4			(2022-07-26 17:42:...	Data Snapshot	ODQR_20220726_154239_000000_F	

Figure 34: Source system transfer of InfoProvider with data volume

Instead of 3.9 GB only 1.7 MB are transferred, which is more suitable for federation.

6.2.3 Option 2: Database Layer Access

If the ABAP layer is not necessary, a connection HANA to HANA is always favorable for the reason of database functionality push down.

Using the external HANA view, a calculation view of the ADSO is generated. This can be consumed via a HANA connection to the underlying Database of the BW/4HANA or BW on HANA system.

General: /IMO/D_CCA10

DataStore Object (advanced)

Technical Name: /IMO/D_CCA10

Description: CO-OM CCA: Actual Line Items

☒ External SAP HANA View

☐ Log Read Access Output

Figure 35: Exposing BW objects via Calculation Views in HANA

This HANA view can be used in the Datasphere model:

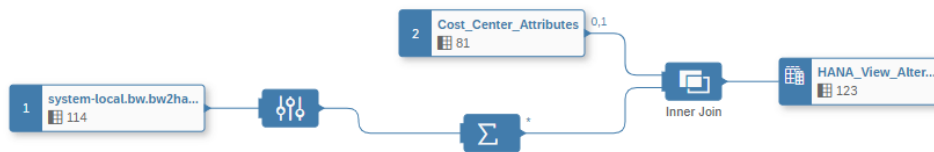


Figure 36: Datasphere Model with HANA Calculation View

In this case it is even possible to replicate statistics to help the execution:

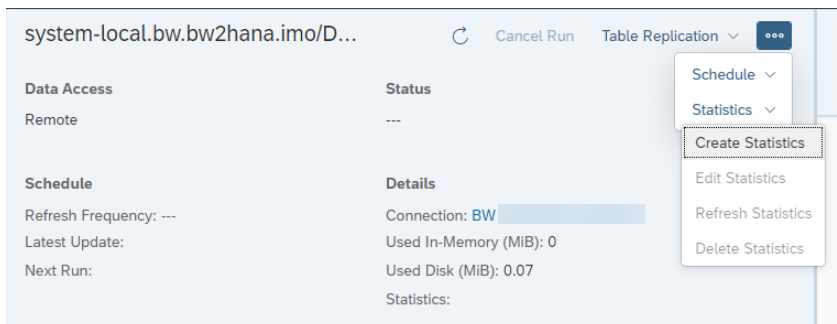


Figure 37: Calculation Views allow creation of statistics

The HANA connection, even with bigger result sets is more forgiving. As an example, an overall result set of more than 1 mio rows is relatively fast:

Remote Table Monitor View Persistence Monitor Data Flow Monitor Remote Query Monitor Task Chain Monitor					
Remote Queries (21)					
Connection	Start Time	Statement Runtime	Rows	Status	Remote SQL Statement
BW CoE HD2 connection	Aug 4, 2022 14:03	12 Seconds	1,174,077	CLOSED	SELECT "systemlocal_bw_bw2hana_imoD_CCA10\$VT"."OACTTYPE", "systemlocal_bw_bw2hana_imoD_CCA10\$VT"."OAUACCTTYPE", "... More

Figure 38: Remote query with execution time

In addition to this, other operations could be pushed down to the remote source to limit the transfer of data, such as aggregations, joins or other database operations.

6.3 View Modelling

6.3.1 Monolithic Modelling

The architecture proposed in the chapter 4.4.2 Layered Modeling Approach not only helps organize the system for many developers. It helps during Root Cause Analysis from application modelers as well as incident processors from SAP.

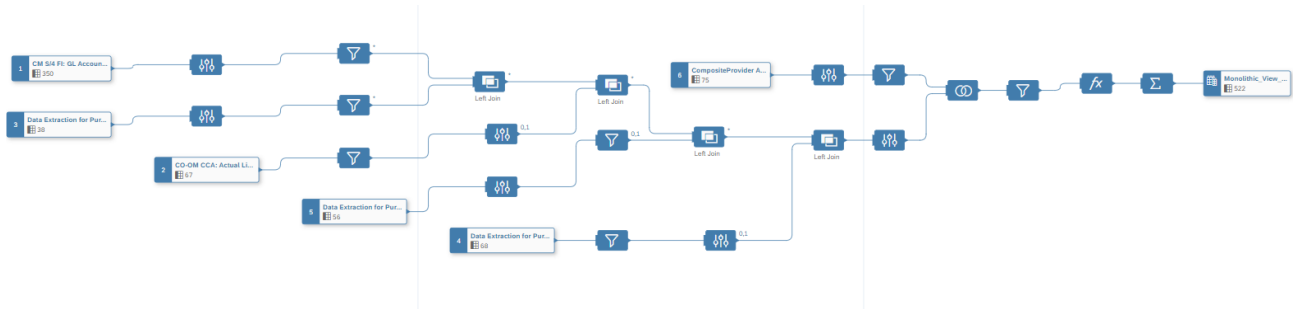


Figure 39: Monolithic model

A view as above brings several problems. In the following are a few very evident issues:

1. If there is an error in a calculation, it is hard to backtrack where the error occurred.
2. Same for performance, in this view it is not possible to analyze subsequent steps. A data preview is never a good performance check, therefore previewing sub-steps will not give any value as to where the performance problem occurs.
3. The modeler tends to get slower, the larger the model, because the design time object update takes long.
4. In the event of bad performance parts of this model will not be able to be persisted.
5. Reuse of model parts is not possible.

Alternative following the modelling approach

Split the model into logical parts.

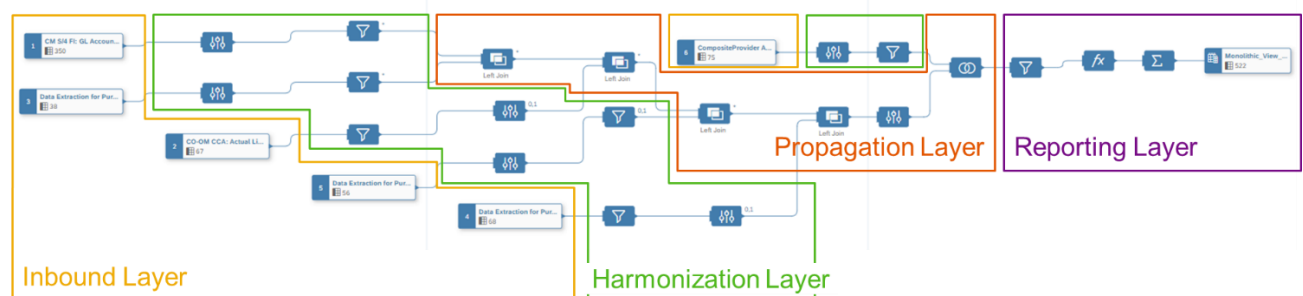


Figure 40: Logical parts of a model

Each logical layers should have a respective view representing it. This allows detailed analysis on each view in between. The propagation layer, which has a lot of joins on a high number of line-item columns can be persisted to boost performance. For re-useability the reporting layer could be transferred to the business builder.

6.3.2 Filters on Formulas

Generally, HANA can act very intelligently. The following example doesn't reflect a useful case but should show the derivative skills of the database. First we calculate a value depending on a column.



Figure 41: Filter defined on column with a formula

Example formula:

Expression
Validate

CASE WHEN ACTTYPE = '1414' THEN '1' ELSE '0' END

Figure 42: Case formula

The subsequent filter chooses one specific cost center. An execution on top of this will result in following execution plan:

Rows (2)			SQL	Download	Refresh	Undo
	OPERATOR_NAME					
1	PROJECT		RH_COOM_CCA_Actuals_LinItem.CO_AREA, RH_COOM_CCA_Actuals_LinItem.CO_DOC_NO, RH_COOM_CCA_Actuals_LinItem.CO_ITEM_NR, RH_COOM_CCA_Actuals_LinItem.FISCVARNT, RH-			
2	TABLE SCAN		FILTER CONDITION: RH_COOM_CCA_Actuals_LinItem.ACTTYPE = '1414' (DETAIL: ((SCAN) RH_COOM_CCA_Actuals_LinItem.ACTTYPE = '1414'))			

Figure 43: Execution plan shows derivation of filter

HANA derives the Activity Type as a filter condition from the formula. This works for an even more complex approach, using a Join with defined cardinality and a remote source:



Figure 44: Filter before operation sequence

Rows (6)			SQL	Download	Refresh	Undo
	OPERATOR_NAME					
1	PROJECT		IFCOSTCENTER.CONTROLLINGAREA, IFCOSTCENTER.COSTCENTER, IFGLACCTLIR_2.SOURCEIDGER, IFGLACCTLIR_2.COMPANYCODE, IFGLACCTLIR_2.FISCALYEAR, IFGLACCTLIR_2.GLACCC			
2	HASH JOIN		HASH CONDITION: IFGLACCTLIR_2.CONTROLLINGAREA = IFCOSTCENTER.CONTROLLINGAREA, HASH SIZE: 1			
3	AGGREGATION		GROUPING: IFGLACCTLIR_2.SOURCEIDGER, IFGLACCTLIR_2.COMPANYCODE, IFGLACCTLIR_2.FISCALYEAR, IFGLACCTLIR_2.CONTROLLINGAREA, IFGLACCTLIR_2.GLACCCOUNT, IFGLACCTLIR			
4	REMOTE SCAN		Remote Source: DW_C_BEST_PRACTICES.PWSCNT715_ABAP, SELECT "DW_C_BEST_PRACTICES.PWSCNT715_ABAP::ABAP_CDS.IFGLACCTLIR\$F_VT"."SOURCEIDGER", "DW_C_BEST_PRACTICES			
5	REMOTE TABLE		FILTER CONDITION: IFGLACCTLIR_2.COSTCENTER = '0000196400'			
6	TABLE SCAN		FILTER CONDITION: IFCOSTCENTER.COSTCENTER = '0000196400' (DETAIL: ((INDEX LOOK-UP) IFCOSTCENTER.COSTCENTER = '0000196400'))			

Figure 45: Filter push down on remote table & local table

Following functions prevent filter push-downs: rank nodes, window function nodes, and nodes that feed into two other nodes.

6.3.3 Data Transposition

In some cases, pure account-based models need to be transposed to mixed or measure based models. Here a simple example of splitting a measure by an “account”, in this case the unit of measure.

In this simple example, there is a table with a few columns where amount and quantity are in the same column, which makes calculations harder later on.

Date	Product	Unit/Currency	Measure
Jan 4, 2022	Banana	EUR	96.450000000000000000000000
Jan 10, 2022	Apple	PCS	9.000000000000000000000000
Jan 13, 2022	Apple	PCS	20.000000000000000000000000
Jan 6, 2022	Apple	EUR	25.900000000000000000000000
Jan 14, 2022	Orange	PCS	12.000000000000000000000000
Jan 17, 2022	Apple	PCS	12.000000000000000000000000
Jan 10, 2022	Cherry	PCS	88.000000000000000000000000
Jan 15, 2022	Pear	EUR	74.250000000000000000000000

Figure 46: Base data for data transpose

The model created on top is defined as follows.



Figure 47: Transpose model via formula

Element Properties

fx

Quantity

0

<Columns / Quantity

Business Name:

Quantity

Technical Name:

Quantity

Data Type:

AA String

Length:

10

Expression

CASE WHEN "Unit/Currency"='PCS' THEN Measure ELSE '0' END

Validate

Figure 48: Formula on 4 columns to determine currency or unit values

An alternative to using a formula, is to use filters and unions. This parallelizes operations and uses more resources to complete the operation. The filter is on Currency/Unit, the projections rename the columns, and the formula adds two columns to match in the union.

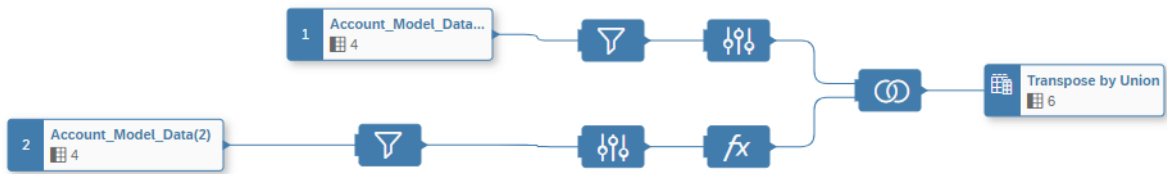


Figure 49: Transpose via union

Execution comparison:

Transpose via Formula

Rows (2)

	OPERATOR_NAME	
1	PROJECT	Transpose_via_formula.Date, Transpose_via_formula.P
2	TABLE SCAN	

Transpose via Union

Rows (4)

	OPERATOR_NAME	
1	PROJECT	Transpose_by_Union.Date, Transpose_by_Union.Product, Transpose_by_Union.Unit, Transpose_I
2	UNION ALL	(Account_Model_Data(2).Date, Account_Model_Data(1).Date) UNION_COLO, (Account_Model_D
3	TABLE SCAN	FILTER CONDITION: Account_Model_Data(2).Unit/Currency = 'PCS' (DETAIL: ([SCAN] Account_M
4	TABLE SCAN	FILTER CONDITION: Account_Model_Data(1).Unit/Currency = 'EUR' (DETAIL: ([SCAN] Account_M

Figure 50: Comparison of execution plans

The execution on the left uses one thread to run a full table scan and perform the operations. This operation can be costly the more data there is and the more formulas need to operate, meaning the more measures need to result out of the transposition. The effort grows the higher the granularity and the more measures are being created.

The execution on the right uses two threads, filters data and operates in parallel on both subsets. This model parallelizes the effort, will consume more memory and due to the parallel operations, consumes more CPU. But is usually faster and a union allows the optimizer to better aggregate data, if the query on top defines an aggregation. This model looks more complex and normally results in more modelling effort.

7 Root cause Analysis for performance problems

7.1 Performance Root Cause Analysis for View Persistency

The purpose of this section is to help you find the probable root cause of a long running or memory intensive Datasphere View Persistency execution. Performance issues may be difficult to diagnose, problems may be rooted in several seemingly unrelated components. By observing the general symptoms shown by the system such as high memory usage, poor performance, etc. we can start to narrow down the possible causes as a first step.

7.1.1 Memory Analysis

[View Persistency Monitor](#)
[Expensive Statement Trace](#)
[SAP HANA Cockpit](#)

There are several ways within Datasphere and SAP HANA to identify the actual/historical memory usage of a view persistency run and put it into relation to the available system resources.

7.1.1.1 View Persistency Monitor

The Datasphere View Persistency Monitor is a good starting point to get an overview of the runtime, number of loaded records and memory usage of a specific Datasphere View Persistency run.

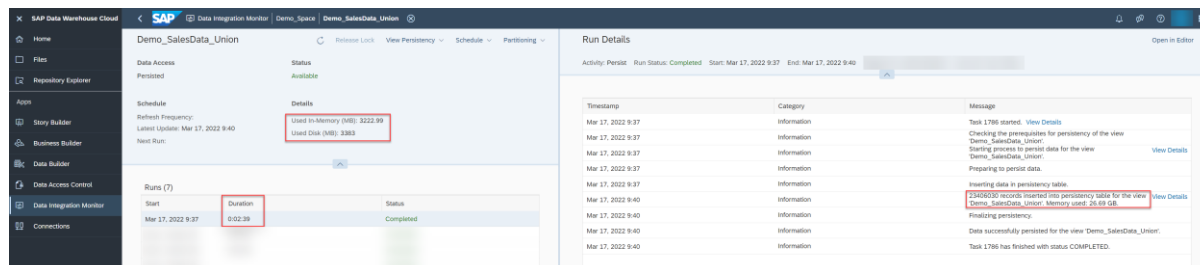


Figure 51: View Persistency Monitor

In the overview section the status of the Datasphere View Persistency as well as the memory and disk usage on database is shown in case the view is persisted. By clicking on a dedicated run, the right side of the screen will update with the run details. Those details show when the run started/finished, how many records got inserted in the persistency table and how much memory it consumed as a maximum during execution.

By checking the run details of historical runs, it could also be verified if the number of records as well as the memory used to persist the data is stable, has increased or even decreased (e.g., due to data volume differences on source tables/views or design changes in the underlying base views/tables).

As an alternative it might also be interesting to query the "Datasphere_GLOBAL"."TASK_LOG_MESSAGES" table directly via HANA Database Explorer to get a better historical view of the past runs. This table contains all Task Log related information which can be found in the Data Integration Monitor on Datasphere. To query the database table via the Database Explorer a Database Analysis User is needed. More details on how to create such a user and how to access the Database Explorer is explained in the [Datasphere Administration Guide – Database Analysis User](#).

In the example below the TASK_LOG_MESSAGES of the Datasphere View 'Demo_SalesData_Union' are queried. The example shows how the number of persisted records increased and how this has impacted the memory used. This kind of check might be of relevance in case the data volume increases month over month or year over year to check if the system itself is still capable of handling the expected growth.

```

1 select TASK_LOG_ID, TIMESTAMP, TEXT, PARAMETER_VALUES from DMC_GLOBAL.TASK_LOG_MESSAGES
2 where MESSAGE_BUNDLE_KEY = 'START_VIEW_PERSISTENCY_RECORD_COUNT_SUCCESS_V2'
3 and PARAMETER_VALUES like '%<DMC_VTEN_NAME>%'
4 order by TASK_LOG_ID desc
5

```

TASK_LOG_ID	TIMESTAMP	TEXT	PARAMETER_VALUES
1786	2022-03-17 08:40:24.573000000	23406030 records inserted into persistency table for the view 'Demo_SalesData_Union'. Memory used: 26.69 GB.	["23406030";"Demo_SalesData_Union";"26.69"]
1784	2022-03-17 08:35:44.929000000	15343953 records inserted into persistency table for the view 'Demo_SalesData_Union'. Memory used: 17.55 GB.	["15343953";"Demo_SalesData_Union";"17.55"]
1782	2022-03-17 08:28:12.127000000	8062077 records inserted into persistency table for the view 'Demo_SalesData_Union'. Memory used: 9.29 GB.	["8062077";"Demo_SalesData_Union";"9.29"]

Figure 52: Task Log Messages

7.1.1.2 Expensive Statement Trace

Besides the View Persistency Monitor the Expensive Statement Trace can give additional information regarding Runtime, Memory Consumption and CPU time of Datasphere View Persistency runs.

To have Expensive Statement Trace information available the trace needs to be turned on. The [Datasphere Administration Guide – Setting up a Monitoring Space in SAP Datasphere](#) shows how to switch on the trace. The monitoring view can be accessed afterwards via a dedicated Monitoring Space or also via the HANA Database Explorer. The relevant HANA Monitoring view is "SYS"."M_EXPENSIVE_STATEMENTS".

The expensive Statements Trace records all queries on HANA which fulfill the configured threshold. To only get the relevant ones for View Persistency, the SQL statement should be restricted as shown below.

```

10 select schema_name,
11 start_time,
12 round((duration_microsec/1000000),2) as duration_sec,
13 records, round((memory_size/1024/1024),2) as memory_gb,
14 round((cpu_time/1000000),2) as cpu_time_sec,
15 workload_class_name,
16 statement_thread_id,
17 statement_memory_limit,
18 statement_string,
19 parameters
20 from SYS.M_EXPENSIVE_STATEMENTS
21 where schema_name = 'DMC_Space_Name'
22 and parameters like '%DMC_Space_Name%'
23 and statement_string like '%START_VIEW_PERSISTENCY_RECORD%'
24 order by start_time desc

```

SCHEMA_NAME	START_TIME	DURATION_SEC	RECORDS	MEMORY_GB	CPU_TIME	WORKLOAD_CLASS_NAME	STATEMENT_THREAD	STATEMENT_MEMORY_LIMIT	STATEMENT_STRING	PARAMETERS
DEMO_SPACE	2022-03-17 10:30:47.919510000	473.820000	23406030	26.690000000000000	783.820000	SAP_DMC_DEMO_SPACE	8	120	CALL 'DEMO_SPACE'EC' '\$\$VIEW_MATERIALIZATION_ON\$S' (' Demo_SalesData_Union', ASALOMON, 1782, Table, DE	
DEMO_SPACE	2022-03-17 10:45:31.351261000	207.190000	15343953	17.550000000000000	504.230000	SAP_DMC_DEMO_SPACE	8	120	CALL 'DEMO_SPACE'EC' '\$\$VIEW_MATERIALIZATION_ON\$S' (' Demo_SalesData_Union', ASALOMON, 1784, Table, DE	
DEMO_SPACE	2022-03-17 10:23:26.406532000	96.110000	8062089	9.290000000000000	268.330000	SAP_DMC_DEMO_SPACE	8	120	CALL 'DEMO_SPACE'EC' '\$\$VIEW_MATERIALIZATION_ON\$S' (' Demo_SalesData_Union', ASALOMON, 1780, Table, DE	

Figure 53: SQL expensive statement trace

The screenshot shows three Datasphere View Persistency runs of View 'Demo_SalesData_Union' with growing data volume and increasing memory consumption. Besides the information which can already be seen within the Datasphere Data Integration Monitor, the Expensive Statement Trace also shows additional information which might be of relevance. Especially the Workload Class Statement Settings which are configured at Space level can be seen in correlation with the Datasphere View Persistency executions. In above example all statements in this Space are allowed to consume in total 8 threads and 120GB of memory. Please keep in mind that on Datasphere Space level only 'total' values can be configured. This means all queries executed in parallel within the context of this space need to be within those limits (this includes View Persistency runs, SAC Queries, 3rd Party tool queries, etc.).

If there are already some expensive Datasphere View Persistency runs that come close to the configured total limits, depending on the parallel activities, runs may occasionally abort with an Out of Memory situation. Here it needs to be considered to either increase the statement limits on space level or reduce the memory footprint of the expensive Datasphere View Persistency run.

There is only a limited possibility to use the Expensive Statement Trace for a historical analysis. Depending on the system load, historical entries are overwritten after a certain time period.

7.1.1.3 HANA Cockpit Load Monitor

As a third option also the HANA Cockpit Load Monitor can be used to check how much Memory and CPU resources of the underlying HANA database are utilized. The [DataspHERE Administration Guide – Connecting with SAP HANA cockpit](#) describes how to connect to the HANA cockpit from DataspHERE. The Load Monitor can be configured, and different dimensions can be turned on/off. For analyzing CPU and Memory consumption it is a good starting point to begin with the Service KPIs (CPU, Memory Used and Memory Allocation Limit).



Figure 54: Load Monitor

The Load Monitor itself shows the overall system resources of the underlying HANA Database of the DataspHERE tenant. Here it is not possible to directly filter on a space level or dedicated DataspHERE View Persistency run. It serves more as a basis for a top-down analysis. If there are dedicated time windows within a day/hour where it can be observed that either a lot of memory is consumed, or the CPU is 100% utilized for a longer time, it makes sense to look at those time windows to identify the expensive statements.

The screenshot above is a 30min time frame of the demo system and the situation is pretty relaxed. Nevertheless, the three DataspHERE View Persistency executions for the DataspHERE View 'Demo_SalesData_Union' (see Expensive Statement Trace) can be mapped with peaks in the memory and CPU lines.

In a production system this load graph often looks different based on the number of DataspHERE View Persistency runs and concurrent end user query load. During those phases with high CPU and Memory usage the Expensive Statement Trace can be checked to identify the most expensive queries. A result of this analysis could be that too many schedules are running in parallel, and a re-scheduling might be helpful.

If an analysis takes place on both SAP HANA and DataspHERE it must be considered that SAP HANA indicates the time in UTC whereas the time in DataspHERE is local time.

7.1.2 Analyzing HANA Execution Plans for Datasphere View Persistency (with examples)

[Retrieving Datasphere View Persistency INSERT INTO statement](#)
[Analyzing SQL Execution with the Plan Explanation \(Explain Plan\)](#)
[Analyzing SQL Execution with the Plan Visualizer \(PlanViz\)](#)

Besides the Memory Analysis on Datasphere and HANA side it can also be helpful to look deeper into the execution plans on HANA side to identify if queries are processed inefficiently. You can technically and logically analyze the steps SAP HANA took to process those queries.

From a technical perspective, analyzing query plans allows you to identify long running steps, understand how much data is processed by the operators and see whether data is processed in parallel. However, if you understand the idea and purpose behind the query, you can also analyze query plans from a logical perspective and answer questions like (Does SAP HANA read too much data from tables which are not needed, are filters processed too late, is the sequence of joins correct etc.).

To gain the insights the Plan explanation and/or Plan visualization Tools can be used. Both tools are available within the SAP HANA Cockpit using the Datasphere Analysis User.

7.1.2.1 Retrieving Datasphere View Persistency INSERT INTO statement

Before it is possible to start the analysis of the SAP HANA execution plans it is necessary to derive the SQL Statement Datasphere triggers to execute the View Persistency run. The easiest way to get to the SQL statement is via the View Persistency Monitor. Within the Run Details section of a dedicated Datasphere View the INSERT INTO statement can be displayed via the 'View Details' link. Please make sure this is checked within a persist activity. From the 'View Details' popup the relevant SQL statement can be copied. Be aware that the View Persistency Monitor also shows other activities like 'Remove persisted data'. From those activities it is not possible to derive Insert statement.

The screenshot displays the 'View Persistency Monitor' for the 'Demo_SalesData_Union' view. The left sidebar shows the view's status as 'Available' and its schedule. The main area is divided into 'Runs (12)' and 'Run Details'. The 'Run Details' section for the 'Persist' activity shows a list of messages. A red box highlights the 'View Details' link in the 'Run Details' table, which opens a popup window. This popup displays the SQL statement: `INSERT INTO 'DEMO_SPACE'/'Demo_SalesData_Union_STRT' (SELECT * FROM 'DEMO_SPACE'/'Demo_SalesData_Union_STV');`. A red arrow points to the 'View Details' link in the 'Run Details' table, and another red arrow points to the 'View Details' link in the 'Run Details' table.

Timestamp	Category	Message
Mar 23, 2022 9:26	Information	Task 1825 started. View Details
Mar 23, 2022 9:26	Information	Checking the prerequisites for persistency of the view 'Demo_SalesData_Union'.
Mar 23, 2022 9:26	Information	Starting process to persist data for the view 'Demo_SalesData_Union'. View Details
Mar 23, 2022 9:26	Information	Preparing to persist data.
Mar 23, 2022 9:26	Information	Inserting data in persistency table.
Mar 23, 2022 9:28	Information	15343953 records inserted into persistency table for the view 'Demo_SalesData_Union'. Memory used: 17.55 GB. View Details
Mar 23, 2022 9:28	Information	Finalizing persistency.
Mar 23, 2022 9:28	Information	17.95 GB of peak memory used in the view persistency runtime.
Mar 23, 2022 9:28	Information	Data successfully persisted for the view 'Demo_SalesData_Union'.
Mar 23, 2022 9:28	Information	Task 1825 has finished with status COMPLETED.

Figure 55: View Persistency Monitor – View Details

7.1.2.2 Analyzing SQL Execution with the Plan Explanation (Explain Plan)

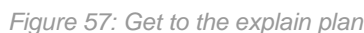
You can generate a plan explanation for any SQL statement and use this to evaluate the execution plan. You may be able to use this information to optimize the Datasphere View/View Stack to reduce run time or the memory consumption. There are several ways to collect the Explain Plan. Below I will only show how to do this directly from the HANA Database Explorer.

View Details

```
INSERT INTO "DEMO_SPACE"."SalesData_Reporting_$TR" (SELECT * FROM "DEMO_SPACE"."SalesData_Reporting_$TV");
```

Copy Close

After entering the SELECT statement into the SQL console you can hit the 'Analyze' button and select 'Explain Plan'.



OPERATOR	NAME	OPERATOR DETAILS	EXECUTION	TABLE NAME	TABLE TYPE	TABLE SIZE	OUTPUT SIZE	SUBSTRATE COST	LEVEL
1	EXS SEARCH	Union_SalesData.CALDAY, Union_SalesData.FISCVARINT, Union_SalesData.FISCOPER, Union_SalesData.IBAT5_PLANT, Union_SalesData.IBAT5_MAT, ESX	NULL	NULL	NULL	NULL	137726664.70579161	2258164.480493059	1
2	HASH JOIN	HASH BUILD: RIGHT JOIN CONDITION: CONCAT(LEFT(SalesData.IBAT5_SLSORG, 1), CONCAT(V, RIGHT(SalesData.IBAT5_SLSORG, 3))) = CONCAT(ESX	NULL	NULL	NULL	NULL	137726664.70579161	22517060.659928334	2
3	REMOTE COLUMN SCAN	Remote Source: DW_CMO_DEMO_SPACE_HANA_A51, SELECT 'SalesData_Remote_H2SVT', 'IBAT5_COMP_CODE', 'SalesData_Remote_H2SVT', 'IBAT5_COUNTRY', CONCAT(LEFT(EXTERNAL	SalesData_Remote_H2SVT	EXTERNAL	NULL	250000	250000	3
4	REMOTE TABLE	FILTER CONDITION: SalesData.ObjValues = 'K'	EXTERNAL	SalesData_ObjValues	VIRTUAL TABLE	1000000	250000	NULL	4
5	COLUMN SEARCH	Union_SalesData.CALDAY, Union_SalesData.FISCVARINT, Union_SalesData.FISCOPER, Union_SalesData.IBAT5_PLANT, Union_SalesData.IBAT5_MAT, COLUMN	NULL	NULL	NULL	NULL	4407.253270585478	2000993.9904834752	5
6	JOIN	JOIN CONDITION: (INNER Union_SalesData.IBAT5_MAT) = ADVERTISEMENT_INFORMATION.MATERIAL	COLUMN	NULL	NULL	NULL	4407.253270585478	2000993.97143308	6
7	COLUMN SEARCH	Union_SalesData.CALDAY, SalesData_Remote_H2SVT, SalesData_FISCOPER, Union_SalesData.IBAT5_PLANT, Union_SalesData.IBAT5_MAT, ESX	NULL	P_SVS_OO_CO_L219_Tbbo	NULL	NULL	2452592.68	2000993.7023177068	7
8	COLUMN UNION ALL	SalesData_Remote_H2_CALDAY, SalesData_Remote_H2_CALDAY, SalesData_HANA_H2_CALDAY, SalesData_HANA_H2_CALDAY, SalesData_BW_H2_CJ, COLUMN	NULL	NULL	NULL	NULL	2162599.149	2000991.923177068	8
9	EXS SEARCH	SalesData_Remote_H2_CALDAY, SalesData_Remote_H2_FISCVARINT, SalesData_Remote_H2_FISCOPER, SalesData_Remote_H2_IBAT5_PLANT, SalesData_ESX	NULL	NULL	NULL	NULL	125000	1000000.5339588874	9
10	FILTER	CONCAT(LEFT(Union_SalesData.IBAT5_SLSORG, 1), CONCAT(V, RIGHT(Union_SalesData.IBAT5_SLSORG, 3))) = 'S0001'	ESX	NULL	NULL	NULL	125000	1000000.5339588874	10
11	REMOTE COLUMN SCAN	Remote Source: DW_CMO_DEMO_SPACE_BW_AP1, SELECT 'SalesData_Remote_H2SVT', 'CALDAY', 'SalesData_Remote_H2SVT', 'FISCVARINT', 'SalesData	EXTERNAL	SalesData_Remote_H2SVT	EXTERNAL	NULL	1000000	1000000	9
12	REMOTE TABLE	FILTER CONDITION: SalesData_Remote_H2SVT = 'K'	EXTERNAL	SalesData_Remote_H2SVT	VIRTUAL TABLE	1000000	1000000	NULL	10
13	EXS SEARCH	SalesData_Remote_H1_CALDAY, SalesData_Remote_H2_FISCVARINT, SalesData_Remote_H1_FISCOPER, SalesData_Remote_H1_IBAT5_PLANT, SalesData_ESX	NULL	NULL	NULL	NULL	125000	1000000.5339588874	7
14	FILTER	CONCAT(LEFT(Union_SalesData.IBAT5_SLSORG, 1), CONCAT(V, RIGHT(Union_SalesData.IBAT5_SLSORG, 3))) = 'S0001'	ESX	NULL	NULL	NULL	125000	1000000.5339588874	8
15	REMOTE COLUMN SCAN	Remote Source: DW_CMO_DEMO_SPACE_BW_AP1, SELECT 'SalesData_Remote_H2SVT', 'CALDAY', 'SalesData_Remote_H2SVT', 'FISCVARINT', 'SalesData	EXTERNAL	SalesData_Remote_H2SVT	EXTERNAL	NULL	1000000	1000000	9
16	REMOTE TABLE	FILTER CONDITION: SalesData_Remote_H2SVT = 'K'	EXTERNAL	SalesData_Remote_H2SVT	VIRTUAL TABLE	1000000	1000000	NULL	10
17	COLUMN SEARCH	SalesData_HANA_H2_CALDAY, SalesData_HANA_H2_FISCVARINT, SalesData_HANA_H2_FISCOPER, SalesData_HANA_H2_IBAT5_PLANT, SalesData_HANA_H2_CJ, COLUMN	NULL	NULL	NULL	NULL	4707.2327	646865.524	17
18	COLUMN TABLE	FILTER CONDITION: CONCAT(LEFT(Union_SalesData.IBAT5_SLSORG, 1), CONCAT(V, RIGHT(Union_SalesData.IBAT5_SLSORG, 3))) = 'S0001'	COLUMN	SalesData_HANA_H2CST	COLUMN TABLE	4707.2327	646865.524	NULL	8
19	COLUMN TABLE	SalesData_HANA_H1_CALDAY, SalesData_HANA_H1_FISCVARINT, SalesData_HANA_H1_FISCOPER, SalesData_HANA_H1_IBAT5_PLANT, SalesData_HANA_H1_CJ, COLUMN	NULL	NULL	NULL	NULL	47852.328	382812.614	19
20	COLUMN TABLE	FILTER CONDITION: CONCAT(LEFT(Union_SalesData.IBAT5_SLSORG, 1), CONCAT(V, RIGHT(Union_SalesData.IBAT5_SLSORG, 3))) = 'S0001'	COLUMN	SalesData_HANA_H1CST	COLUMN TABLE	47852.328	382812.614	NULL	8
21	COLUMN SEARCH	SalesData_BW_H2_CALDAY, SalesData_BW_H2_FISCVARINT, SalesData_BW_H2_IBAT5_PLANT, SalesData_BW_H2_CJ, COLUMN	NULL	NULL	NULL	NULL	4787.84	409666666666665	21
22	COLUMN TABLE	FILTER CONDITION: CONCAT(LEFT(Union_SalesData.IBAT5_SLSORG, 1), CONCAT(V, RIGHT(Union_SalesData.IBAT5_SLSORG, 3))) = 'S0001'	COLUMN	SalesData_BW_H2CST	COLUMN TABLE	4787.84	409666666666665	NULL	8
23	COLUMN SEARCH	SalesData_BW_H1_CALDAY, SalesData_BW_H1_FISCVARINT, SalesData_BW_H1_FISCOPER, SalesData_BW_H1_IBAT5_PLANT, SalesData_BW_H1_IBAT5_CJ, COLUMN	NULL	NULL	NULL	NULL	47983.1	3989516666666657	23
24	COLUMN TABLE	FILTER CONDITION: CONCAT(LEFT(Union_SalesData.IBAT5_SLSORG, 1), CONCAT(V, RIGHT(Union_SalesData.IBAT5_SLSORG, 3))) = 'S0001'	COLUMN	SalesData_BW_H1CST	COLUMN TABLE	47983.1	3989516666666657	NULL	8
25	COLUMN SEARCH	ADVERTISEMENT_INFORMATION.ADVVERTISEMENT, ADVERTISEMENT_INFORMATION.MATERIAL, ADVERTISEMENT_INFORMATION.ATYPE, ADVVERTISEMENT_INFORMATION	EXTERNAL	P_SVS_OO_CO_L219_Tbbo	NULL	NULL	976.5625	976.5625	5
26	REMOTE COLUMN SCAN	Remote Source: DW_CMO_DEMO_SPACE_HANA_A51, SELECT 'ADVERTISEMENT_INFORMATION', 'ADVERTISEMENT_INFORMATION', 'ADVERTISEMENT_INFORMATION', 'ADVERTISEMENT_INFORMATION'	EXTERNAL	NULL	NULL	NULL	976.5625	976.5625	6
27	JOIN (INNER)	JOIN CONDITION: Material.Material = 'ADVERTISEMENT_INFORMATION.MATERIAL'	EXTERNAL	NULL	NULL	NULL	2441448.625	976.5625	7
28	REMOTE TABLE	FILTER CONDITION: Material.ObjValue = 'K'	EXTERNAL	MaterialObjValue	VIRTUAL TABLE	1000000	976.5625	NULL	8
29	REMOTE TABLE	FILTER CONDITION: ADVERTISEMENT_INFORMATION.SALESORG = 'S0001'	EXTERNAL	ADVERTISEMENT_INFORMATION	VIRTUAL TABLE	1000000	976.5625	NULL	8

Figure 58: Explain Plan Result

Besides the involved tables it can also be seen what type of Join, Union, etc. is used and in what sequence those are planned to be executed. The 'Execution Engine' highlights in which HANA Engine the plan is executed and what is running in parallel or sequentially (visually indented in column 'Operator Name' or

highlighted in column 'Level'). If there are multiple execution engines involved intermediate results might be required which can increase the memory usage of the Datasphere View Persistency. In above Explain Plan example there are two intermediated result tables required. They are indicated with technical names like '#_SYS_*'. The execution engines are chosen according to the cost-based estimations and based on the features modeled in the Datasphere views. Not all engines in HANA support all features.

In the 'Operator Details' section you find information about selected columns, filter conditions, join columns, etc. This might be of importance e.g., if you want to check that applied filters in the Datasphere View are pushed down to all involved tables or if they are somehow stuck in higher levels of the plan because of data type issues, formulas, etc..

As you can see, the Plan explanation functionality can already give you a comprehensive overview. There are already a lot of information available without even executing the query. To better understand those plans it is also helpful to know the Datasphere scenario. This makes things easier to better map the information provided in the plan explanation to the actual Datasphere View or Datasphere View Stack.

Possible Findings:

	OPERATOR_NAME	OPERATOR_DETAILS	EXECUTION	TABLE_NAME	TABLE_TYPE	TABLE_SIZE	OUTPUT_SIZE	SUBTREE_COST	LEVEL
1	ESX SEARCH	Union_SalesData.CALDAY, Union_SalesData.FISCVARINT, Union_SalesData.BA7IS_PLANT, Union_SalesData.BA7IS_MAT, ESX	NULL	NULL	NULL	35258026164.68382	4335617.985264753	1	
2	HASH JOIN	HASH BUILD: RIGHT, JOIN CONDITION: CONCAT(LEFT(Union_SalesData.BA7IS_SL5ORG, 1), CONCAT('0', RIGHT(Union_SalesData.BA7IS_SL5ORG, 3))) = 'S0001'	ESX	NULL	NULL	35258026164.68382	2511918.8967012423	2	
3	HASH JOIN	HASH BUILD: RIGHT, JOIN CONDITION: CONCAT(LEFT(Union_SalesData.BA7IS_MAT, 1), CONCAT('0', RIGHT(Union_SalesData.BA7IS_MAT, 3))) = 'S0001'	ESX	NULL	NULL	1126256.887269882	2250019.24504824	3	
4	COLUMN UNION ALL	(SalesData_Remote_H2_CALDAY, SalesData_Remote_H1_CALDAY, SalesData_HANA_H1_CALDAY, SalesData_BW_H2_CALDAY, SalesData_Remote_H2_CALDAY, SalesData_Remote_H1_CALDAY, SalesData_HANA_H1_CALDAY, SalesData_BW_H2_CALDAY)	ESX	NULL	NULL	2165299.348	2000017.029177068	4	
5	ESX SEARCH	SalesData_Remote_H2_CALDAY, SalesData_Remote_H2_FISCVARINT, SalesData_Remote_H2_BA7IS_PLANT, SalesData_Remote_H2_BA7IS_MAT, ESX	ESX	NULL	NULL	125000	1000000.5339588874	5	
6	FILTER	CONCAT(LEFT(Union_SalesData.BA7IS_SL5ORG, 1), CONCAT('0', RIGHT(Union_SalesData.BA7IS_SL5ORG, 3))) = 'S0001'	ESX	NULL	NULL	125000	1000000.5339588874	6	
7	REMOTE COLUMN SCAN	Remote Source: DW_C_DEMO_SPACE.BW.AP1, SELECT "SalesData_Remote_H2SVT"."CALDAY", "SalesData_Remote_H2SVT"."FISCVARINT", "SalesData_Remote_H2SVT"."BA7IS_PLANT", "SalesData_Remote_H2SVT"."BA7IS_MAT"	EXTERNAL	NULL	NULL	1000000	1000000	7	
8	REMOTE TABLE		EXTERNAL	SalesData_Remote_H2SVT	VIRTUAL TABLE	1000000	1000000	8	
9	ESX SEARCH	SalesData_Remote_H1_CALDAY, SalesData_Remote_H1_FISCVARINT, SalesData_Remote_H1_FISCVARINT, SalesData_Remote_H1_BA7IS_PLANT, SalesData_Remote_H1_BA7IS_MAT, ESX	ESX	NULL	NULL	125000	1000000.5339588874	9	
10	FILTER	CONCAT(LEFT(Union_SalesData.BA7IS_SL5ORG, 1), CONCAT('0', RIGHT(Union_SalesData.BA7IS_SL5ORG, 3))) = 'S0001'	ESX	NULL	NULL	125000	1000000.5339588874	10	
11	REMOTE COLUMN SCAN	Remote Source: DW_C_DEMO_SPACE.BW.AP1, SELECT "SalesData_Remote_H1SVT"."CALDAY", "SalesData_Remote_H1SVT"."FISCVARINT", "SalesData_Remote_H1SVT"."BA7IS_PLANT", "SalesData_Remote_H1SVT"."BA7IS_MAT"	EXTERNAL	NULL	NULL	1000000	1000000	11	
12	REMOTE TABLE		EXTERNAL	SalesData_Remote_H1SVT	VIRTUAL TABLE	1000000	1000000	12	
13	COLUMN SEARCH	SalesData_HANA_H2_CALDAY, SalesData_HANA_H2_FISCVARINT, SalesData_HANA_H2_FISCVARINT, SalesData_HANA_H2_BA7IS_PLANT, SalesData_HANA_H2_BA7IS_MAT, COLUMN	NULL	NULL	47072127	564885.524	47072126999999995	13	
14	COLUMN TABLE	FILTER CONDITION: CONCAT(LEFT(Union_SalesData.BA7IS_SL5ORG, 1), CONCAT('0', RIGHT(Union_SalesData.BA7IS_SL5ORG, 3))) = 'S0001'	COLUMN	SalesData_HANA_H2SVT	COLUMN TABLE	47072127	564885.524	14	
15	COLUMN SEARCH	SalesData_HANA_H1_CALDAY, SalesData_HANA_H1_FISCVARINT, SalesData_HANA_H1_FISCVARINT, SalesData_HANA_H1_BA7IS_PLANT, SalesData_HANA_H1_BA7IS_MAT, COLUMN	NULL	NULL	47952328	382818.624	3.19015519999999995	15	
16	COLUMN TABLE	FILTER CONDITION: CONCAT(LEFT(Union_SalesData.BA7IS_SL5ORG, 1), CONCAT('0', RIGHT(Union_SalesData.BA7IS_SL5ORG, 3))) = 'S0001'	COLUMN	SalesData_HANA_H1SVT	COLUMN TABLE	47952328	382818.624	16	
17	COLUMN SEARCH	SalesData_BW_H2_CALDAY, SalesData_BW_H2_FISCVARINT, SalesData_BW_H2_FISCVARINT, SalesData_BW_H2_BA7IS_PLANT, SalesData_BW_H2_BA7IS_MAT, COLUMN	NULL	NULL	487784	487784	4.064666666666665	17	
18	COLUMN TABLE	FILTER CONDITION: CONCAT(LEFT(Union_SalesData.BA7IS_SL5ORG, 1), CONCAT('0', RIGHT(Union_SalesData.BA7IS_SL5ORG, 3))) = 'S0001'	COLUMN	SalesData_BW_H2SVT	COLUMN TABLE	487784	487784	18	
19	COLUMN SEARCH	SalesData_BW_H1_CALDAY, SalesData_BW_H1_FISCVARINT, SalesData_BW_H1_FISCVARINT, SalesData_BW_H1_BA7IS_PLANT, SalesData_BW_H1_BA7IS_MAT, COLUMN	NULL	NULL	479831	479831	3.998591666666665	19	
20	COLUMN TABLE	FILTER CONDITION: CONCAT(LEFT(Union_SalesData.BA7IS_SL5ORG, 1), CONCAT('0', RIGHT(Union_SalesData.BA7IS_SL5ORG, 3))) = 'S0001'	COLUMN	SalesData_BW_H1SVT	COLUMN TABLE	479831	479831	20	
21	REMOTE COLUMN SCAN	Remote Source: DW_C_DEMO_SPACE.HANA.AS1, SELECT "ADVERTISEMENT_INFORMATIONSVT"."ADVERTISEMENT", "ADVERTISEMENT_INFORMATIONSVT"."MATERIAL", "ADVERTISEMENT_INFORMATIONSVT"."OBJVERS"	EXTERNAL	NULL	NULL	250000	250000	21	
22	JOIN (INNER)	JOIN CONDITION: Material.ObjVERS = 'A'	EXTERNAL	NULL	NULL	157135249984	250000	22	
23	REMOTE TABLE		EXTERNAL	MaterialSVT	VIRTUAL TABLE	1000000	250000	23	
24	REMOTE TABLE	FILTER CONDITION: ADVERTISEMENT_INFORMATION.SALESORG = 'S0001'	EXTERNAL	ADVERTISEMENT_INFORMATIONSVT	VIRTUAL TABLE	628541	628540.9999371495	24	
25	REMOTE COLUMN SCAN	Remote Source: DW_C_DEMO_SPACE.HANA.AS1, SELECT "SalesOrgSVT"."BA7IS_COMP_CODE", "SalesOrgSVT"."BA7IS_COUNTRY", CONCAT(LEFT(Union_SalesData.BA7IS_SL5ORG, 1), CONCAT('0', RIGHT(Union_SalesData.BA7IS_SL5ORG, 3))) = 'S0001'	EXTERNAL	NULL	NULL	250000	250000	25	
26	REMOTE TABLE		EXTERNAL	SalesOrgSVT	VIRTUAL TABLE	1000000	250000	26	

Figure 59: Explain Plan Findings

- Especially for remote tables it is important to have accurate statistics because those can have a huge impact on the execution plans. In the demo scenario there is a mixture of HANA and ABAP remote sources. For SDI ABAP sources it is not possible to create statistics but for the HANA tables involved I have created statistics. The table included in the scenario is relatively small and contains approx. 630.000 records. This is also now shown in the Explain Plan after the [statistics got created](#) on the remote table (column 'Tables Size'). In addition, the Filter on 'SALESORG = S0001' is applied on the remote table and therefore pushed to the remote database. The 'Output Size' gives you an indication on the selectivity of the Filter. Here the table only contains data for Sales Organization 'S0001' and hence does not restrict the data volume. If you also compare the original plan (check screenshot further above) with the plan after the creation of the statistics, you can see that the SAP HANA optimizer decided for a new join sequence based on updated estimations/costs.
- Similar to the remote table it is also important to check if modeled filters are considered on local or already replicated Datasphere tables. In our scenario there are four replicated Datasphere tables. The two tables replicated from the SAP HANA source Database ('SalesData_HANA*') contain approx. 47 million data records and the filter on 'SALESORG' will reduce the data volume to approx. 565.000 records for one and 382.000 records for the other table. Imagine there would be somehow a blocker within the model (Datasphere view) which would avoid the SAP HANA SQL Optimizer to push down the filter directly on the table. That could mean that joins/unions with other tables would happen on much higher intermediate result sets and could lead to much higher runtime and memory consumption.
- The Demo View has beside the replicated Datasphere tables and the remote HANA tables also two remote tables based on an ABAP source. It is not possible to create statistics for those types of remote tables and in addition the 'REMOTE TABLE' scan happens without the Filter on

Besides the plan explanation you can also derive detailed information of the remote query within the Datasphere Remote Query Monitor. The Demo View triggers in total four remote queries.

Figure 60: Remote Query Monitor

Figure 61: Remote Query Monitor -Details

With that information you can now also have a closer look at the Datasphere View/View Stack to identify possible blockers for the filter push down. In the Demo View the filter is applied on a calculated column with string operators. The Datasphere modeler tried to cleanse data type differences within the model. Some tables deliver the 'SALESORG' as Char4 and other as Char5 field. For some sources (remote/local) the SAP HANA Optimizer is able to convert this calculated column as a suitable filter. For others like the remote ABAP tables the conversion is not possible and hence the filter needs to be processed after all remote data is transferred to Datasphere. To show the impact I have now changed the position where the filter on 'SALESORG' was modeled within the Datasphere View. After the model change the filter is also considered for the remote ABAP sources.

In both cases the Datasphere View persists 1.8 million records and the result is still the same, but the runtime reduces from two minutes to one minute. From a memory usage point of view the query which reads less data from the remote sources uses 4.5 GB whereas the query without filter consumes 7.7 GB.

	Original scenario without optimization	Scenario with optimization (Statistics, Remote filter)																																																																		
Runtime	<table border="1"> <thead> <tr> <th>Start</th><th>Duration</th><th>Status</th></tr> </thead> <tbody> <tr> <td>Mar 25, 2022 10:23</td><td>0:01:59</td><td>Completed</td></tr> </tbody> </table>	Start	Duration	Status	Mar 25, 2022 10:23	0:01:59	Completed	<table border="1"> <thead> <tr> <th>Start</th><th>Duration</th><th>Status</th></tr> </thead> <tbody> <tr> <td>Mar 25, 2022 12:47</td><td>0:01:01</td><td>Completed</td></tr> </tbody> </table>	Start	Duration	Status	Mar 25, 2022 12:47	0:01:01	Completed																																																						
Start	Duration	Status																																																																		
Mar 25, 2022 10:23	0:01:59	Completed																																																																		
Start	Duration	Status																																																																		
Mar 25, 2022 12:47	0:01:01	Completed																																																																		
Run details	<table border="1"> <thead> <tr> <th>Timestamp</th><th>Category</th><th>Message</th></tr> </thead> <tbody> <tr><td>Mar 25, 2022 10:23</td><td>Information</td><td>Task 1844 started. View Details</td></tr> <tr><td>Mar 25, 2022 10:23</td><td>Information</td><td>Checking the prerequisites for persistency of the view 'SalesData_Reporting'. View Details</td></tr> <tr><td>Mar 25, 2022 10:23</td><td>Information</td><td>Starting process to persist data for the view 'SalesData_Reporting'. View Details</td></tr> <tr><td>Mar 25, 2022 10:23</td><td>Information</td><td>Preparing to persist data.</td></tr> <tr><td>Mar 25, 2022 10:23</td><td>Information</td><td>Inserting data in persistency table.</td></tr> <tr><td>Mar 25, 2022 10:25</td><td>Information</td><td>1805290 records inserted into persistency table for the view 'SalesData_Reporting'. Memory used: 7.69 GB. View Details</td></tr> <tr><td>Mar 25, 2022 10:25</td><td>Information</td><td>Finalizing persistency.</td></tr> <tr><td>Mar 25, 2022 10:25</td><td>Information</td><td>7.69 GB of peak memory used in the view persistency runtime.</td></tr> <tr><td>Mar 25, 2022 10:25</td><td>Information</td><td>Data successfully persisted for the view 'SalesData_Reporting'.</td></tr> <tr><td>Mar 25, 2022 10:25</td><td>Information</td><td>Task 1844 has finished with status COMPLETED.</td></tr> </tbody> </table>	Timestamp	Category	Message	Mar 25, 2022 10:23	Information	Task 1844 started. View Details	Mar 25, 2022 10:23	Information	Checking the prerequisites for persistency of the view 'SalesData_Reporting'. View Details	Mar 25, 2022 10:23	Information	Starting process to persist data for the view 'SalesData_Reporting'. View Details	Mar 25, 2022 10:23	Information	Preparing to persist data.	Mar 25, 2022 10:23	Information	Inserting data in persistency table.	Mar 25, 2022 10:25	Information	1805290 records inserted into persistency table for the view 'SalesData_Reporting'. Memory used: 7.69 GB. View Details	Mar 25, 2022 10:25	Information	Finalizing persistency.	Mar 25, 2022 10:25	Information	7.69 GB of peak memory used in the view persistency runtime.	Mar 25, 2022 10:25	Information	Data successfully persisted for the view 'SalesData_Reporting'.	Mar 25, 2022 10:25	Information	Task 1844 has finished with status COMPLETED.	<table border="1"> <thead> <tr> <th>Timestamp</th><th>Category</th><th>Message</th></tr> </thead> <tbody> <tr><td>Mar 25, 2022 12:47</td><td>Information</td><td>Task 1850 started. View Details</td></tr> <tr><td>Mar 25, 2022 12:47</td><td>Information</td><td>Checking the prerequisites for persistency of the view 'SalesData_Reporting'. View Details</td></tr> <tr><td>Mar 25, 2022 12:47</td><td>Information</td><td>Starting process to persist data for the view 'SalesData_Reporting'. View Details</td></tr> <tr><td>Mar 25, 2022 12:47</td><td>Information</td><td>Preparing to persist data.</td></tr> <tr><td>Mar 25, 2022 12:47</td><td>Information</td><td>Inserting data in persistency table.</td></tr> <tr><td>Mar 25, 2022 12:48</td><td>Information</td><td>1805290 records inserted into persistency table for the view 'SalesData_Reporting'. Memory used: 4.44 GB. View Details</td></tr> <tr><td>Mar 25, 2022 12:48</td><td>Information</td><td>Finalizing persistency.</td></tr> <tr><td>Mar 25, 2022 12:48</td><td>Information</td><td>4.44 GB of peak memory used in the view persistency runtime.</td></tr> <tr><td>Mar 25, 2022 12:48</td><td>Information</td><td>Data successfully persisted for the view 'SalesData_Reporting'.</td></tr> <tr><td>Mar 25, 2022 12:48</td><td>Information</td><td>Task 1850 has finished with status COMPLETED.</td></tr> </tbody> </table>	Timestamp	Category	Message	Mar 25, 2022 12:47	Information	Task 1850 started. View Details	Mar 25, 2022 12:47	Information	Checking the prerequisites for persistency of the view 'SalesData_Reporting'. View Details	Mar 25, 2022 12:47	Information	Starting process to persist data for the view 'SalesData_Reporting'. View Details	Mar 25, 2022 12:47	Information	Preparing to persist data.	Mar 25, 2022 12:47	Information	Inserting data in persistency table.	Mar 25, 2022 12:48	Information	1805290 records inserted into persistency table for the view 'SalesData_Reporting'. Memory used: 4.44 GB. View Details	Mar 25, 2022 12:48	Information	Finalizing persistency.	Mar 25, 2022 12:48	Information	4.44 GB of peak memory used in the view persistency runtime.	Mar 25, 2022 12:48	Information	Data successfully persisted for the view 'SalesData_Reporting'.	Mar 25, 2022 12:48	Information	Task 1850 has finished with status COMPLETED.
Timestamp	Category	Message																																																																		
Mar 25, 2022 10:23	Information	Task 1844 started. View Details																																																																		
Mar 25, 2022 10:23	Information	Checking the prerequisites for persistency of the view 'SalesData_Reporting'. View Details																																																																		
Mar 25, 2022 10:23	Information	Starting process to persist data for the view 'SalesData_Reporting'. View Details																																																																		
Mar 25, 2022 10:23	Information	Preparing to persist data.																																																																		
Mar 25, 2022 10:23	Information	Inserting data in persistency table.																																																																		
Mar 25, 2022 10:25	Information	1805290 records inserted into persistency table for the view 'SalesData_Reporting'. Memory used: 7.69 GB. View Details																																																																		
Mar 25, 2022 10:25	Information	Finalizing persistency.																																																																		
Mar 25, 2022 10:25	Information	7.69 GB of peak memory used in the view persistency runtime.																																																																		
Mar 25, 2022 10:25	Information	Data successfully persisted for the view 'SalesData_Reporting'.																																																																		
Mar 25, 2022 10:25	Information	Task 1844 has finished with status COMPLETED.																																																																		
Timestamp	Category	Message																																																																		
Mar 25, 2022 12:47	Information	Task 1850 started. View Details																																																																		
Mar 25, 2022 12:47	Information	Checking the prerequisites for persistency of the view 'SalesData_Reporting'. View Details																																																																		
Mar 25, 2022 12:47	Information	Starting process to persist data for the view 'SalesData_Reporting'. View Details																																																																		
Mar 25, 2022 12:47	Information	Preparing to persist data.																																																																		
Mar 25, 2022 12:47	Information	Inserting data in persistency table.																																																																		
Mar 25, 2022 12:48	Information	1805290 records inserted into persistency table for the view 'SalesData_Reporting'. Memory used: 4.44 GB. View Details																																																																		
Mar 25, 2022 12:48	Information	Finalizing persistency.																																																																		
Mar 25, 2022 12:48	Information	4.44 GB of peak memory used in the view persistency runtime.																																																																		
Mar 25, 2022 12:48	Information	Data successfully persisted for the view 'SalesData_Reporting'.																																																																		
Mar 25, 2022 12:48	Information	Task 1850 has finished with status COMPLETED.																																																																		
Remote queries	<table border="1"> <thead> <tr> <th>Statement Runtime</th><th>Rows</th><th>Remote SQL Statement</th></tr> </thead> <tbody> <tr> <td>1 Minutes</td><td>968,760</td><td>SELECT "SalesData_Remote_H2\$VT"."CALDAY", "SalesData_Remote_H2\$VT".</td></tr> <tr> <td>1 Minutes</td><td>955,577</td><td>SELECT "SalesData_Remote_H1\$VT"."CALDAY", "SalesData_Remote_H1\$VT".</td></tr> </tbody> </table>	Statement Runtime	Rows	Remote SQL Statement	1 Minutes	968,760	SELECT "SalesData_Remote_H2\$VT"."CALDAY", "SalesData_Remote_H2\$VT".	1 Minutes	955,577	SELECT "SalesData_Remote_H1\$VT"."CALDAY", "SalesData_Remote_H1\$VT".	<table border="1"> <thead> <tr> <th>Statement Runtime</th><th>Rows</th><th>Remote SQL Statement</th></tr> </thead> <tbody> <tr> <td>53 Seconds</td><td>488,152</td><td>SELECT "SalesData_Remote_H2\$VT"."CALDAY", "SalesData_Remote_H2\$VT".</td></tr> <tr> <td>52 Seconds</td><td>480,193</td><td>SELECT "SalesData_Remote_H1\$VT"."CALDAY", "SalesData_Remote_H1\$VT".</td></tr> </tbody> </table>	Statement Runtime	Rows	Remote SQL Statement	53 Seconds	488,152	SELECT "SalesData_Remote_H2\$VT"."CALDAY", "SalesData_Remote_H2\$VT".	52 Seconds	480,193	SELECT "SalesData_Remote_H1\$VT"."CALDAY", "SalesData_Remote_H1\$VT".																																																
Statement Runtime	Rows	Remote SQL Statement																																																																		
1 Minutes	968,760	SELECT "SalesData_Remote_H2\$VT"."CALDAY", "SalesData_Remote_H2\$VT".																																																																		
1 Minutes	955,577	SELECT "SalesData_Remote_H1\$VT"."CALDAY", "SalesData_Remote_H1\$VT".																																																																		
Statement Runtime	Rows	Remote SQL Statement																																																																		
53 Seconds	488,152	SELECT "SalesData_Remote_H2\$VT"."CALDAY", "SalesData_Remote_H2\$VT".																																																																		
52 Seconds	480,193	SELECT "SalesData_Remote_H1\$VT"."CALDAY", "SalesData_Remote_H1\$VT".																																																																		

Figure 62: Scenario Comparison

Considering the size and complexity of the scenario this has already a huge impact. Especially when source data volume growth or the complexity of the scenario increases it is important to only read the data which is really needed in an optimal way.

7.1.2.3 Analyzing SQL Execution with the Plan Visualization (PlanViz)

To help you understand and analyze the execution plan of an SQL statement, you can generate a graphical view of the plan.

To visualize the plan, you can enter the statement in the SQL console and choose 'Generate SQL Analyzer Plan File'. You need to specify a file name, run the query and download the file afterwards. Contrary to the Plan Explanation the Plan Visualization executes the query and does not only generate the execution plan. Therefore, it is important to only use the SELECT Statement and not the INSERT INTO statement. Additionally, this plan is only helpful if the query runs successfully and does not end in out of memory situation.

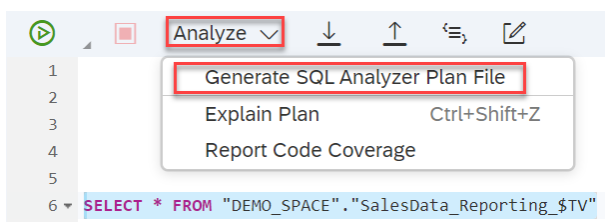


Figure 63: Generate SQL Analyzer Plan File

After downloading the *.plv file you can either use Visual Studio Code with the 'SQL Analyzer Tool for SAP HANA' Extension or a HANA Studio to open the file.

For the walk-through I will use the previous Datasphere View which I have used in the Plan Explanation scenario but without any filter on 'SALESORG'. To visualize the plan, I will use the SQL Analyzer Extension within Visual Studio Code.

The Datasphere View without any filter on 'SALESORG' loads ~ 60 million records, takes around 7 minutes and consumes about 120 GB of memory.

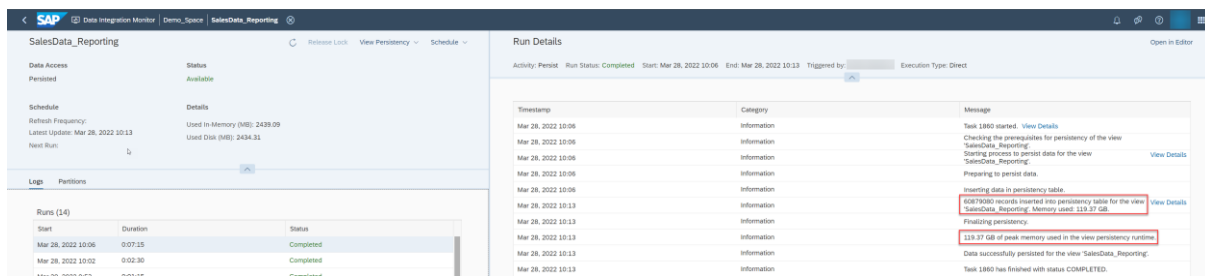


Figure 64: Monitoring

By checking the Plan Visualization without the INSERT INTO part, we can check how long the SELECT from the Datasphere View takes. There is anyway no chance from Datasphere side to improve the INSERT performance on the underlying HANA database. While comparing the runtime of the Plan Visualization and the runtime from the Datasphere View persistency monitor it is already possible to roughly estimate how much time is spent for writing the data into the target table. For our Datasphere Demo View we see that writing the data (60 million records) roughly takes five minutes (seven minutes end to end time from Datasphere View Persistency Monitor – 2 minutes for the SELECT statement from the Plan Visualization).

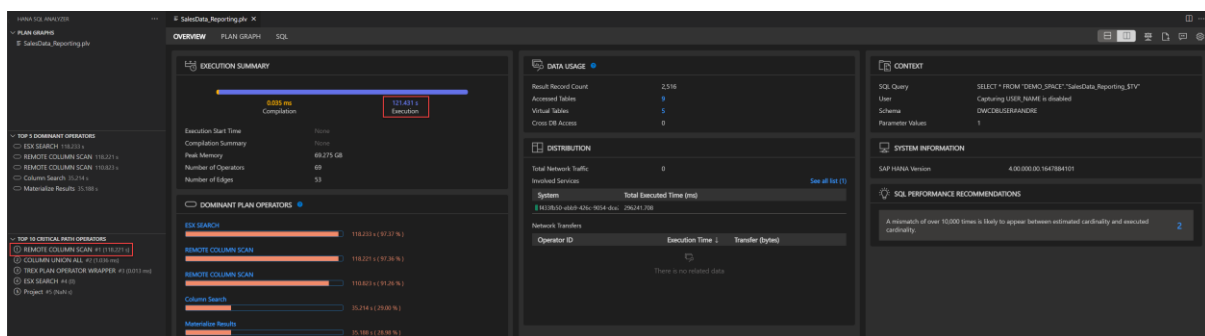


Figure 65: Visual Studio Code

Let's focus on the Plan Visualization itself. After opening the PLV-file with the SQL Analyzer there are already several sections which give a high-level overview. The 'Execution Summary' shows the 'Compilation' and 'Execution' time. On the left side there are two section which list the 'Top 5 dominant operators' and 'Top 10 critical path operations'. Those already give a good starting point in analyzing the Plan. Here we can already see that the main driver of this statement is a remote column scan which already consumes 118 of 121 seconds.

The Plan Graph visualizes the physical execution plan. It shows all plan operations with inclusive time (time spent up to this point) and exclusive time (runtime of this dedicated operator). The graphical structures also give information on what steps run in parallel and what operation needs to finish before others can start.

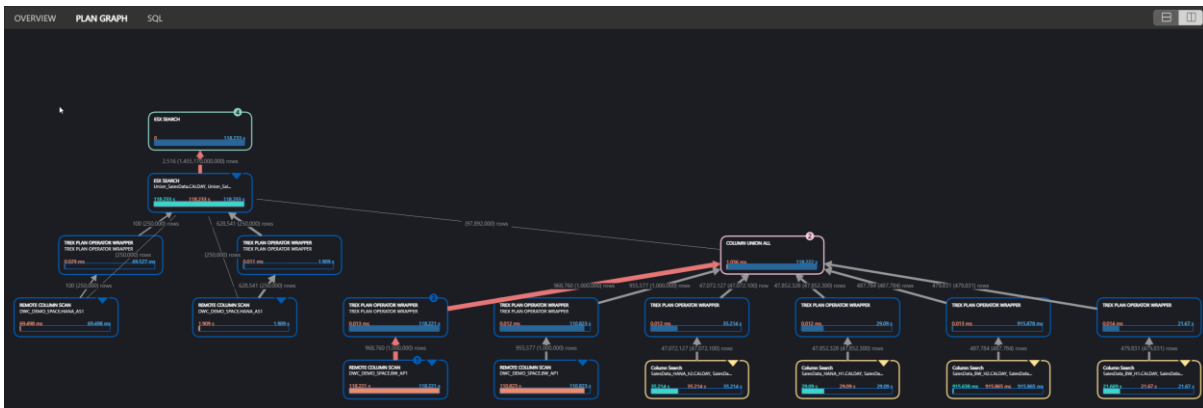


Figure 66: Visual Studio Code Plan Graph

Within the Union All operator we can see that there are six table searches. Four of them are on local Datasphere tables and two are on remote tables. The main runtime driver are the two remote column scans running in parallel with 118 seconds (inclusive time) each. Two further remote tables scans (left side of the plan) are fast and also the join within the ESX Search are no runtime factors in this demo scenario.

At the end the complete runtime of this Datasphere View Persistency is either spent in reading data from remote sources or writing data into the target tables. There are no expensive Union or Joins involved in this scenario. Hence, optimization within the Datasphere View itself are not possible. But since the Datasphere View persists 60 million records it is memory intensive. Therefore, considering physical or semantic partitioning are options to reduce the peak memory usage of this Datasphere View Persistency to safeguard system stability.

In our scenario we persist data of a complete calendar year coming from different sources(local/remote). From a data distribution we know that roughly every month has a similar data volume. Therefore, using partitions on Calendar Month for the Demo Datasphere View Persistency is a good option to reduce the peak memory consumption. You can check the [SAP help](#) on how to create partitions for Datasphere Views.

I have created 13 partitions (one per month and one 'other') partition on the field Month within the Datasphere View Persistency Monitor and started a new snapshot. After changing to partitioned loading Datasphere will trigger multiple INSERT INTO statements (one per partition) instead of one statement. As a result, you can see that the run details show the information per partition. The runtime of the view persistency was similar, and the number of persisted records is identical. But the peak memory usage of the Datasphere persistency reduced from ~120 GB (without partitions) to 11,5GB (with partitions).

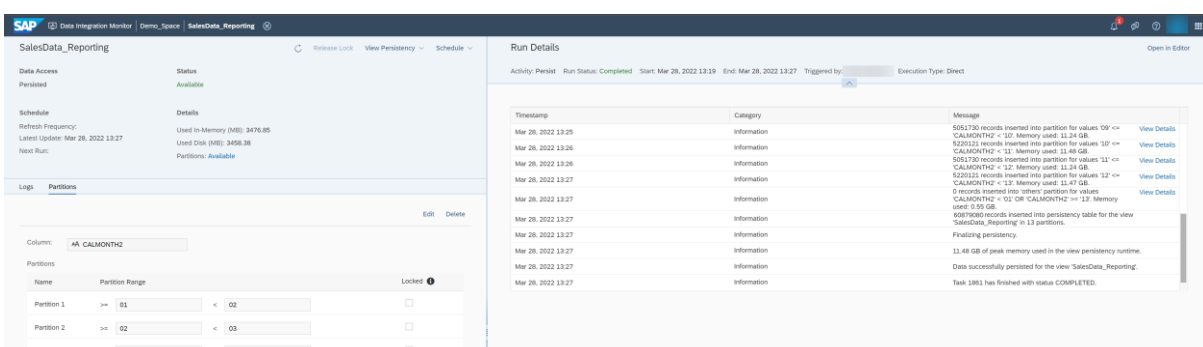


Figure 67: Monitoring

When working with partitions it is important to verify that the filter criteria which will be created and added as a where clause to the INSERT INTO statement is applied to the source fact tables to reduce the number of processed records per partition as much as possible. When this is not possible then working with partitions within the Datasphere View Persistency will only have limited or no impact on the memory usage and could

even result in a much longer runtime. It is also beneficial to choose a partition criteria and partition ranges which will result in similar partition sizes (number of records) based on the overall data volume.

It is possible to use Plan Explanation or Plan Visualization to check if the filter is applied correctly. Therefore, get one of the statements from the Run Details (view Details) in the View Persistency Monitor.

View Details

```
INSERT INTO "DEMO_SPACE"."SalesData_Reporting_$TRT" (SELECT * FROM "DEMO_SPACE"."SalesData_Reporting_$TV" WHERE "CALMONTH2">='01' AND "CALMONTH2"<'02');
```

Copy Close

After creating a new Plan Visualization with the SELECT statement including the where-clause we can check the Plan Graph based on the partitioned loading and compare it with the previous one without partitioning. E.g., in column searches on local Datasphere tables you can see the applied filter and also the reduced record count.

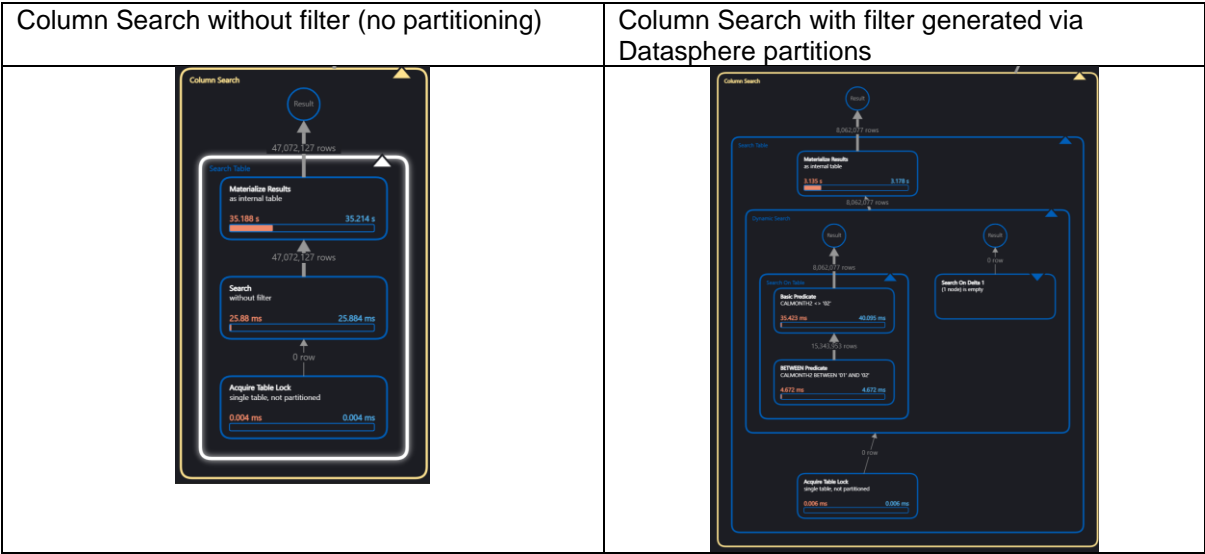


Figure 68: Comparison with and without partitions

While cross checking the remote statements in the Remote Query Monitor, we also see the filter based on the partition ranges for the relevant remote accesses.



Figure 69: SQL Statement

As mentioned above, the partitioned loading is not helpful for all Datasphere Views. There are also use-cases where it is not possible at all, or scenarios that do not benefit from it. It might also be the case that the peak memory consumption remains the same, but the runtime increases a lot. Therefore, it is important to analyze and validate the impact.

If the physical Datasphere partitions do not help it might be of relevance to investigate semantic partitioning of the Datasphere Sales View / View Stack. Please check next chapter for semantic partitioning.

7.1.3 Optimization Possibilities for Datasphere View Persistency

Create Statistics for remote tables

Using physical Partitioning for View Persistency

Using semantic Partitioning for View Persistency

Vertical Semantic Partitioning

Horizontal Semantic Partitioning

Optimize Remote Table access

This chapter summarizes possibilities to optimize runtime and/or memory usage of a Datasphere View Persistency. Many of them have already been addressed in the previous chapter either as part of Plan Explanation or Plan Visualization examples. Thus, this chapter serves as an overview of the Datasphere View Persistency optimizations currently available.

7.1.3.1 Create Statistics for remote tables

SAP Datasphere is using SAP HANA SDI and SDA to connect to remote tables. Tables and/or Views from remote sources are made available in Datasphere via virtual tables which get created when deploying a remote table. To ensure the best possible performance when accessing your remote tables and provide efficient query execution plans, you can create statistics for your remote tables. SAP Datasphere uses the SAP HANA SQL Optimizer that relies on accurate data statistics for virtual tables in SAP HANA and help you take decisions.

Statistics can be created only for remote tables within the Remote Table Monitor Detail Screen. There are three different kinds of statistics available.

Statistic Type	Description
RECORD COUNT	Creates a data statistics object that helps the query optimizer calculate the number of records (rows) in a table data source. The RECORD COUNT type is a table-wide statistic.
SIMPLE	Creates a data statistics object that helps the query optimizer calculate basic statistics, such as min, max, null count, count, and distinct count.
HISTOGRAM	Creates a data statistics object that helps the query optimizer estimate the data distribution.

It should be noted that remote statistics cannot be created for all adapters and connections. Besides this the creation of statistics can cause significant workload on the source system. RECORD COUNT is the simplest statistic object and complexity and workload on the source system increases for SIMPLE and HISTOGRAM. Please also consider updating the statistics from time to time when you get aware that based on the chosen statistic type the data volume, distribution of data, etc. changes on the source system.

For more information, see [Creating Statistics for Your Remote Tables](#).

7.1.3.2 Using physical Partitioning for Datasphere View Persistency

For persisting Datasphere views with larger data volumes it can be helpful to create physical database partitions to reduce the peak memory usage and therefore avoiding out-of-memory situations.

Partitions can be created from the View Persistency Monitor – Details Screen. Here it is possible to define a column from which the partition will be defined. After selecting a column, the number of ranges is determined

via a range selection. It is important to have in-depth knowledge about the data and data distribution especially for the selected partition column to define meaningful and equal-sized partitions.

As a result, and based on the specified partitioning, Datasphere will now sequentially start the INSERT INTO statement that receives a where clause based on the defined ranges and writes the data into a defined database partition. The number of statements is derived by the number of partitions.

In the chapter [Analyzing SQL Execution with the Plan Visualization \(PlanViz\)](#) within this document it is explained how to verify the impacts of the partitioned loading and to verify if the defined partitions lead to the expected result.

Please keep in mind that using partitions especially in combination with remote sources (because not all adapters can push down filters) or Datasphere SQL Script Views (could block filter push-down) could also result in much longer runtimes and memory usage. Therefore, it is important to check via Datasphere View Persistency monitor statistics and/or Plan Explanation/Plan Visualization that the result with defined partitioning does not end up being the opposite of the expectation.

When working with partitions you can check also the lock option when you don't want to update one or multiple partition(s) in the next run. For example, you have partitioned by month and you are aware that after a certain period of time data of historical months do not change anymore you can lock those partitions to avoid unnecessary data load.

For more information, see [Creating Partitions for Your Persisted Views](#).

7.1.3.3 Using semantic Partitioning for Datasphere View Persistency

It is not always possible to work with physical database partitions/partitioned loading ([Creating Partitions for Your Persisted Views](#)) for all kinds of Datasphere Views. Sometimes the design or the complexity of the Datasphere View or Datasphere View Stack will not allow the usage or will not lead to the expected improvement.

In this case it makes sense to think about cutting the Datasphere View Persistency into smaller pieces. There are two approaches to consider. On the one hand there is the vertical semantic partitioning and on the other hand the horizontal semantic partitioning.

7.1.3.3.1 Vertical semantic partitioning

The vertical semantic partitioning will result in multiple similar Datasphere Views with distinct filters. If you are familiar with SAP BW or BW/4HANA you have usually taken a similar approach when you expected high data volumes or data volume growth over years e.g., with multiple aDSOs for different years. The same approach is also possible with Datasphere Views. To avoid that too many records are persisted within one Datasphere View Persistency or when data volume will constantly increase over time it might be a good way to create e.g., one Datasphere View per year (or any other suitable characteristic) and restrict the sources accordingly. Those multiple Datasphere Views (e.g., one per year) can be unioned again in a new Datasphere View which serves as basis for SAC or third-party clients.

With this approach, the data volume can be distributed over several Datasphere View Persistencies and thus the peak memory usage can be reduced.

7.1.3.3.2 Horizontal semantic partitioning

Besides the vertical semantic partitioning, the horizontal semantic partitioning could also be an option to reduce the peak memory usage of Datasphere View Persistency runs. Depending on the Datasphere View / Datasphere View Stack this can easily be achieved with additional View Persistencies in underlying levels.

If the Datasphere View you want to persist has multiple levels containing other Datasphere Views as sources, it might be beneficial to already persist one or multiple of those low-level views, to reduce the peak memory usage of the top-level view. If the high peak memory can already be observed within one

Datasphere View you could think about splitting the logic into multiple Datasphere Views that build on each other to persist intermediate steps.

As can be seen, depending on the scenario, the horizontal semantic partitioning could be easy to implement and test. The only disadvantage of introducing additional layers might be the increased memory footprint on the underlying SAP HANA database for storing those data/layers.

There are several ways to partition Datasphere Views physically and semantically. Each of the individual approaches described above can already help, but of course they can also be flexibly combined with one another.

7.1.3.4 Optimize Remote Tables access

Remote Table access can sometime be really time consuming or memory intensive because remote data needs to be stored in an uncompressed way into intermediate result tables. Therefore, it is important to only read the data that is really required to persist the Datasphere View. Adapter and Connections offer different types of optimizations. Some of the connections offer filter pushdown (e.g., BW ABAP, HANA, HANA Cloud, Microsoft SQL Server, etc.) others, for instance, do not support the push down of filters (e.g., PostgreSQL, Generic JDBC, etc.). But also, those connections that allow filter push down do not allow every kind of filter push-down. If there are data type mismatches, complex filters, long in-lists, etc. the adapter or even the SAP HANA SQL Optimizer could decide to not push down the filter.

In addition to the filter push down some connections also offer the possibility of join relocation (e.g., HANA, HANA Cloud, etc.). Modeled joins on remote tables from the same remote source can be pushed down from the HANA Optimizer directly to the remote source to reduce the transferred data volume.

The Remote Query Monitor can be used to evaluate the statement(s) created from the Datasphere View Persistency to the remote source(s). Within the View details it can be checked if required or expected filters are included and pushed down or if a join will be executed on the remote source.

When the remote access seems to be the bottleneck with regards to runtime and/or memory usage of your Datasphere View Persistency you can think about replicating the remote table to Datasphere. To reduce the memory impact of a replicated table on Datasphere you can store the data in NSE (native storage extension) which is anyway the default setting for replicated remote tables.

7.2 Performance Root Cause Analysis for Data Flows

This section aims to provide some tools for how to analyse potential root causes for Data Flows with performance issues.

Data Flows are designed in the Data Builder in the Datasphere tenant. During connection to a source system, the meta data of the sources which are enabled for Data Flow modelling gets stored in the Datasphere HANA Database repository. During execution, the request is routed through a Data Intelligence Proxy (DI Proxy) to the Data Intelligence Embedded (DI Embedded) Kubernetes cluster. Inside the DI Embedded server, the request is turned into a graph by the pipeline and executed.

Due to the execution of Data Flows being handled by the DI Embedded server, it can be more challenging to find the root cause of performance related issues than for other Datasphere objects. In this section, we will try to give you some tools for narrowing down the root cause.

7.2.1 Data Integration Monitor

A good starting point is the Data Integration Monitor (Figure 70).

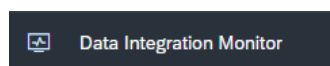


Figure 70: Data Integration Monitor in menu bar

In the Data Integration Monitor both manually executed and scheduled Data Flow runs can be seen under the tab *Data Flow Monitor* (Figure 71).

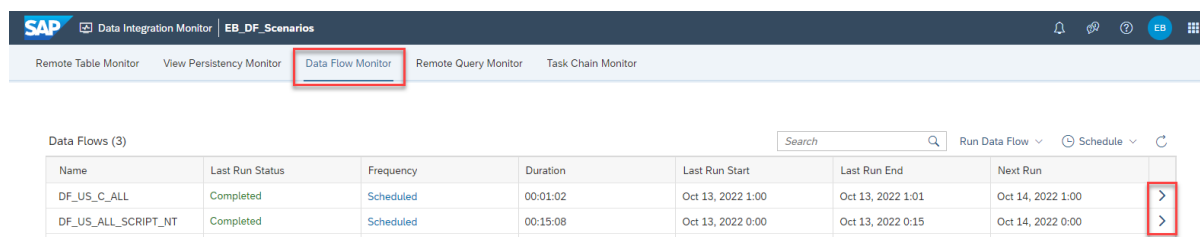
A screenshot of the SAP Data Integration Monitor interface. The 'Data Flow Monitor' tab is selected and highlighted with a red box. Below the tab, there is a table with columns: Name, Last Run Status, Frequency, Duration, Last Run Start, Last Run End, and Next Run. The table contains two rows of data. The first row is for 'DF_US_C_ALL' with status 'Completed'. The second row is for 'DF_US_ALL_SCRIPT_NT' with status 'Scheduled'. Both rows have a red box around the rightmost cell, which contains a right-pointing arrow. Above the table, there is a search bar and some filters like 'Run Data Flow' and 'Schedule'.

Figure 71: Data Integration Monitor

In the initial overview in the screenshot above, you can see if there were any failed runs in the column *Last Run Status*. The Data Flow will have either the status *Completed* or *Failed*. You also see the runtime for the last executed Data Flow in the column *Duration*.

To see the details of a Data Flow, click into the run details with the arrow on the right side of that row in the run overview.

Run Details

On the Run Details page (Figure 72), you can see a list of all the latest runs, along with their status on the left side of the screen. Per default, the details of the latest run are displayed on the right side of the screen. If you would like to see the details of a previous run, select it from the *Runs* list on the left.

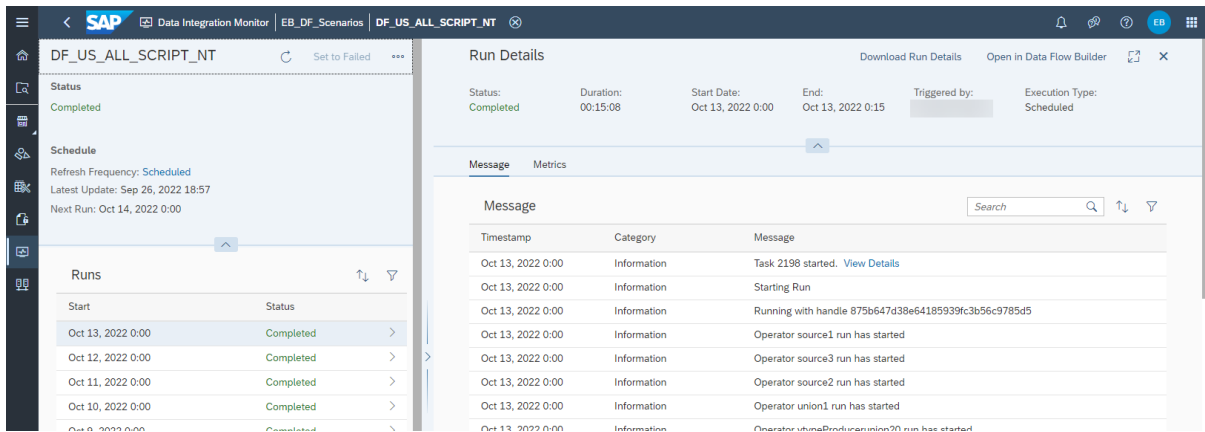


Figure 72: Run Details page - list of all the latest runs

On the right, in the *Run Details* overview (Figure 73), you can already see the duration of that run at the top. If you would like to see further details, many of the individual steps of the run have the option to open a context box with further details. If you scroll to the end of the list of messages, you find two entries called *Data flow run has finished*.

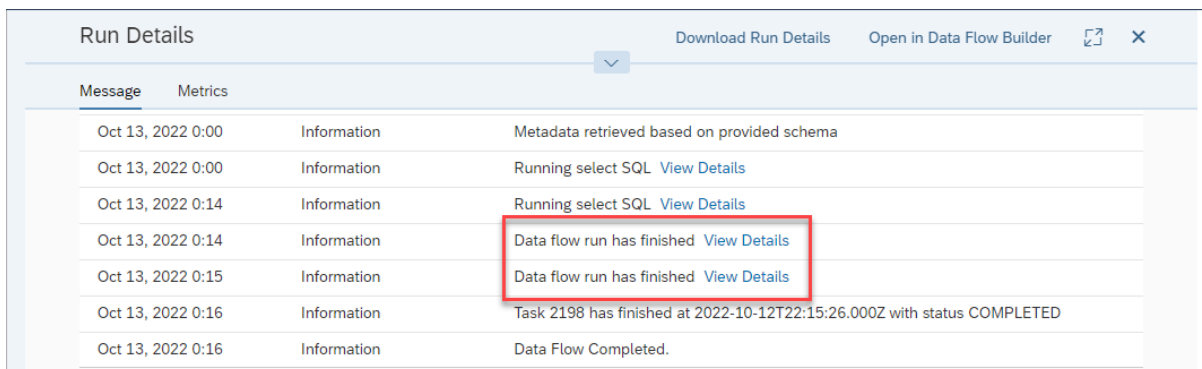


Figure 73: Individual Run Details (1)

When you click on *View Details*, you can see information about the runtime of your Data Flow, and how many rows have been written into your target table (Figure 74/Figure 75).

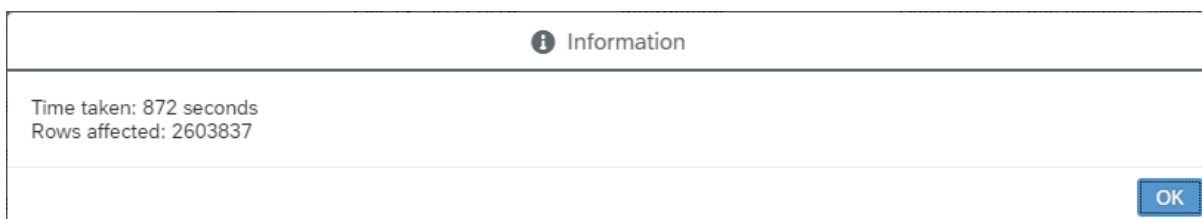


Figure 74: Individual Run Details (2)

Download Run Details

Another option in the Data Flow Monitor details screen is to download the run details in the form of a JSON file by clicking *Download Run Details* at the top right of the screen for Run Details (Figure 75).

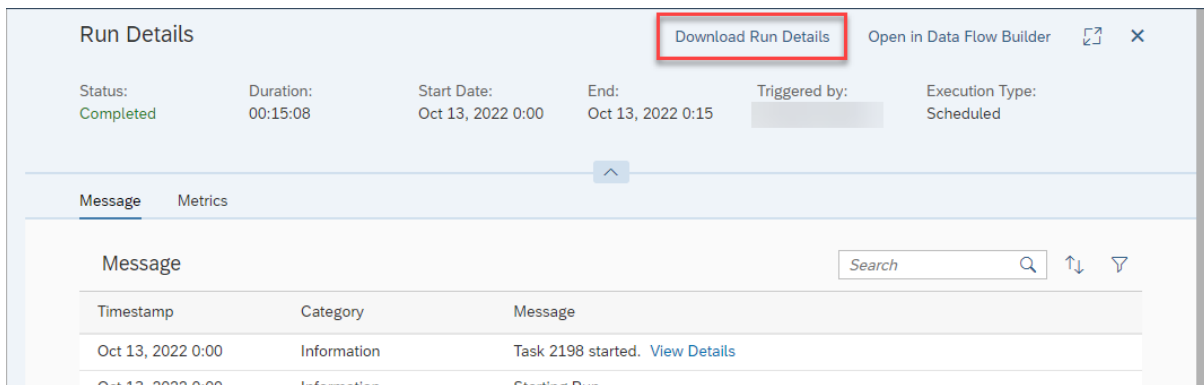


Figure 75: Download Run Details

This file is in JSON format, and does not contain any timestamps, so in the context of analysing performance issues, it will only help you gain an overview of the execution process of your Data Flow, and not where most of the time is spent. This can be useful if you download the details of a Data Flow run before making any changes. Should your changes to the Data Flow result in performance issues, you can compare the two JSON files and see which changes were made. Please note, that because the Run Details file is only available in the system for 3 days, you need to manually download this file before making changes if you want to have it available for comparison.

As well, if you want to see the design of the Data Flow model behind the poorly performing run, the button *Open in Data Flow Builder* next to the *Download Run Details* will take you to the model in the Data Builder.

Run with Checkup

Back on the left side of the details screen, you also have the option of running the Data Flow directly, with a checkpoint option. To do this, either click on Run Data Flow and choose *Run with Checkup* (Figure 76) or, if you do not see this option, click the ellipsis to find it.

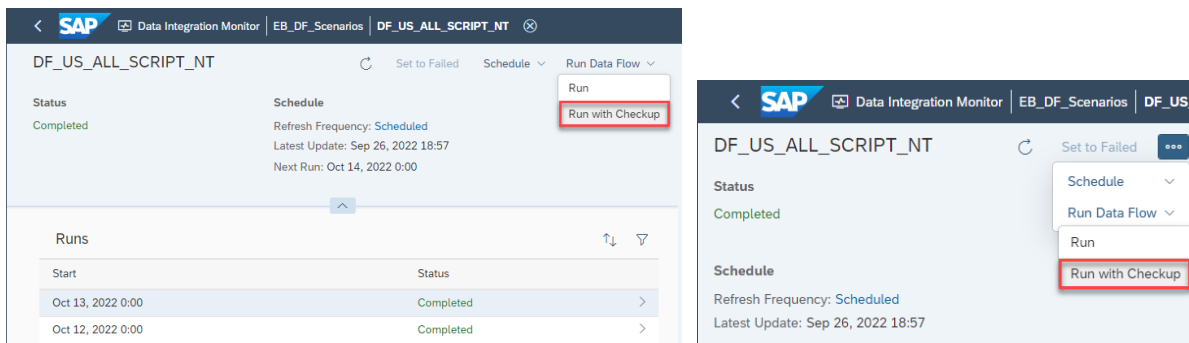


Figure 76: Run with Checkup

This will generate a detailed log of the run, which is currently not publicly available. In order to get this file, you will need to contact SAP through a ticket or direct contact in the context of a support service.

7.2.2 Checking Network Activity

in Google's Chrome browser. If you are not familiar with the developer tools, you can access them in the browser from the ellipsis symbol on the top right of the window and navigate to them over *More tools* and then choosing *Developer tools* (Figure 77).

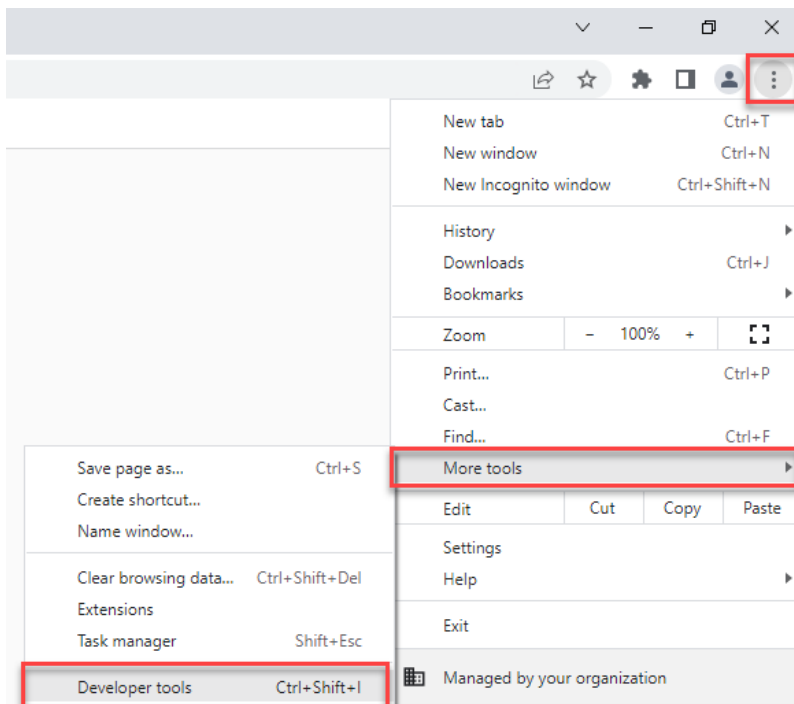


Figure 77: Chrome Developer tools

You can also use the shortcut displayed to access the tool faster in the future, but this will depend on your operating system, so check what the shortcut is on your device.

Once you access the tools, you get a separate screen for them, which you can place in a separate window, or on the right, left or bottom of your current screen. For more information on how to use the tools in general, refer to Google's own [documentation](#).

For the purpose of checking the Data Flow network activity, we focus on the *Network* tab in the tools (Figure 78).

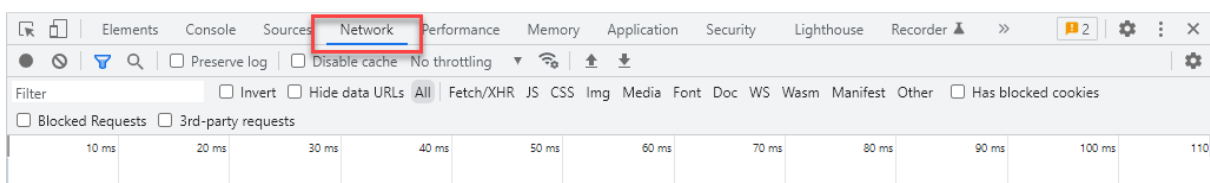


Figure 78: Chrome Developer tools - Network

When you open the Developer tools, they will be recording per default (Figure 79). Start by 1) turning off recording, 2) clearing any requests, and 3) clearing the console:

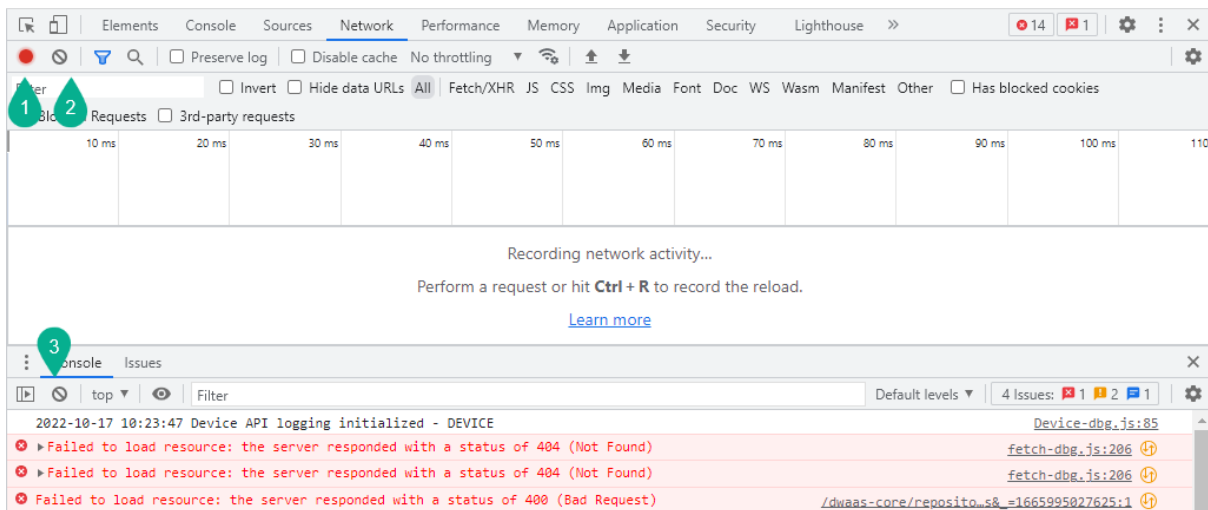


Figure 79: Chrome Developer tools – Network details

We do this to ensure that we capture just the network traffic related to our Data Flow execution. Afterwards, it should look something like this (Figure 80):

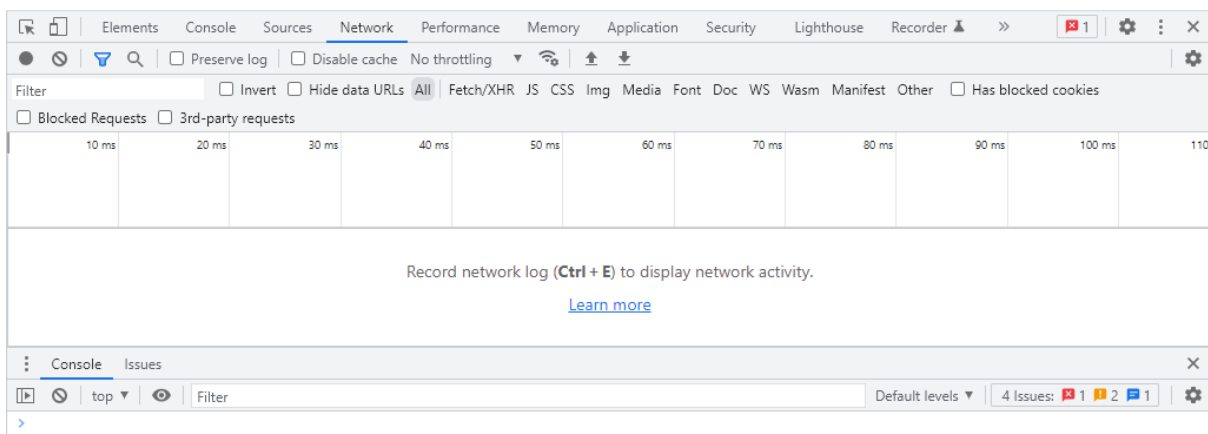


Figure 80: Clearing the Console

You can manually run the Data Flow either from the Data Flow editor, or from the Data Integration Monitor. Start by turning the recording back on (step 1 from above), then run the Data Flow with the performance issue, and turn the recording off again once the Data Flow has finished. Before analysing further, you have the option to export the files captured, so you can look at them again later, share them, etc. You do this by clicking on the *Export HAR* button (Figure 81).

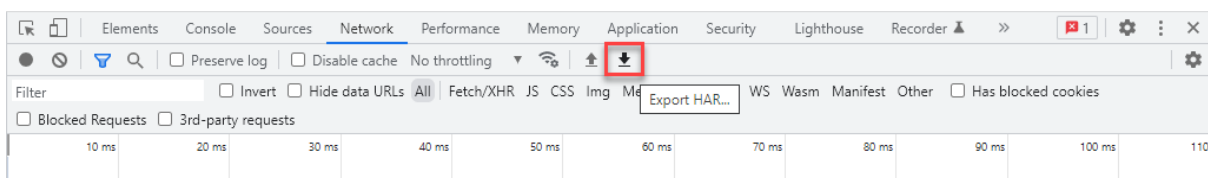
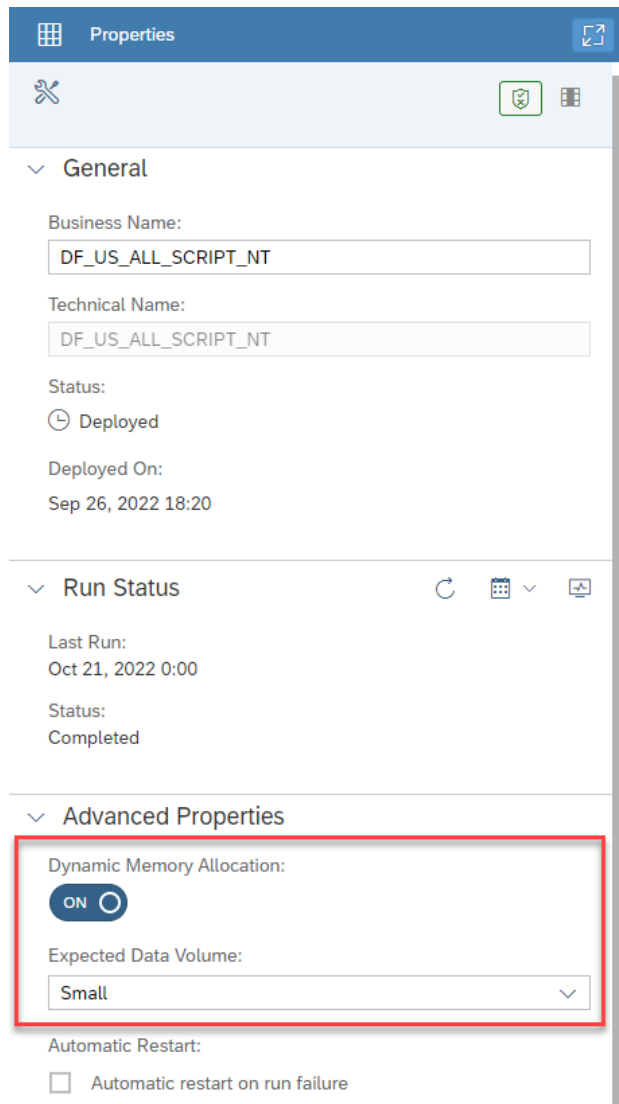


Figure 81: Export HAR File

7.2.3 Potential Performance Improvements

If you are facing performance issues with your Data Flows, it can sometimes be a good idea to return to the model and analyse if it is possible to split the Data Flow up into smaller individual flows, and have them run in sequence using a Task Chain.

If you experience performance issues that lead to out of memory situations, you can use the *Dynamic Memory Allocation* feature, which you enable in the Properties of your Data Flow (Figure 82). Specify here whether the expected data volume is small, medium, or large.



The screenshot displays the 'Properties' dialog for a Data Flow. The 'General' section includes fields for 'Business Name' and 'Technical Name', both set to 'DF_US_ALL_SCRIPT_NT', and a 'Status' of 'Deployed'. The 'Run Status' section shows the 'Last Run' as 'Oct 21, 2022 0:00' and 'Status' as 'Completed'. The 'Advanced Properties' section, highlighted with a red box, shows 'Dynamic Memory Allocation' set to 'ON' and 'Expected Data Volume' set to 'Small'. Below this, the 'Automatic Restart' section has an unchecked checkbox for 'Automatic restart on run failure'.

Figure 82: Data Flow Advanced Properties

Please be aware that this is recommended specifically for situations where you are facing out of memory failures (Figure 83):

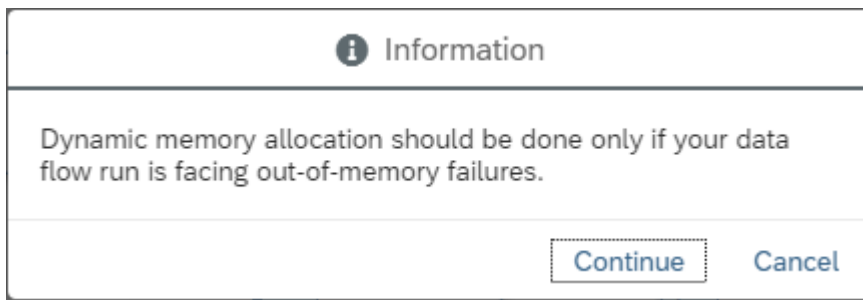


Figure 83: Dynamic Memory Allocation

8 Additional Information

Change History

Version 3.0

New	Landscape Chapter added
Update	Spaces chapter updated added new SAP BW bridge section
Update	Major updates in the data modeling chapter, especially the layered modelling approach
New	Performance Root Cause Analysis for Data Flows

Version 2.0

Update	Major updates in all areas of the document
Update	More details on the layered modelling approach
New	Added chapters on performance considerations and performance analysis

Version 1.0

New	Initial Version
-----	-----------------

Find more information and related blog posts on [the topic page for SAP Datasphere](#). You will find further product information on our Community with various subpages about [Getting Started](#), [Business Content](#), the [SAP BW Bridge](#) as well as content for [Best Practices & Troubleshooting](#) and the [FAQ for SAP Datasphere](#).

www.sap.com.

