SAP Merchandise and Assortment Planning on BW Integrated Planning

Best Practice Guide for Performance tuning

Version 1.01
November 2009
**Document History**

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>First official release of this document</td>
</tr>
<tr>
<td>1.01</td>
<td>Examples for query properties corrected (chapter 9.1)</td>
</tr>
</tbody>
</table>

**Applicable Releases**

This Best Practice Guide is based on internal tests in a system on SAP NetWeaver 7.01 with support package stack 04. The recommendations should also apply to SAP NetWeaver 7.00 with an up to date support package stack.

**Contacts**

**Georg Meier**  
Head of Competence Center MAP  
Field Services Trade Hub – Global  
SAP Deutschland AG & Co. KG  
Neue Bahnhofstraße 21  
D-66386 St. Ingbert, Germany  
T  +49 6227 7-71416  
F  +49 6227 78-023312  
M  +49 173 3090057  
mailto:georg.meier@sap.com

**Dominik Graus**  
Principal Consultant  
Field Services Trade Hub – Global  
SAP Deutschland AG & Co. KG  
Neue Bahnhofstraße 21  
66386 St. Ingbert, Germany  
T  +49 6227 7-71718  
F  +49 6227 78-28847  
M  +49 173 3090316  
mailto:dominik.graus@sap.com
# Table of Contents

1. Introduction ......................................................................................................................................................... 1
2. Hardware ............................................................................................................................................................... 2
3. BW Accelerator ................................................................................................................................................... 3
4. BW Architecture .................................................................................................................................................. 6
   4.1 Characteristics............................................................................................................................................... 6
   4.2 Compounded Characteristics ...................................................................................................................... 6
   4.3 Navigation Attributes ........................................................................................................................................ 7
   4.4 Key Figures........................................................................................................................................................ 9
   4.5 DataStore Objects .......................................................................................................................................... 9
   4.6 InfoCubes .......................................................................................................................................................... 10
5. BW Integrated Planning Architecture .............................................................................................................. 15
   5.1 Input-ready Query on Aggregation Level ................................................................................................... 15
   5.2 Input-ready Query on MultiProvider .......................................................................................................... 16
   5.3 Pros and Cons ................................................................................................................................................ 17
   5.4 Performance .................................................................................................................................................. 17
      5.4.1 Design of Aggregation Levels ................................................................................................................ 17
      5.4.2 Comparison of Planning Architectures ................................................................................................. 18
      5.4.3 Detailed Example .................................................................................................................................. 19
6. Characteristic Relationships ................................................................................................................................. 24
7. Variables ............................................................................................................................................................... 27
8. Hierarchies ............................................................................................................................................................. 28
   8.1 BW Hierarchy .................................................................................................................................................. 28
   8.2 Master Data Attribute driven Hierarchy ...................................................................................................... 28
   8.3 Display Hierarchy .......................................................................................................................................... 28
9. Query Design with BEx Query Designer ............................................................................................................. 30
   9.1 Query Properties ........................................................................................................................................... 30
   9.2 Filter ......................................................................................................................................................... 31
   9.3 Rows/Columns .............................................................................................................................................. 32
   9.4 Free Characteristics .................................................................................................................................... 32
   9.5 Key Figures ................................................................................................................................................... 32
   9.6 Structures .................................................................................................................................................... 32
   9.7 Frontend Disaggregation ............................................................................................................................ 33
   9.8 F4 Value Help ............................................................................................................................................ 33
   9.9 Inverse Formulas and Cell Locking ............................................................................................................ 33
10. Web Template Design with BEx Web Application Designer .......................................................................... 35
    10.1 Application Building ................................................................................................................................... 35
    10.2 Planning Functions ....................................................................................................................................... 35
    10.3 Save Button ................................................................................................................................................ 35
    10.4 Paging ...................................................................................................................................................... 35
    10.5 Flicker Free Rendering ................................................................................................................................ 36
11. Scalability of Planning Applications .................................................................................................................. 37
Appendix – List of Figures ........................................................................................................................................ 39
1 Introduction

SAP Merchandise and Assortment Planning (SAP MAP) delivers a fully integrated retail planning solution to bring together comprehensive real-time performance metrics with planning and simulation capabilities. The solution, based on SAP NetWeaver Business Warehouse and its planning component Integrated Planning (BW-IP), offers extensive planning functionalities, analysis and reporting options, for seasonal and non seasonal retailers.

Trade customers utilize SAP MAP to implement their retail and wholesale planning processes. Based on five different customer implementations, a worst-case scenario has been built, analyzed and optimized. The analysis included single and multi-user tests to prove performance and scalability of SAP MAP.

This document summarizes the lessons learned and provides guidance for best practices in SAP MAP implementations. For the most part, these recommendations are not specific to SAP MAP and apply to any Integrated Planning implementation project.

Please see following links to get more information all around SAP MAP, Workshop content/schedule and a process description.
SAP MAP Workshop
www.service.sap.com/retail -> "W26MIP SAP Merchandise and Assortment Planning
SAP MAP Process Description and Configuration Guide
www.help.sap.com -> SAP NetWeaver -> BI Content -> BI Content AddOn 4 -> Industry Solutions -> Trading Industries
 -> Retail Trade -> Merchandise and Assortment Planning

Use the information and links on this page to get an overview of SAP Business Intelligence features and settings that impact performance, and of the tools that help identify and address performance problems.
http://www.sdn.sap.com/irj/sdn/bi-performance-tuning
2 Hardware

In terms of required hardware resources, planning differs from pure reporting in a few aspects. *Reporting* typically involves a high portion of database activity – at least in a system landscape without BW Accelerator. Database reads lead to I/O accesses on the storage subsystem. Therefore, a high-performance storage subsystem is essential for short query runtimes.

Performance critical navigation steps in a *planning* query, like disaggregation, are processed in ABAP and therefore mainly consume processor computing cycles. Furthermore, disaggregation is processed sequentially in a single work process. Therefore, the speed of a single CPU core, i.e. its architecture and clock frequency, determines the runtime of ABAP-dominated sequential processing steps like disaggregation.

These considerations should be taken into account when the hardware sizing of a planning application is elaborated. It is not sufficient to ensure that the calculated amount of SAPS is provided by the hardware. Instead, it is important to prefer a system architecture with fast, high clock processors to an alternative system with a higher number of slower CPU cores.

This advice is not only valid for planning but also for reporting, especially in reporting scenarios with complex calculations (exception aggregation, Virtual InfoProviders, …). However, planning typically involves a higher portion of sequential ABAP processing as compared to reporting. Therefore, it is essential for short response times in planning applications to choose a powerful CPU architecture with high clocked processors.

This approach does not necessarily involve higher hardware costs. Keep in mind that short response times are beneficial in two ways: First, the user experiences a fast system with short runtimes for query execution and navigation. Second, a fast processor is freed up sooner and available to process other user requests.
The SAP NetWeaver BW Accelerator is based on TREX technology. It is packaged as an appliance for use with SAP NetWeaver BW and provides greatly enhanced performance for online analytic processing in an Enterprise Data Warehousing scenario.

This enhanced performance is delivered by clones of a TREX engine that run in parallel on the blade servers in the appliance hardware. The TREX engine indexes business data from InfoCubes and stores the indexes in an attached storage subsystem. The accelerator then loads these indexes into the main memory of the blade servers and uses them to answer analytic queries entirely in memory.

The BW Accelerator offers two main benefits:

- **Enhanced productivity**
  
  Response times delivered by the BW system for analytic queries are often much faster. Even for huge InfoCubes with billions of rows, results can be returned in seconds.

  The response times for repeated executions of a query remain stable even under high system load. By contrast, response times with aggregates often vary unpredictably, depending on whether an existing aggregate exactly matches a query or whether a database optimizer finds a perfect execution plan for the query.

- **Reduced TCO**

  The total cost of ownership of the BW system is reduced by introducing the accelerator compared with other strategies for boosting performance like aggregate tuning.

  The accelerator creates only one BIA index for each InfoCube, so there is no need to waste administration resources maintaining multiple aggregates. Because the definition and creation of aggregates is a task requiring a high skill level, and the ongoing task of maintaining them is laborious, the human resource costs for this approach to performance optimization are high. By contrast, both the initial indexing effort for an InfoCube and the subsequent maintenance effort for change runs and roll-ups are an order of magnitude lower.

In planning projects, there are often doubts whether BW Accelerator can also help to reduce runtimes of planning queries. The clear answer is: yes. Of course, it is possible to create BWA indexes for InfoCubes storing actual data. But you can also create BWA indexes for real-time InfoCubes used to store plan data.

To benefit optimally from BW Accelerator, it is recommended to roll-up new requests in real-time InfoCubes into the corresponding BWA index as soon as possible. For InfoCubes with actual data, the roll-up of new requests into the BWA index is typically scheduled as part of a process chain. New requests in real-time InfoCubes created by planning users can be rolled up automatically into the BWA index by setting the automatic request processing property 'Roll Up Data in Aggregates and/or BIA Index'. This enforces an automatic roll-up of any new request immediately after it has been closed by the system. Of course, it is also possible to roll-up new requests by scheduling the roll-up as batch job or within a process chain.

To activate automatic roll-up, go to Data Warehousing Workbench and right-click on the real-time InfoCube. Choose option 'Manage' and select the menu 'Environment' → 'Automatic Request Processing'.
Keep in mind that the roll-up of a new request into the BWA index is done synchronously when the user saves data and the threshold of 50,000 records is exceeded. This may lead to increased response times when saving data. On the other side, the roll-up of new requests into the BWA index is recommended to minimize the amount of data that must be read from the database.

Activating the automatic roll-up of new requests is only necessary and useful in planning scenarios with many concurrent users who create a high number of new requests per day. In such scenarios, it may also be useful to design the save button in the web template of the planning application as described in chapter 10.3. In scenarios with a small number of new data records per day, it is sufficient to roll-up new requests once per day within the batch schedule.

As part of the roll-up of a new request into a BWA index, the system invalidates the BWA index and merges the new data into the existing index. This step can take a few seconds which is acceptable for roll-ups in process chains but rather unfavorable for roll-ups that take place when a planning user saves data.
The roll-up runtime can be reduced by activating the so-called delta index. For InfoCubes with delta index, the BW Accelerator does not merge the new request into the existing index during the roll-up process. Instead, it keeps the main index and rolls up the new request into a so-called delta index. Main and delta index can be merged later, for example during the nightly batch window, by scheduling the report RSDDTREX_DELTAINDEX_MERGE.

For large real-time InfoCubes with automatic roll-up of new requests, it is recommended to activate the delta index. To do so, call transaction RSDDBIAMON and choose menu 'BI Accelerator' $\Rightarrow$ 'Index Settings' $\Rightarrow$ 'Set Delta Index'.

Do not forget to schedule the report RSDDTREX_DELTAINDEX_MERGE regularly to merge main and delta index.

The size of the delta index can be checked in transaction RSRV: 'Tests in Transaction RSRV' $\Rightarrow$ 'All Elementary Tests' $\Rightarrow$ 'BI Accelerator' $\Rightarrow$ 'BI Accelerator Performance Checks' $\Rightarrow$ 'Size of Delta-Index'. The repair mode of this check can also be used to merge main and delta index manually.

Please verify that your BW Accelerator is configured properly with regards to delta indexes. This implies that delta indexes are not being merged automatically by the TREX software. See SAP Note 1052146 for further details.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1052146</td>
<td>BIA 7.00: Do not use merge for delta index by memwatch</td>
</tr>
</tbody>
</table>
4 BW Architecture

This chapter shall provide general guidelines for modeling applications in SAP BW. Since IP is technically based on BW, those guidelines and rules also apply here.

4.1 Characteristics

Characteristics are sorting keys, such as article hierarchy level, style, store, fiscal year, period, or region. In an InfoCube, characteristics are grouped into dimensions. These dimensions are linked by dimension IDs (DIMID) to the key figures in the fact table. The characteristics determine the granularity (the degree of detail) at which the key figures are stored in the InfoCube.

Units are also required so that the values for the key figures have meanings.

Time characteristics are characteristics such as period (e.g. month, week, quarter), fiscal year, and so on.

4.2 Compounded Characteristics

Sometimes there is a need to compound InfoObjects in order to map the data model. Some InfoObjects cannot be defined uniquely without compounding.

Example: The ERP maintenance transaction for Retail articles (MM42) displays several views on an article: basic data, purchasing, sales, logistics store etc. For a given article, attributes on the logistics store view, like MRP profile or source of supply, may differ from store to store. They cannot be appended to InfoObject 0MATERIAL. Instead, a separate InfoObject with "key" article/store is required. This can be achieved by compounding InfoObjects. In BI Content, 0MAT_PLANT is compounded to 0PLANT and stores attributes like MRP profile and source of supply. In this example, 0MAT_PLANT is called compounded characteristic and 0PLANT is its compounding parent.

In MAP, compounded InfoObjects are also used to store valid characteristic combinations. For example, the InfoObject 0RT_STYCOCR is compounded to 0MATERIAL and stores the valid style/color combinations. It is not used as characteristic in InfoCubes, but only ready by the user exit coding of characteristic relationships.

Using compounded InfoObjects extensively in InfoProviders, particularly if you include a lot of InfoObjects in compounding, can have a negative impact on performance! A maximum of 13 characteristics can be compounded for an InfoObject. Note that characteristic values can have a maximum of 60 characters. This includes the concatenated value, meaning the total length of the characteristic(s) in compounding plus the length of the characteristic itself.

Furthermore, compounded characteristics can lead to so-called CMP problems in MultiProviders. Processing characteristics with CMP problems is very complicated and can have a negative impact on performance. A CMP problem occurs when the compounding parent of a compounded characteristic is identified in all part providers of the MultiProvider but the compounded characteristic itself is not identified by all part providers. CMP problems are displayed as warnings in the MultiProvider maintenance. To verify if a MultiProvider is affected by CMP problems, go to Data Warehousing Workbench, right-click on the MultiProvider, choose ‘Change’ and confirm the relevant Info-Providers. Then, select menu ‘MultiProvider’ → ‘Check’.

November 2009
For a new MultiProvider, this check is only available after the first save of the MultiProvider definition. The long text of the 'CMP problem' message contains very detailed information on the problem and offers options how to avoid it. In some scenarios, all data records in the InfoProvider may have the same value with respect to the compounding parent. Then, this InfoObject can be set to a constant value in the InfoProvider maintenance. This eliminates the CMP problem. The long text of the warning message explains further options to remove a CMP problem.

If possible, try to avoid MultiProviders with CMP problems. But also keep in mind that CMP problems do not necessarily lead to a negative impact on performance for any query defined on the MultiProvider. In most cases, a CMP problem is less performance consuming than a poorly designed aggregation level which contains too many characteristics. Therefore: check first if it is possible to avoid a CMP problem as described above. In some scenarios, the choice may be only between 'accepting the CMP problem' and 'extending the aggregation level by additional characteristics to avoid the CMP problem'. In this case, you should rather accept the CMP problem than extending the aggregation level.

4.3 Navigation Attributes

Characteristic attributes can be converted into navigation attributes. They can also be used to design a time-dependant scenario, to show always the data “as it is” without reposting the data. If you need to have always the historical view on your data, the characteristic itself needs to be in the InfoCube.

The merchandise category of an article, for example, can be designed as navigation attribute. An article can be assigned to just one merchandise category at a specific time.

When designing the query, there is no difference between navigation attributes and the characteristics of an InfoCube. If a navigation attribute is needed in a planning query, you need to have the characteristic it is assigned to in the query (can be added to the free characteristics). The navigation attribute can also be included in the filter (e.g. to show all articles of a specific merchandise category only).

Navigation attributes can also be used in planning filters as a base for planning functions, for example, to execute a top-down distribution via planning function for the comparable stores only.

Performance

From a performance point of view, you should prefer characteristics over navigation attributes, because:

- In the enhanced star schema of an InfoCube, navigation attributes require an additional join, compared with a query with the same object as a characteristic.

- For the same reason, queries with navigation attributes result in complex SQL statements. In some situations, the database optimizer may not be able to determine the optimal execution plan for these complex statements,
especially when navigation attributes are restricted in the query. In some cases, you can solve this problem by creating a secondary index for the navigation attribute in the corresponding master data tables. However, it is also possible that these secondary indexes change the performance of other queries to the worse.

- If a navigation attribute is used in an aggregate, the aggregate has to be adjusted using a change run to "activate" changed values of the navigation attribute. This is why avoiding using navigation attributes, or not using navigation attributes in aggregates, can improve the performance of the changerun. On the other hand, not using navigation attributes in aggregates can lead to poor query response times.

You may argue that these considerations are not valid when BW Accelerator is in use. This is true. With BW Accelerator, it is not necessary to adapt BWA indexes for fact tables in order to "activate" changed values of navigation attributes. Instead, the system must only re-index the master data tables. Besides, processing navigation attributes in BW Accelerator is usually not an issue.

However, keep in mind that new requests in real-time InfoCubes must be read from the database until they have been rolled up into the BWA index. These database accesses may be more complex if navigation attributes are involved. See chapter 3 for hints how to roll-up new requests in real-time InfoProviders automatically. Additionally, it may be reasonable to design the save button in the web template of the planning application as described in chapter 10.3. These recommendations can help to reduce the negative impact of navigation attributes on the query performance to a minimum.

Navigation Attributes as Lock Characteristics

In the default settings, navigation attributes are not relevant for lock checks in planning applications. This means that they are always completely locked. The reason for this is that attribute values may change in a planning session. However, in some cases, it is useful to specify navigation attributes as lock characteristics to reduce the size of the selection tables. For example, this allows you to use selections based on a product group instead of selections based on an extensive list of products that belong to the product group.

If you want to use navigation attributes as lock characteristics, you must ensure from an organizational point of view that these cannot be changed during planning. This applies to manual changes or changes that are caused by Warehouse Management processes. See SAP Note 928044 for further details.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>928044</td>
<td>BI lock server</td>
</tr>
</tbody>
</table>

In transaction RSPLSE, navigation attributes can only be defined as lock characteristics in the expert mode. To enable the expert mode, call transaction RSPLSE and enter the ok_code EXPERT as shown in figure 6. This will display all available navigation attributes in the list of potential lock characteristics. The ok_code NOEXPERT can be used to switch the expert mode off again.
4.4 Key Figures

The key figures provide the values that are reported on in a query. Key figures can be quantity, amount, or number of items and they need to get a currency or unit assigned.

Key figures can have different types of aggregation, e.g. summation, average or no aggregation. The field exception aggregation determines how the key figure is aggregated in the Business Explorer in relation to the exception characteristic. This reference characteristic must be unique in the query. In general, this refers to time. Note that exception aggregations can impede to performance.

Cumulative key figures are used to build up sales figures etc. Values for this key figure have to be posted in each time unit, for which values for this key figure are to be reported.

Non-cumulative key figures can be used to build up inventories. Two additional key figures belong to this key figure, which display the in- and outflow of the non-cumulative value. Note that non-cumulative can impede to performance (no constraints if BW Accelerator is used).

4.5 DataStore Objects

SAP BW knows three types of DataStore Objects:

- The Direct Update DSO consists of a single flat table with only the user-defined semantic key as the technical key of the table. It is primarily used as a data target in analysis processes and cannot be used as data target within a standard data flow. Reporting is technically possible although the performance of this depends heavily on the data volume stored in this object.

- The Write-Optimized DSO is designed as a fast inbound or staging layer. Like the Direct Update DSO, it also consists of a single flat table. Instead of the user-defined semantic key, it uses the request information (request ID,
data package number and record number) as a technical key. Reporting is possible, but like for the Direct Update DSO, this should be supported by additional secondary indexes and used only in rare cases.

- Unlike the other two DSO types, the Standard DSO consists of the active data table (the data that can be used for reporting or further staging), activation queue (data that still has to be activated) and a changelog table (keeps track of the changes). The technical key is again completely user-defined. Its intended use cases are data consolidation or delta determination. Due to its overwriting capabilities, it is ideal for volatile data. Reporting is again possible, but should be supported by secondary indexes and very selective queries.

General recommendations:

- Usually, DSOs are used for modeling a consolidation or pass-through layer. Although from a technical point-of-view, all three types can be used in a reporting scenario, the user creating such a model should be aware of the fact that DSOs are in that case little more than flat tables. The more data is stored in that table, the slower the reporting performance might get. Of course it is possible to support reporting requirements on DSOs with the help of secondary indexes, but in terms of flexibility, a DSO can never compete with an InfoCube.

- Generation of SIDs during activation of a Standard DSO’s data: If this flag is set, the characteristic values from the records that are activated will be checked against the SID tables of the corresponding InfoObjects. If necessary, SIDs will be generated for values that are unknown. This SID determination and generation process takes place during loading time. If the flag is not set, the SIDs (as a prerequisite for reporting) will be temporarily generated during reporting time. So with this flag, the user can decide if the DSO is designed for fast loading times or fast(er) reporting times.

Typical reporting scenarios for DSOs:

- Real-time reporting

  Data is loaded very frequently into SAP BW and shall be available for reporting with as little delay as possible. This could be achieved with a direct reporting on an intermediate, write-optimized staging DSO. While the data is not yet fully updated into the final data targets (InfoCubes), the users can still see the data as it is in the DSO. From a performance perspective, the data volume in this DSO should stay as small as possible to ensure fast query runtimes.

- Infrequent reporting on highly granular data

  Most of the reporting takes place on aggregated data. Only in rare cases, the detailed source data has to be accessed. If the performance of those detailed reports does not have to be as good as possible, the reporting can be directed directly to the data in the staging layer instead of loading this data into yet another InfoCube.

4.6 InfoCubes

InfoCubes are the standard reporting objects in SAP BW. Their multi-dimensional structure is perfectly suited for both fast and flexible reporting across all data stored in the InfoCube. In order to achieve such a good performance and flexibility, a number of guidelines should be followed during the modeling process. Those rules are summarized in the following chapters.

Semantic Partitioning

The distribution of data to several structurally identical InfoProviders by a logical criterion is called “semantic partitioning”. Usually, those InfoProviders are combined under one MultiProvider. The goal of this modeling strategy is to limit the data volume per InfoProvider to increase the overall performance both from a reporting as well as from an administrative point-of-view.
Figure 7: Semantic Partitioning by Calendar Year

Reporting advantages: When a BW Query is executed on a MultiProvider, the basic InfoProviders underneath can be accessed in parallel. So if a query requests data both from 2008 and 2009, the corresponding InfoCubes will be read simultaneously. This can result in significantly lower query runtimes compared to a scenario where all data is located in one single InfoProvider.

The criterion by which the data is split across the InfoCubes is usually a time characteristic (like ‘calendar year’ in the example above). Data usually grows over time and choosing a time characteristic is a proven way to effectively limit the data volume per InfoProvider. But there is no reason not to combine a time characteristic with another suitable characteristic like ‘country’. For non-cumulative InfoCubes, it is not possible to choose time characteristics as split criteria for semantic partitioning.

Administrative advantages: Data that has exceeded its retention period and shall be deleted does not have to be deleted by using the selective deletion process (which has a series of disadvantages like the automatic invalidation of any BWA index or classic aggregate connected to the InfoProvider). Instead the whole InfoCube can be dropped. New data can be directed to a new InfoCube attached to the reporting MultiProvider.

Figure 8: Administrative Advantages of Semantic Partitioning
In addition to the administrative advantages, the limitation of the data volume stored in a single InfoCube also improves technical database processes like the (re-)creation of a table index where the overall runtime depends heavily on the number of records stored in the table.

**Physical Partitioning**

Physical partitioning (for Oracle and MSSQL databases) or multi-dimensional clustering (for DB2/DB6 databases) is the database equivalent to the semantic partitioning. In both cases the fact table of an InfoCube is split into several smaller pieces (partitions or multi-dimensional cells). During the execution of a report, the database does not have to access the complete huge fact table but instead only reads from the partitions (or MDC cells) that have the data which was requested by the query.

For Oracle and MSSQL, only selected time characteristics can be chosen for physical (range) partitioning. The system applies the settings for physical partitioning only to the E-fact table of an InfoCube. This means that the user can only benefit from the performance advantages of physical partitioning, if the InfoCube data is compressed. To set up physical partitioning, the user has to define the number of partitions.

![Physical Partitioning Example](image)

**E-fact table, partitioned by 0CALMONTH with 12 partitions**

<table>
<thead>
<tr>
<th>0CALMONTH</th>
<th>0CALMONTH = 2008/05</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008/01</td>
<td></td>
</tr>
<tr>
<td>2008/02</td>
<td></td>
</tr>
<tr>
<td>2008/03</td>
<td></td>
</tr>
<tr>
<td>2008/04</td>
<td></td>
</tr>
<tr>
<td><strong>2008/05</strong></td>
<td></td>
</tr>
<tr>
<td>2008/06</td>
<td></td>
</tr>
<tr>
<td>2008/07</td>
<td></td>
</tr>
<tr>
<td>2008/08</td>
<td></td>
</tr>
<tr>
<td>2008/09</td>
<td></td>
</tr>
<tr>
<td>2008/10</td>
<td></td>
</tr>
<tr>
<td>2008/11</td>
<td></td>
</tr>
<tr>
<td>2008/12</td>
<td></td>
</tr>
<tr>
<td>MAXVALUE</td>
<td></td>
</tr>
</tbody>
</table>

- **MAXVALUE** partition contains all records with a **CALMONTH** value greater than „2008/12“
- „2008/01“ partition contains all records with a **CALMONTH** value less than or equal „2008/01“

Figure 9: Physical Partitioning Example

The actual decision which characteristic should be used to setup a physical partitioning depends on the reporting behavior. If most reports (that access this very InfoCube) use a restriction on calendar month, use 0CALMONTH as a partitioning characteristic. If the reports all provide selections on fiscal period, use 0FISCPER as partitioning characteristic. The number of partitions that shall be created initially is determined by the data that is about to be stored in the InfoCube: if you know that only 2008’s data is going to be stored in the cube, it’s sufficient to create only partitions for that time range.

SAP BW offers a variety of tools to add partitions to an InfoCube table, drop empty partitions and change the partitioning characteristic even if the InfoCube contains data. But because of the long runtimes of such a repartitioning process, these tools should be considered as emergency functionalities.
Data Volume

Due to its multidimensional structure, an InfoCube is capable of fast and flexible reporting on high volumes of data. Nevertheless, it can be reasonable to split up InfoCubes and use semantic partitioning concepts as described above. As a rule of thumb, you should consider using semantic partitioning if a single InfoCube would contain more than 200 to 300 million records.

Dimensions

The overall reporting performance on an InfoCube also depends heavily on the layout of its dimensions. First of all, again only as a good rule of thumb, a dimension table should not have more than 50,000 to 100,000 records. The dimension tables are joined to the fact table during reporting and the more data is in the dimension tables the more expensive and slower the database operation gets.

To follow this rule of thumb and keep the number of records in a dimension table as low as possible, the characteristics have to be distributed to the dimensions not from a business perspective but from a purely technical point-of-view. A redistribution according to the business needs can (and should) be done later in a MultiProvider.

The safest way how to decide which characteristics should be grouped together in a dimension is to look at the total number of their characteristic values and assume the worst case: If characteristic A with 500 master data values and characteristic B with 100 master data values are put in the same dimension, it might lead to a total of 500x100=50,000 distinct combinations and thus records in the dimension table. The only exception to this rule is if characteristic B is an attribute to characteristic A and it is technically impossible that A and B in the same dimension will lead to more than 500 records (A’s master data values).

Of course this “worst case” seems like a very technical scenario. But experience shows that business requirements change and a situation that was not considered possible at the beginning of a project might become very real in only a couple of years. And then improperly designed dimensions can have a severe impact on the overall reporting performance which can only be solved by an expensive data remodeling.

Line Items

Like stated in the previous chapter, the number of records in a single dimension table should not exceed 50,000 to 100,000 records. But what about characteristics which have alone much more master data values? Those high-volume InfoObjects should be placed in a separate dimension and the dimension marked as “line-item”.

Line-item dimensions can only contain one InfoObject. The great benefit is that, technically, no real database dimension table is created for this InfoObject, which means that during reporting on this characteristic, the database has to perform one expensive table join less. For this reason, it can also be helpful to place characteristics with low granularity into line-item dimensions. Like this, the complexity of the star scheme can be reduced considerably. The system uses this approach for so-called ‘flat aggregates’ i.e. for aggregates with less than 14 user-defined or time characteristics.

On the other hand, the non-existence of the dimension table also implies some functional restrictions. If a characteristic is used in a line-item dimension, it is no longer possible to easily determine the “posted values” of the characteristic in the InfoCube. For non-line item dimensions, the dimension table can be read to determine the (previously) posted characteristic values of an InfoObject. This is used by the F4 value help mode ‘Only values in InfoProvider’ as well as by the MultiProvider hint. See the following SAP Notes for further details.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>984229</td>
<td>F4 modes for input help as of SAP NetWeaver 7.0 BW</td>
</tr>
<tr>
<td>911939</td>
<td>Optimization hint for logical MultiProvider partitioning</td>
</tr>
</tbody>
</table>
More information on the MultiProvider hint is available in the guide 'How to create efficient MultiProvider queries':

http://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/b03b7f4c-c270-2910-a8b8-91e0f6d77096

An InfoObject with low granularity should only be placed into a line item dimension if the characteristic will not be used as partitioning criterion in semantic partitioning and only if F4 help mode 'Only values in InfoProvider' is not required.
This chapter shall provide general guidelines for modeling Integrated Planning scenarios in SAP BW.

You can define an input-ready query on any of the following InfoProviders:

- Aggregation levels
- MultiProviders that include at least one simple aggregation level

When implementing a planning application, you must opt for one of these modeling alternatives. The following chapters describe the restrictions and pros and cons of both planning architectures and provide a decision matrix.

5.1 Input-ready Query on Aggregation Level

The standard BW-IP architecture allows you to create planning functions and planning queries on top of an aggregation level. Aggregation levels are created in the planning modeler.

Figure 10: Aggregation Level on top of MultiProvider

However, this architecture cannot be used to handle non-cumulative key figures within aggregation levels. When using non-cumulative key figures within actual cubes that are also relevant for your planning queries, for example plan/actual comparison of inventories, the input-ready query needs to be defined on a MultiProvider. For further explanation, please see chapter 5.2.
In the query definition, the characteristic 0INFOPROV should be restricted for each key figure. In a query created on top of an aggregation level, the structure element containing the input-ready key figure must contain 0INFOPROV restricted to a plan InfoCube.

5.2 Input-ready Query on MultiProvider

As mentioned above, input-ready queries must be defined on a MultiProvider if non-cumulative key figures are used within actual cubes that are also relevant for your planning queries, for example plan/actual comparison of inventories.

Remark: Inventory scenarios based on the so-called snapshot concept use cumulative key figures instead of non-cumulative key figures. Therefore, when inventories are designed using a snapshot concept, you can create a planning query on top of a MultiProvider or on top of an aggregation level. More details on inventory scenarios are described in the guide 'How to handle inventory management scenarios in BW'.

http://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/f83be790-0201-0010-4fb0-98bd7c01e328

In the majority of cases, SAP MAP planning applications compare plan inventories with actual inventories (non-cumulative key figures) in the same query. Therefore, the following BW-IP architecture must be used.

![BW - IP Architecture Diagram](http://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/f83be790-0201-0010-4fb0-98bd7c01e328)

**Figure 11: MultiProvider on top of Aggregation Level(s)**

In the query definition, the characteristic 0INFOPROV should be restricted for each key figure. In a query created on top of a MultiProvider, the structure element containing the input-ready key figure must contain 0INFOPROV restricted to the aggregation level.

Please note: For planning functions, you usually need to define an aggregation level on top of a MultiProvider (or directly on top of an InfoCube) as shown in the picture above.
5.3 Pros and Cons

The decision on the appropriate architecture depends on several factors. First of all, it is driven by functional aspects:

- **Inventory Scenarios**
  Input-ready queries must be defined on a MultiProvider if non-cumulative key figures are used within actual cubes that are also relevant for your planning queries.

- **Planning Functions**
  For planning functions, you usually need to define an aggregation level on top of a MultiProvider (or directly on top of an InfoCube). Provided that the planning function is not used in an online planning application (BEx Web or Excel workbook), you could build your planning functions on top of an aggregation level (with a MultiProvider or InfoCube underneath), and your planning queries on top of a MultiProvider (with one or more aggregation levels underneath). So, of course, it is possible to run both architectures side by side.

Apart from the functional aspects, it is also important to compare both architectures from a performance point of view. This is done in a very detailed way in chapter 5.4. To make a long story short, it is advisable for performance reasons to build input-ready queries on top of a MultiProvider (with one or more aggregation levels underneath).

Last but not least, you should consider that reusable query elements, such as structures, query filters, restricted and calculated key figures, are always stored for the object on which the query is defined.

- If input-ready queries are defined on top of a MultiProvider (with one or more aggregation levels underneath), you can reuse these elements for every query which is based on the same MultiProvider.

- If input-ready queries are defined on top of an aggregation level (with a MultiProvider underneath), you can reuse these elements for every query which is based on the same aggregation level.

The latter may cause a higher implementation and maintenance effort. Typically, you need different aggregation levels depending on the required data granularity. For each aggregation level, you need at least one query. Reusable elements must be copied from an aggregation level to another. If they are changed later on, you must change them manually per aggregation level.

If the queries are defined on the MultiProvider, you could add all aggregation levels to this MultiProvider. In this case, you may only need a single MultiProvider and such, redundant maintenance can be avoided. In the query definition, the non-relevant aggregation levels can be "excluded" by restricting 0INFOPROV to the appropriate aggregation level.

5.4 Performance

The collective SAP Note 1056259 provides very detailed and valuable information on best practices in BW planning. The note explains background knowledge for BW planning architecture, important influencing factors for performance and recommendations for modeling.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1056259</td>
<td>Collective note: Performance of BI planning</td>
</tr>
</tbody>
</table>

5.4.1 Design of Aggregation Levels

For optimal performance, it is essential to design lean aggregation levels. Based on the specific query and planning function requirements, an aggregation level should use a set/slice of characteristics and key figures from the underlying InfoProvider. The slice of characteristics and key figures for an aggregation level should not include all data, but match
the specific set you need for the planning query and/or functions. The key figures included in the aggregation level are aggregated using the characteristics that are not included in the aggregation level.

An aggregation level should only contain the characteristics that are required for a specific query or planning function. Instead of using a generic aggregation level for all scenarios, it is highly recommended to design very specific aggregation levels according to the different planning scenarios.

5.4.2 Comparison of Planning Architectures

As a general observation, the runtimes of a well-modeled input-ready BW query running on an aggregation level (with a MultiProvider underneath) are almost the same as the runtimes of a comparable query running on a MultiProvider (with one or more aggregation levels underneath). Performance tests have shown differences in the overall runtime of up to 10% depending on the query definition. The location of the key figures (X- or Y-axis), the number of characteristic combinations, the usage of hierarchies and the complexity of the restricted and calculated key figures are elements which might tip the scales in favor of one of the alternative planning architectures.

However, in a specific scenario, the performance of a planning query on top of an aggregation level may be considerably worse than a comparable query on top of a MultiProvider. This results from technical restrictions in the internal implementation of aggregation levels. Let us have a look at some basics of how a query is being processed internally to better understand these restrictions.

During the query execution, BW’s Analytical Engine tries to minimize the time spent reading the required data by creating so-called selection groups. These groups are determined for each involved basic InfoProvider. They usually correspond to the summarized selections of the key figures’ definitions. This way, each underlying InfoProvider can be accessed in a very restricted way and only the data that is actually needed for the query is retrieved.

The logic described before is used for an input-ready query defined on a MultiProvider (with one or more aggregation levels underneath). However, due to a technical restriction in the internal implementation of aggregation levels, it is not possible to create distinct selection groups per basic InfoProvider if the query is defined on an aggregation level (with a MultiProvider underneath). In that case, BW’s Analytical Engine summarizes the selections across all InfoProviders and creates one all-embracing selection covering all key figure definitions. This selection is then finally passed to all involved InfoProviders. Depending on the specific query definition, this may lead to a situation where a lot more data is read from each InfoProvider than necessary. And that means not only higher database runtimes, but also higher OLAP runtimes since the data has to be sorted, filtered and grouped for the key figures calculations.

If you use a data model where an aggregation level is modeled on top of a MultiProvider, please make sure the key figures are restricted as much as possible: If a key figure is already restricted to an InfoProvider that contains only records for a certain characteristic value (e.g. “version = 0”), it seems unnecessary to restrict it additionally to that value. But since all the selections will be summarized in the Analytical Engine, it can make a huge difference (from a performance perspective), if “version = 0” or no restriction at all (on version) is passed as a selection criterion to the underlying InfoProviders.

To avoid such problems, it is recommended to design the data model so that input-ready queries are defined on top of a MultiProvider (with one or more aggregation levels underneath). In most scenarios, this is the data model of choice. The risk of running into performance problems due to too complex and distinct key figure restrictions is a lot smaller compared to a scenario with input-ready queries defined on top of an aggregation level (with a MultiProvider underneath).
5.4.3 Detailed Example

A planning query uses three kinds of data:

- Current planning data from InfoCube “PW_PY_AM1”. The relevant planning data is identified by version = “FO0” and status = “1”. This InfoCube is accessed for both reading and writing.
- Previous year’s planning data from InfoCube “PW_PLYAM1”. The relevant data is identified by version = “FO0” and status = “1”. This InfoCube is only accessed for reading.
- Current sales data from InfoCube “PW_ACT_AM”. This InfoCube holds only data for version = “0” and status = “0”.

Figure 12: Example Scenario

**Option 1:** The MultiProvider is modeled on top of an aggregation level.

Figure 13: MultiProvider on top of Aggregation Level
Option 2: The aggregation level is modeled on top of a MultiProvider.

The query itself reads the data with the following selections from the involved InfoProviders:
- The current plan data from InfoCube “PW_PY_AM1” (or the overlying Aggregation Level) is read in version “FO0” and status “1”.
- Last year’s plan data from InfoCube “PW_PLYAM1” is also read in version “FO0” and status “1”.
- The current sales data from InfoCube “PW_ACT_AM” is not restricted to any version and/or status since the InfoCube contains only one type of data anyway.

Figure 14: Aggregation Level on top of MultiProvider

Figure 15: Query Selections
When the query is executed on the MultiProvider-based scenario, the first object processed by BW’s Analytical Engine is the MultiProvider. BW “knows” which basic InfoProviders are located underneath the MultiProvider and can create the selection groups based on the query definition:

- InfoCube “PW_ACT_AM” is read without any restriction on version and status.
- InfoCube “PW_PLYAM1” is read with version = “FO0” and status = “1”.
- Aggregation level “PW_AL_AM1” is read with version = “FO0” and status = “1”.

![Figure 16: Execution on MultiProvider-based Scenario](image-url)
However when the same query is executed on the Aggregation level-based scenario, it is handled slightly differently: When BW’s Analytical Engine processes the first object (here: the aggregation level on top), it “knows” only about the next object, the MultiProvider. Therefore it cannot split the query’s selections and restrictions into provider-specific groups but instead it has to summarize everything into one all-embracing selection across all InfoProviders:

- InfoCube “PW_ACT_AM” is again read without restriction to version and status.
- InfoCube “PW_PLYAM1” is now read without restriction to version and status.
- InfoCube “PW_PL_AM1” is also read without restriction to version and status.

In total, a lot more data than necessary is read from database. This data has to be filtered out later in the Analytical Engine. And this leads to a big impact on the overall query performance.

Figure 17: Execution on Aggregation Level-based Scenario
A possible way to prevent such a situation from happening is to additionally restrict the reading of data from InfoCube “PW_ACT_AM” to version = “0” and status = “0”. Then the summarized selections wouldn’t be “no restriction on version and status” but “version = [0;F00]” and “status = [0;1]”.

Of course BW still reads too much data from the InfoCubes “PW_PY_AM1” and “PW_PLYAM1” (the result for version = “0” and status = “0” is not needed and just thrown away afterwards), but the impact on the overall performance is much smaller than in the previous example.
6 Characteristic Relationships

This chapter shall provide general guidelines for setting up characteristic relationships for real-time InfoCubes in SAP BW.

Characteristic relationships can be used to define rules to check permitted combinations, generate new valid combinations and derive dependant characteristics. The system provides four different kinds of characteristic relationships. So it is possible to build characteristic combinations based on master data attributes, hierarchies, DataStore objects and exit classes. Characteristic relationships are defined in the Planning Modeler for real-time InfoCubes.

Please keep in mind that the system does not only display data which match the characteristic relationship. It will also display already posted values of the real-time InfoCube in the query result which do not match. Thereby, the system draws your attention to incorrectly booked values. But new records created via input ready queries and planning functions will only be saved in real-time InfoCubes if they are consistent with the characteristic relationships.

Use characteristic relationships sparingly since the system has to check the validity of each data record that can be changed and for each delta record, the system processes all applicable derivation steps. The more relationships you use, the more checks are required.

Each step in a characteristic relationship implements two methods: check and create. In addition, a relation of type with derivation also implements the method derive.

The check method is used to check a record used in input ready queries and planning functions. Input ready queries check records that correspond to input ready cells, including records created in new lines. Planning functions will only check new records, of course with the exception of planning functions types designed to work explicitly with characteristic relationships, e.g. ‘Delete Invalid Combinations’. Whether the check method of a relation is applied, depends on the characteristics used in the aggregation level.

- The relation is of type with combination check: The check method is applied when all characteristics used in the relation are contained in the aggregation level.

- The relation is of type with derivation:
  - For relations of type master data attribute or data store, the check method is applied if all source characteristics and at least one target characteristic are contained in the aggregation level.
  - For relations of type hierarchy and exit, the check method is applied when all characteristics used in the relation are contained in the aggregation level.

The derive method of a relation of type with derivation will be called in the delta buffer for all new delta records to derive characteristics values from characteristic values filled in the aggregation level. The derivation method of a relation will be called when all source characteristics are contained in the set of ‘known’ fields and at least one target characteristic is contained in the set of unknown fields. Initially, the set of known fields is the set of characteristics contained in the aggregation level, the set of unknown fields is the set of characteristics used in the real-time InfoCube not used in the aggregation level. Applying the above rule, the system calls the derive method of the relations as long as new fields can be filled.
Observe that the initial value of a characteristic plays a special role in the check, derive and create methods of a relation. This is explained in more details in the attached document of SAP Note 994853, cf. section 2.2.

### SAP Note 994853

**Description**

Input-ready queries: Top-down distribution, budgeting

---

Example: The system shall derive higher nodes of an article hierarchy from a lower one. The planning content has two aggregation levels. The first aggregation level just contains 0CM_CDT4 and the second one 0CM_CDT3. Therefore, two characteristic relationships must be defined with different source fields.

<table>
<thead>
<tr>
<th>Source characteristics</th>
<th>Target characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0CM_CDT4, 0CM_HIEID</td>
<td>0CM_CDT1, 0CM_HIEID, 0CM_CDT2, 0CM_CDT3</td>
</tr>
<tr>
<td>0CM_CDT3, 0CM_HIEID</td>
<td>0CM_CDT1, 0CM_HIEID, 0CM_CDT2</td>
</tr>
</tbody>
</table>

As stated before, a characteristic relationship is only applied if all source fields of a characteristic relationship are included in the aggregation level. However, the system is able to process cascading characteristic relationships. Therefore, it would also be possible to define relationships as follows:

<table>
<thead>
<tr>
<th>Source characteristics</th>
<th>Target characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0CM_CDT4, 0CM_HIEID</td>
<td>0CM_CDT3, 0CM_HIEID</td>
</tr>
<tr>
<td>0CM_CDT3, 0CM_HIEID</td>
<td>0CM_CDT2, 0CM_HIEID</td>
</tr>
<tr>
<td>0CM_CDT2, 0CM_HIEID</td>
<td>0CM_CDT1, 0CM_HIEID</td>
</tr>
</tbody>
</table>

Regarding the 'access type for result rows' in the query definition, it is recommended to choose the option 'characteristic relationships' for planning scenarios. The option 'master data' should only be used in reporting. While using the setting 'characteristic relationship' for an InfoObject which gets used in a characteristic relationship during runtime, all other characteristics within this relation get this setting automatically as well.
Processing characteristic relationships can be time-consuming. For relationships of type exit class, the system provides an infrastructure to buffer the results of checks and derivations. This buffer is not active by default. If compatible with your coding logic, you can decide to activate the buffer. To do so, you must redefine the constructor of your exit class and add the following coding line after the data declaration in the redefined 'constructor' method.

* Performance improvement: Use external buffer
  if_rspls_char_relation_n_use_external_buffer = 'X'.

Redefining a constructor is only possible if you create your exit class by copying the class CL_RSPLS_CR_EXIT_BASE. It is not possible, if your exit class is a subclass of CL_RSPLS_CR_EXIT_BASE. In that case, replace your exit class by a copy of class CL_RSPLS_CR_EXIT_BASE.

For further information on buffering characteristic relationships, please read SAP Note 1067433.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1067433</td>
<td>Performance optimization when you use derivations</td>
</tr>
</tbody>
</table>

Additional information on characteristic relationships can also be found in the collective SAP Note 1056259.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1056259</td>
<td>Collective note: Performance of BI planning</td>
</tr>
</tbody>
</table>
7 Variables

Variables are used to parameterize a query, a planning function, a filter, a characteristic relationship or a data slice. When a query, planning function or Web application is executed, they are filled with values. Variables act as placeholders for characteristic values, hierarchies, hierarchy nodes, texts, and formula elements, and can be processed in different ways (manual entry/default value, replacement path, customer exit, SAP exit, authorizations).

For characteristics with many master data records, it can be beneficial from a performance point of view to do a preselection in the query by introducing user exit variables which get filled by other (user) selections. This approach can improve the query runtime even if characteristic relationships are used.

An example coding was implemented in the SAP MAP Content 7.04 Support Package 04, for example variable 0P_MCDT1_FROM_CDTX to determine multiple categories from higher category levels (see also coding logic in function module RSVAREXIT_0P_MCDT1_FROM_CDTX).
This chapter explains how BW hierarchies and query display hierarchies can be used to optimize query performance.

8.1 BW Hierarchy

The SAP MAP Content uses so called time hierarchies which are characteristic hierarchies on InfoObject 0FISCPER3. Time hierarchies can easily be generated in transaction UPARP_HIERT_T. A time hierarchy can be used at query runtime to navigate through differently granular levels of time (e.g. from year to half-year to quarter to month to week). All nodes and subnodes are ready for input and can be locked using the cell locking feature.

It is recommended to use BW hierarchies instead of query structures. Hierarchies provide easier handling, central maintenance and full support of features like cell locking and calculation engine.

Using BW hierarchies also implies some modeling restrictions. Calculations can only be done on the lowest level (which is 0FISCPER3 in case of MAP time hierarchies). If these calculations are different on the higher levels (month, quarter, etc.), a time hierarchy might have to be modeled differently or the calculations need to be done on the frontend itself (Excel, Web).

8.2 Master Data Attribute driven Hierarchy

In SAP MAP, a so called article hierarchy is used. This is not an object in SAP NetWeaver BW, but a master data attribute driven hierarchy. Therefore you need to define characteristic relationships of type 'master data attributes' to create and derive the right combinations during a planning session as well as when writing data into the InfoCubes.

8.3 Display Hierarchy

In the Query Designer, it is possible to display characteristics hierarchically. Hereby, it can be specified that one or both axes (rows and columns) are displayed as hierarchy. These functions enable users to evaluate the data of a query by browsing through the hierarchy. To activate display hierarchies, left-click on the 'Rows' or 'Columns' label in the query designer. Typically, the display hierarchy is configured not to display data down to the lowest level already in the initial query view.

![Display Hierarchy in Query Designer](image)

Query navigation through display hierarchies is very similar to BW hierarchies.
First of all, display hierarchies reduce the query response times significantly. In many scenarios, using display hierarchies is a key success factor for well-performing planning applications. All front-end navigation steps (open query, open view, drag & drop, sort, set filter, swap axis) and even calculation steps (aggregation, disaggregation) perform much faster. This performance gain results from the fact that the system displays a smaller query result which involves fewer records to be retrieved and processed (input ready check).

Display hierarchies do not only have a positive impact on performance. Because of the smaller initial view, the query gets more usable. It is evident that a user can navigate easily through a query with only a few hundreds or thousands of cells being displayed at a time. By browsing through the hierarchy, the user can still access a wide range of data without changing the variable selections or opening another planning application.

On the other side, the use of display hierarchies leads to some modeling advices/restrictions. Query characteristics and key figures should be separated on the axes and not mixed up in the columns and/or rows.

Furthermore, the use of display attributes is not recommended in combination with display hierarchies. The headers and “empty cells” of these display attributes will be shown over all display hierarchy levels even if the level of the display attributes is still collapsed.

The following table summarizes the pros and cons of display hierarchies:

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initial query runtime as well as navigation runtime can be reduced significantly.</strong></td>
<td>Display attribute headers of characteristics are shown over all levels of the hierarchy even if the corresponding characteristic is not expanded.</td>
</tr>
<tr>
<td>Large queries get clearer and more structured. The usability of the planning application is improved considerably.</td>
<td>In the case that key figures are not separated (columns) from the characteristics (rows), totals are not shown on the higher levels of the display hierarchy. Additionally in specific modeling scenarios, totals are not input ready for planning.</td>
</tr>
<tr>
<td>Scrolling works in a more proper way.</td>
<td>It is not possible to navigate to a special level of the hierarchy via context menu as it works for BW hierarchies.</td>
</tr>
</tbody>
</table>
This chapter explains performance relevant aspects of query design as well as a few functional enhancements to the BW Analytical Engine.

As mentioned already, we recommend reading the collective SAP Note SAP Note 1056259. It also contains very valuable input regarding best practices for query (and web application) design.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1056259</td>
<td>Collective note: Performance of BI planning</td>
</tr>
</tbody>
</table>

### 9.1 Query Properties

Technical query properties are not maintained in the Query Designer but in the query monitor. To maintain these settings, call transaction RSRT and left-click on the button 'Properties'.

![Query Properties in Query Monitor](image)

A very detailed and useful explanation of these settings is provided in SAP Note 1136163.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1136163</td>
<td>The query settings in RSRT -&gt; Attributes</td>
</tr>
</tbody>
</table>

This document focuses on the recommendations for input-ready queries. Please keep in mind that specific scenarios may require differing settings.
<table>
<thead>
<tr>
<th>Query Property</th>
<th>Plan Buffer P/!!1P (1)</th>
<th>Scenario as described in chapter 5.1</th>
<th>Scenario as described in chapter 5.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read mode</td>
<td>H</td>
<td>X</td>
<td>H</td>
</tr>
<tr>
<td>Request status</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Cache Mode</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Delta Cache</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SP Grouping</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use Selection of Structure Elements</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

(1) with P = MultiProvider, if aggregation level is defined on top of MultiProvider
(2) with P = real-time InfoCube, if MultiProvider is defined on top of aggregation level
(3) with A = aggregation level
(4) with Q = technical query name
with M = MultiProvider

Example A:

You created a query with technical name **Z_PLANQUERY_A** on top of an aggregation level with technical name **Z_ALVL** (with a MultiProvider underneath).

- To maintain the plan buffer properties, call transaction RSRT and enter **Z_MPRO/!!1Z_MPRO** as 'query'. Then, press button 'Properties' and adjust the settings as described in the table above in column 'Plan Buffer'.
- Then, to maintain the plan query settings, enter **Z_ALVL/Z_PLANQUERY_A** as query and press again button 'Properties'. Adjust the settings as described in the table above in column 'Plan Query on Aggregation Level'.

Example B:

You created a query with technical name **Z_PLANQUERY_B** on top of a MultiProvider with technical name **Z_MPRO**, with an aggregation level **Z_ALVL** underneath. The aggregation level **Z_ALVL** is defined on top of the real-time InfoCube **Z_PCUBE**.

- To maintain the plan buffer properties, call transaction RSRT and enter **Z_PCUBE/!!1Z_PCUBE** as 'query'. Then, press button 'Properties' and adjust the settings as described in the table above in column 'Plan Buffer'.
- Then, to maintain the plan query settings, enter **Z_MPRO/Z_PLANQUERY_B** as query and press again button 'Properties'. Adjust the settings as described in the table above in column 'Plan Query on MultiProvider'.

9.2 Filter

In the BEx Query Designer, you can specify filter values as 'characteristic restrictions' or as 'default values'. Characteristic restrictions (on the left-hand side of the filter area) are also called 'global filter' and cannot be changed during query navigation. Default values (on the right-hand side of the filter area) are only used for the initial query view and can be changed by the user.
From a performance point of view, it is highly recommended to restrict the query result as much as possible by defining tight characteristic restrictions in the query designer.

9.3 Rows/Columns

As explained before, it is highly recommended to use display hierarchies. They improve performance and usability especially of large plan queries. See chapter 8.3 for further details.

For performance reasons, you should display result rows in queries only if really required. Whenever possible, choose the option 'always suppress' result rows. Displaying result rows can have a very negative impact on query runtimes, but may be dispensable in many situations as the following example may show. It is a specific and at the same time widespread scenario where result rows can be suppressed without any loss of functionality:

Quite often, users create queries with characteristics that are restricted by mandatory, single value variables. This characteristic is then added to the query drilldown in the rows or columns in order to show the selected value in the query result. The drilldown will never show more than a single value and it is obvious that there is no need to display a result row for this characteristic. In such cases, the characteristic property for result rows should be set to 'always suppress'. In a sample query with several characteristics in the drilldown, each restricted to a mandatory, single value variable, it was possible to improve the query performance by factors, simply by adjusting the properties for 'result rows'.

9.4 Free Characteristics

Regarding free characteristics, it is important to keep in mind that they play a 'hidden' role in input-ready queries when it comes to disaggregation. The system disaggregates a value also on the level of free characteristics if they belong to the aggregation level.

9.5 Key Figures

Each key figure selection should restrict the InfoObject 0INFOPROV. Depending on the architecture of your data model, an input-ready key figure must be restricted to either the aggregation level or the real-time InfoCube.

In a query created on top of an aggregation level, the structure element containing the input-ready key figure must contain 0INFOPROV restricted to a plan InfoCube.

In a query created on top of a MultiProvider, the structure element containing the input-ready key figure must contain 0INFOPROV restricted to the aggregation level.

9.6 Structures

For performance reasons, you should avoid using two structures in queries. Besides performance issues, there are also limitations in connection with the cell locking feature as well as the calculation engine.

Queries with two structures and, if necessary, cell definitions are almost always 'large' in the following sense: Each cell is described by selections; in addition, there are selections from the filter. As a result, the query has a high fixed cost portion that is required just for the cell description. In addition, there is the effort required to assign transactional data to these cells and, if necessary, also to determine whether the cells are ready for input. For queries with two structures, you should use one characteristic in exactly one structure or as a free characteristic. If you use variables in a structure, replace these by a single value but never by a long list of single values.

Characteristics that are restricted in the structures should not then also be used in drilldowns. In an extreme case, this makes it necessary for the intersection to be calculated for each cell to determine if it is ready for input or not.
9.7 Frontend Disaggregation

The Query Designer offers different options regarding the disaggregation behavior of a key figure. An input ready key figure can be disaggregated with equal distribution, self reference or analog to another key figure within the query. The different options seem to be performing equally.

Disaggregation requires posted values as reference data. If no posted values are available, you have to use the setting 'characteristic relationships' as 'access type for result values'. In the latter case, it is important to have tight restrictions in the query, especially for those characteristics that are not in the query drilldown, but in the aggregation level. These characteristics are relevant for disaggregation.

A very detailed documentation with regards to disaggregation is provided in the attached document of SAP Note 994853.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>994853</td>
<td>Input-ready queries: Top-down distribution, budgeting</td>
</tr>
</tbody>
</table>

9.8 F4 Value Help

Many projects have clear functional requirements regarding the F4 value help. For example, users only want to see a list of values according to a characteristic relationship. Since NetWeaver 7.00, it is easy to implement this for the value help dialog when filtering a characteristic (relational browse). However, it is not possible to influence the value help in the variable screen.

SAP NetWeaver 7.20 will introduce a new BAdI to restrict the value list in the F4 input help in the variable screen. The name of the BAdI is RSR_VARIABLE_F4_RESTRICT_BADI. As soon as the documentation for 7.20 is released, there will be further information regarding the implementation of this BAdI.

9.9 Inverse Formulas and Cell Locking

NetWeaver 7.02 introduces a new feature called 'inverse formulas'. With Support Package SP05, inverse formulas are also available in NetWeaver 7.01. See SAP Note 1358614 for further details. Unfortunately, it is not possible to downport the new feature to NetWeaver 7.00.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1358614</td>
<td>Downport: Inverse formulas in input-ready queries</td>
</tr>
</tbody>
</table>

From 7.01 SP05 onwards, it is possible to make calculated key figures input-ready. Inverse formulas define calculation rules that will be applied when a user changes a calculated key figure.

Example: The calculated key figure 'Average Sales Price' is calculated as NDIV0( 'Sales' / 'Sales Quantity' ). Since 'Average Sales Price' is a calculated key figure and should be input ready, the system needs rules that describe how changed values of 'Average Sales Price' should be calculated back to either 'Sales' or 'Sales Quantity'. These rules are defined by so called 'inverse formulas'.

Up to now, such scenarios could only be covered by help of planning functions that were triggered by the user to recalculate the query result. Now, these planning functions can be replaced by inverse formulas (if the operands of the calculations are also included in the query on the same level of aggregation). First measurements show that inverse formulas do not have a negative impact on query runtimes.

SAP Note 1236347 contains an attachment providing a detailed documentation of inverse formulas.
<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1236347   | Downport: Inverse formulas in input-ready queries

This chapter explains performance relevant aspects of web application design and gives an outlook on the new flicker-free rendering mode based on AJAX technology.

10.1 Application Building

Quite often, projects build very large, universal planning applications i.e. all functional planning requirements are packed in a single web template. It is obvious that such an approach may lead to severe performance problems. Instead, it is recommended to build several web applications that are tailored to specific needs.

To design fast web applications, it is recommended to keep the number of data providers in a web template to a minimum. A web template with a data provider for planning and several other data providers for actual/plan comparison, reference data of last year, year before last and so on, will not only have bad response times; it will also be difficult to handle by the user.

A tabstrip container can help to structure a planning application. When the web template is loaded, the system instantiates all data providers in the application (i.e. generates the queries, reads variable default values and so on). However, it does not read and calculate the data required for the queries on inactive tab pages.

10.2 Planning Functions

Event based execution of planning functions, for example while opening a web template, can impede performance. It can be very painful if the execution of an expensive planning function is bound to the ENTER key (button property ‘Execute with ENTER’, FIRE_ACTION_ON_ENTER_KEYPRESS). The property ‘Execute with ENTER’ should rather be used to trigger the refresh of an application.

10.3 Save Button

In scenarios with BW Accelerator index on the planning InfoCube (and only there!), you should implement a button to save planning data as a two-step command sequence with:

1. command SAVE_DATA
2. command REFRESH_DATA

The command REFRESH_DATA ensures that the query refreshes technical information, as for example the current roll-up status of the involved InfoProviders. This may have a positive impact on the query response times. For further details see chapter 3.

10.4 Paging

We strongly recommend using the standard paging feature of the web application designer for analysis items. Please see online help for further information:

http://help.sap.com/saphelp_nw70ehp1/helpdata/en/76/489d39d342de00e10000000a11402f/framset.htm
10.5 Flicker Free Rendering

BEx Web always refreshes the whole page even if only smaller parts of the page were changed. SAP is about to change this behavior by introducing a new flicker free rendering mode. The new (AJAX) rendering mode will be not defined as default rendering (to prevent customer scenarios which are using onLoad event in HTML pages). Flicker free rendering must be activated system-wide or locally per web template. The feature is officially supported once SAP Note 1177038 is released for customers. The note will also describe how to activate the feature.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1177038</td>
<td>BEx Web 7.01: Flicker Free (AJAX rendering)</td>
</tr>
</tbody>
</table>
11 Scalability of Planning Applications

To ensure the scalability of a planning application, it is essential to take the appropriate decisions regarding the data model. Furthermore, it is crucial to build "reasonable" queries as well as web applications. This document provided many recommendations with regards to these aspects in the previous chapters.

However, the best-designed application cannot scale if the underlying hardware and software layers do not scale. It is evident that the system landscape must be designed for scalability as well. Both, SAP Application Server ABAP and JAVA, offer mature technologies to build scalable system landscapes. While many projects have experience with large installations of AS ABAP for years, it is often not so well-known how to set up a scalable system landscape with AS JAVA. Due to its complexity, this topic cannot be covered within this document. Fortunately, the cluster architecture of AS JAVA and its deployment options are described in detail in the SAP Library:

http://help.sap.com/saphelp_nw70ehp1/helpdata/en/2e/611724f410254ca12a3f396ec5ae85/frameset.htm

A scalable system architecture is still not enough. The proper configuration of the system landscape is just as important to handle high data volumes as well as a high number of concurrent users. This applies to the hardware, the operating system, the database and the SAP instances. These configuration tasks are very specific depending on the hardware, operating system and database in use. Nevertheless, this chapter is intended to share some lessons learned with you as far as the configuration of the SAP system is concerned.

- **Basis Parameters in AS ABAP**
  Recommendations how to configure the SAP basis for high user load are summarized in several collective SAP Notes.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>384971</td>
<td>System parameters for high interface load</td>
</tr>
<tr>
<td>192658</td>
<td>Setting parameters for BW systems</td>
</tr>
</tbody>
</table>

Please take your time to read also the related SAP Notes that are mentioned in the collective SAP Notes.

- **Connections for SAPFactory in AS JAVA**
  To be prepared for more than 100 concurrent users, you should increase the maximum number of connections for the application resource SAPFactory in AS JAVA. To do so, call NetWeaver Administrator → System Management → Configuration → Application Resources. Then, choose "Connection Factories" from the drop-down list, mark the entry 'SAPFactory' and choose the tabstrip 'Pooling'. There, you should increase the parameter 'Maximum Connections' to a value higher than the expected maximum number of concurrent users.

- **BW Planning Lock Server**
  The correct configuration of the BW lock server for planning is an important task in case you expect medium or high user load in planning applications. Settings are maintained in transaction RSPLSE. SAP Note 928044 explains the different lock mechanisms down to the lowest details. In case you use the default lock mechanism, you should pay attention to the recommendations regarding parameter abap/shared_objects_size_MB. Its default value is too small for planning applications with medium or high user load.
OLAP Cache

Query results can be stored in the so-called OLAP Cache. Later, these results can be reused by the same or other users. Depending on the cache mode, the cache entries are stored in main memory, in flat files or in database tables. The cache mode can be set on the level of the InfoProvider or directly in the query properties (transactions RSDIPROP, RSRT). See also chapter 9.1 for further details.

The standard cache modes 1, 2, 3 and 4 require a cache directory. Write accesses to this cache directory are protected by SAP enqueues. These SAP enqueues can lead to locking conflicts i.e. users have to wait until the SAP enqueue of another user is released. This can lead to increased query response times. In rare cases, it may also end up in an enqueue deadlock. Enqueue conflicts due to the OLAP cache directory only occur in rare situations and only if the number of concurrently active reporting or planning users is very high (>50). In this context, "concurrently active" means that all users start queries at the same time.

SAP Note 1026944 introduces a new, enhanced cache mode. Unlike the previous cache modes, it does not need a central cache directory. This architecture eliminates the need of SAP enqueues, thus avoiding locking conflicts or enqueue deadlocks. Cache entries are stored in database tables if the enhanced cache mode is chosen. The new cache mode must be activated by setting an RSADMIN parameter. See SAP Note 1026944 for further details. As stated before, it is recommended to use this cache mode only if the number of concurrently active users is very high.

Further information on locking conflicts in context with the OLAP Cache is provided in the collective SAP Note 1107434.

<table>
<thead>
<tr>
<th>SAP Note</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1026944</td>
<td>New cache mode for BI 7.0 without directory</td>
</tr>
<tr>
<td>1107434</td>
<td>OLAP cache consultation: Long wait times due to locks</td>
</tr>
</tbody>
</table>
Appendix – List of Figures

Figure 1: Automatic Request Processing...................................................................................................................... 4
Figure 2: Delta Index Mechanism................................................................................................................................. 4
Figure 3: Activation of Delta Index ............................................................................................................................... 5
Figure 4: RSRV Check ‘Size of Delta-Index’ ................................................................................................................ 5
Figure 5: MultiProvider Check..................................................................................................................................... 7
Figure 6: Transaction RSPLSE Expert Mode................................................................................................................ 9
Figure 7: Semantic Partitioning by Calendar Year ....................................................................................................... 11
Figure 8: Administrative Advantages of Semantic Partitioning .................................................................................. 11
Figure 9: Physical Partitioning Example .................................................................................................................... 12
Figure 10: Aggregation Level on top of MultiProvider ............................................................................................... 15
Figure 11: MultiProvider on top of Aggregation Level(s) .......................................................................................... 16
Figure 12: Example Scenario ..................................................................................................................................... 19
Figure 13: MultiProvider on top of Aggregation Level ............................................................................................... 19
Figure 14: Aggregation Level on top of MultiProvider ............................................................................................... 20
Figure 15: Query Selections....................................................................................................................................... 20
Figure 16: Execution on MultiProvider-based Scenario ........................................................................................... 21
Figure 17: Execution on Aggregation Level-based Scenario ...................................................................................... 22
Figure 18: Execution on Aggregation Level-based Scenario (optimized) ................................................................. 23
Figure 19: Access Type for Result Values in Query Designer ..................................................................................... 26
Figure 20: Display Hierarchy in Query Designer ......................................................................................................... 28
Figure 21: Query with Display Hierarchy .................................................................................................................... 29
Figure 22: Query Properties in Query Monitor ........................................................................................................... 30