**Распределение памяти**

− Есть ли возможность ограничить кол-во памяти для режима(сесии)? Развелось немало программок которые смело кушают по 3-5 гигов оперативки (**sm0**4). Хотелось бы выставить параметр в профайле чтоб такие проги выпадали с дампом по постижении некого установленного лимита.

− Думаю изменение **ztta/roll\_extension** /максимум EM выделяемой пользователю на процесс/ и **abap/heap\_area\_dia** /максимум кучи/ поможет.Необходимо, что бы в сумме по обоим параметрам, набиралось максимально желаемое выделяемое пользователю кол-во памяти. У пользователя при превышении будет дамп.

**Параметры, влияющие на скорость обращения к объектам BW**

* ztta/roll\_extension
* abap/heap\_area\_total
* abap/heap\_area\_dia
* em/initial\_size\_MB

# [BWonHANA: Large Data Volumes and Partitioning](https://wiki.scn.sap.com/wiki/display/BI/BWonHANA%3A+Large+Data+Volumes+and+Partitioning)

<https://wiki.scn.sap.com/wiki/display/BI/BWonHANA%3A+Large+Data+Volumes+and+Partitioning>

# Optimizing models in BW/4HANA mixed scenarios

<https://blogs.sap.com/2020/03/04/optimizing-models-in-bw-4hana-mixed-scenarios/>

## **ADSO partitioning - physical and logical /with semantic group/.**

Partitioning only works when the ADSO has more than 2 billion of records – for ADSO with less of 2⋅1012, ***physical partition*** can be harmful. ***Logical partition*** with semantic group is neutral.

Removing historical data from these ADSO didn’t have relevant improvement - we removed 20% of historical data in these ADSO and only obtained 2-5% of gain in performance.

We had some problems with the remodelation process of big ADSO. Finally we decided to do first a copy of them, drop the data in the ADSO, to do the partition and reload from the copy.

## **Add input-parameters in HANA views**

Changing the model by adding a input parameter instead a query-filter, we had about **5-20%** of improvement.

## **Join on Key columns and indexed columns**

SAP recommends that all relevant joins should be done using the key columns or indexed columns /*не всегда*/.

Rem

We checked all the mains joins and added indexes where needed - in most cases *the gain wasn’t relevant or even a little bit worst*.

Only in one case, where some fields were used in several joins in a view, we gained about **17%** of performance.

## **Optimize joins columns flag in HANA view**

## ***Optimize joins columns*** flag in HANA view - after activate this flag, we gained about **50%** in performance.

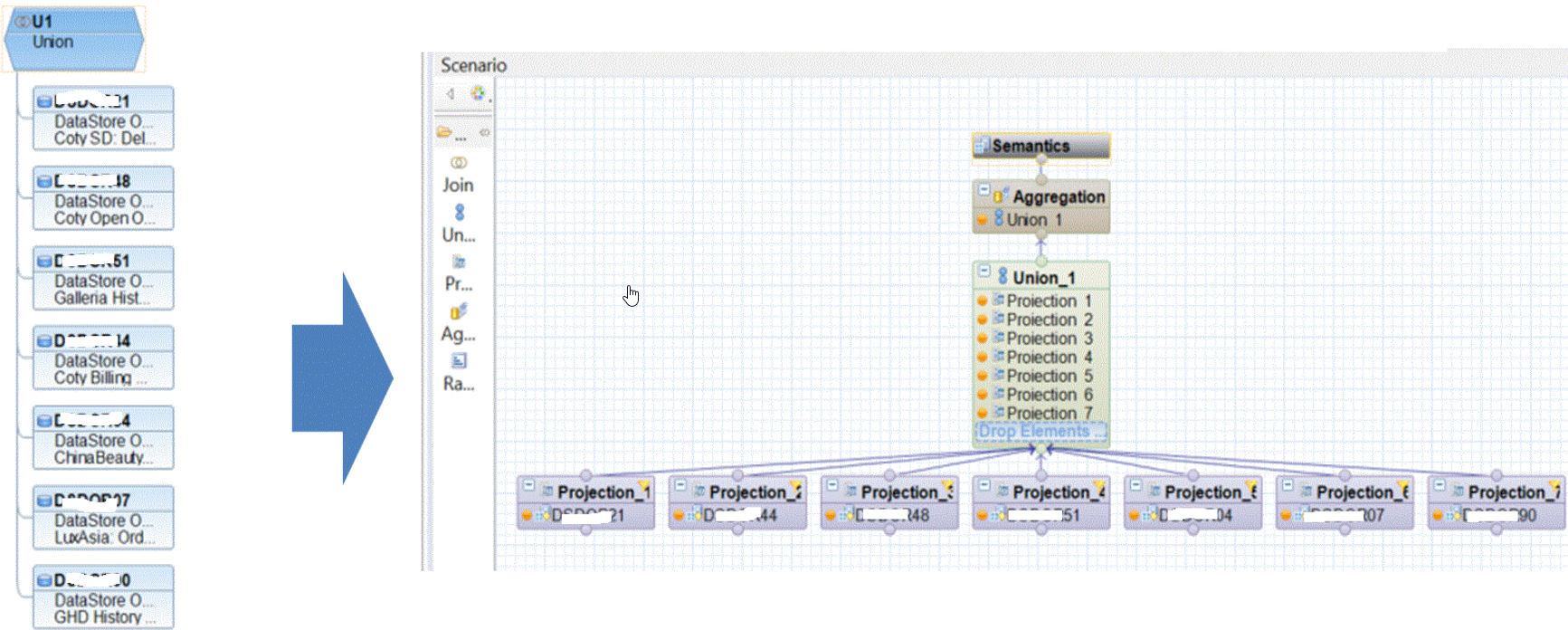
Rem

# Optimize Join Columns Flag

<https://blogs.sap.com/2018/08/10/optimize-join-columns-flag/>

## **Replace composite unions by HANA unions in top**

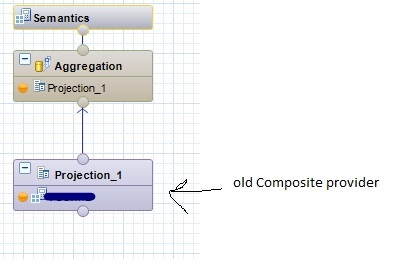
If you have a composite provider with unions, is good idea to replace this BW-unions by HANA-unions.



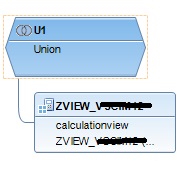
## **Navigational attributes**

Rremap the navigational attributes on their own infoobject /*не всегда*/.

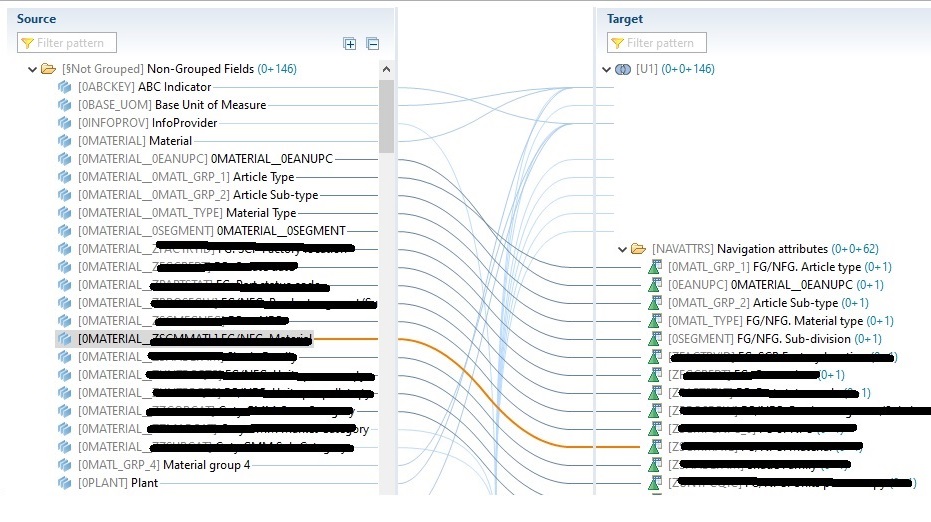
1. Create a new HANA view using the old CP in a projection



1. Create a new CP using this new hana – view



1. Remap the navigational attributes on their own infoobject



In this way, depending on the number of navigational attributes used in the query, *the improvement was about* ***25%-45%*.**

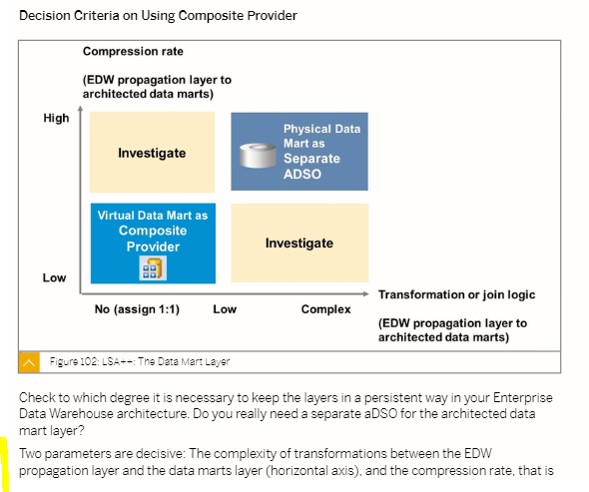
Rem

In some drill-down or filter selections, the performance is worse than in the old model - I think the *problem is with the composite provider behind the hana-view*, but this doesn’t invalidate the solution commented.

## **Purge** /pə:dʒ чистка/ **historical master data**

The point here is that of this 45 millions of historical master data changes rows, only 8 millions of rows were being joined with the transactional data /because we don’t keep all the historical data in the transactional data/. After purge the master data not used, the improvement was about **55%**.

## **Materialize the joins**



In one area, where the EDW layer has billions of rows with a high granularity, this approach must be considered in future projects.

## **Left joins vs Inner joins**

SAP recommends whenever possible, make inner joins instead of left joins /*спорно*/.

Rem

I’m not sure about this - we changed some left join by inner joins, without relevant improvements.

**ADSO logical partitioning**

### **Logical partitioning**

The InfoProvider is split to several smaller BW objects. In comparison to database partitioning, here you do not have any limitation for the characteristic by which you want to split. A typical example for such characteristics would be **0CALYEAR** or **0COMPCODE**.

First, the reports and applications become significantly faster because the system can execute several database requests simultaneously. Second, the model is much easier and nicer for administration purposes.

Semantically Partitioned Object **SPO** is an InfoProvider that consists of several InfoCubes/DataStore objects with the same structure. Semantic partitioning in the context of SPO is a property of the InfoProvider, which is specified upon creation.

− We will need to Partition our ADSOs very soon. Does anyone know if this needs to be configured for the ADSO in Eclipse and then transported to each System? Or manually done in each system?

− SAP note [2374652 - Handling very large data volumes with advanced DataStore objects in SAP BW on SAP HANA and BW/4HANA](https://launchpad.support.sap.com/#/notes/2374652)