Performance Tuning for Queries with Aggregates
ASAP FOR BW ACCELERATOR

BUSINESS INFORMATION WAREHOUSE

Documentation on Using Aggregates in SAP BW
Document Version 1.0 of June/2000

Table of Contents

1 Introduction

This document provides an overview of the option of using aggregates in the SAP Business Information Warehouse. It does not refer to a particular release, except where indicated.

The data in a Data Warehouse is largely very detailed. In SAP BW, the InfoCube is the primary unit of storage for data for reporting purposes. The results obtained by executing a report or query represents a summarized dataset. Drilling down from the higher aggregation levels provides more detailed data, but at the same time restricts the resulting dataset. Aggregates can be created to optimize the performance of these analyses.

An aggregate is a materialized, summarized view of the data in an InfoCube. It stores a subset of InfoCube data in a redundant form. When an appropriate aggregate for a query exists, summarized data can be read directly from the database during query execution, instead of having to perform this summarization during runtime. Aggregates reduce the volume of data to be read from the database, speed up query execution time, and reduce the overall load on the database.

When a query is executed or query navigation takes place, the OLAP Engine splits the request into several database queries[1]. The system then looks for the best possible aggregate for each of the database queries. An easy way to conceptualize an aggregate is to think of them as providing a similar benefit as adding an additional index to a database table. Aggregates require no special knowledge by the end-user, as they are completely transparent. The only way an end-user might recognize the existence of an aggregate is by the performance gain that is observed.

You can only create aggregates for BasicCubes[2]. BasicCubes are InfoCubes that have a fact table and dimension tables. You do not have to create aggregates for MultiCubes because the system splits up a MultiCube query into subqueries for the individual sub-InfoCubes. These subqueries can then use the aggregate of the sub-InfoCubes.

2 Aggregates

Aggregates should be built to enhance the performance of specific queries or reports. When building aggregates, consider the specific characteristics, navigation attributes, or hierarchy nodes that the query requires to fulfill the request it makes of the database. Aggregates can then be tailored to meet the requirements of queries or sets of queries in a more optimal manner.

---

[1] In the 1.2B System there is no more than one database query for every navigation step.

[2] System 1.2B only has BasicCubes. RemoteCubes and MultiCubes are not supported here.

Aggregates can be built from the characteristics and navigation attributes of an InfoCube. This means that you can also store the values in the hierarchy level of an external hierarchy. You can use individual filter values in an aggregate. Characteristics and navigation attributes, which you want grouped according to their values in the aggregate, are indicated by a '*'. If you want a characteristic or navigation attribute to be filtered according to a single, fixed, value, an 'F' is used to indicate this, and the specific filter value must be specified. If you want the aggregate to be grouped according to the nodes that are on one level of an external hierarchy, use an 'H' to symbolize this; the hierarchy and the level must be specified.

*	Grouped according to the values of the characteristic / navigation attribute.
H	Grouped according to the nodes of a hierarchy level.
F	Filtered according to a fixed value.

Aggregate Definition Description

Essentially, you can summarize the values in the aggregates further. For example, if an aggregate is grouped by values of a characteristic, which is not used in a query, aggregation takes place with this characteristic from the database. This analogy also goes for nodes from a hierarchy. If the query needs the values for hierarchy nodes then you can use all the aggregates that are grouped according to hierarchy nodes, whose level is below the smallest level needed. Exceptions to this rule will be discussed later.

Please note that time-dependent navigation attributes and time-dependent hierarchy structures cannot be used in an aggregate. In these cases, however, you can group the aggregate according to the values of its characteristics. If the entire hierarchy is time-dependent it can be used in an aggregate, but if the hierarchy structure is time-dependent it cannot be used. See the relevant ASAP document on hierarchies for details.

2.1 Examples

The following examples of aggregates are based on a very simplified InfoCube with the two characteristics "Country" and "Customer" and the key figure "Sales". The data is as follows:

Country	*Customer*	*Sales*
USA	Buggy Soft	10
Germany	Ocean Networks	15
USA	Funny Duds	5
Austria	Ocean Networks	10
Austria	Thor Industries	10
Germany	Funny Duds	20
USA	Buggy Soft	25

Example InfoCube

2.1.1 Aggregate grouped according to characteristic value

An aggregate that contains the country information and is summarized with the customers looks like this:

Country	*Sales*
USA	40
Germany	35
Austria	20

Aggregate: Country '*'

This example aggregate could be used by a query that reports the sales by country or total sales. The aggregate could also be used for reports according to a navigation attribute for the characteristic country or according to a hierarchy containing the countries. If during query navigation there is a drilldown or a selection is made for a "customer", this aggregate would not be used. This is because this aggregate does not contain any detailed information about the customers, only summarized customer data. Aggregate filtered according to a fixed value.

2.1.2 Aggregate filtered according to a fixed value

An aggregate that only contains German sales would only be used by queries that have the same filter. Filter values for aggregates should only be used if you only require one value for a specific query, or navigation step. Typical examples are the plan/actual sign, the present financial year and some sort of version.

Country	*Customer*	*Sales*
Germany	Ocean Networks	15
Germany	Funny Duds Inc	20

Aggregate: Country 'F' = DE; Customer '*'

2.1.3 Aggregate grouped according to navigation attribute value

The customer would have the navigation attribute of the industry with the following master data table:

Customer	*Industry*
Buggy Soft	Technology
Ocean Networks	Technology
Funny Duds	Consumer
Thor Industries	Chemicals

Customer Master Data Table

An aggregate for the industry contains the following values:

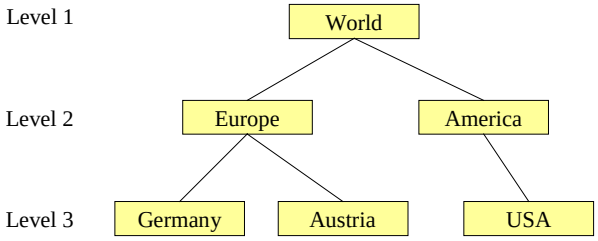
Industry	*Sales*
Technology	60
Consumer	25
Chemicals	10

Aggregate: Industry '*'

In this industry aggregate example, aggregation takes place with customers. If additional detail is required, such as the specific customer detail within these industries, this aggregate would not be utilized. However, you can obtain the total sales as well as node values if there is a hierarchy above the industry.

2.1.4 Aggregate grouped according to the nodes of a hierarchy level

An external hierarchy with the countries is required for this example:



Country Hierarchy 1

Customer	*Sales*
Europe	55
America	40

Aggregate: Country 'H' level 2

This aggregate can be used by queries that report on sales for a hierarchy node on level 2 or higher (meaning level ≤ 2). In this example these are the nodes "Europe", "America" and "World". It can also be used by queries that have this country hierarchy as a presentation hierarchy in the drilldown, but the drilldown goes no lower than the second level. The read mode of the query plays an important role in presentation hierarchies. If the hierarchy does not have multiple leaves, it means that every characteristic value appears not more than once in the hierarchy, and if the "remaining" node is not explicitly hidden[3], the total sum of this aggregate can be determined.

[3] That a hierarchy should not have any "remaining" nodes is a Release 2.0B feature. If the "remaining" node is hidden it means that the value for the unassigned characteristic values is not calculated.

3 Maintaining Aggregates

It is guaranteed that queries always deliver consistent data when you drilldown. This means that data provided when querying against an aggregate is always from the same set of data that is visible within an InfoCube. It is therefore recommended that you carry out certain actions after loading data or making changes to master data or hierarchies, so the new data becomes “visible”, for both InfoCubes and aggregates.

3.1 Activating and Filling

In order to use an aggregate in the first place, it must be defined activated and filled. When you activate it, the required tables are created in the database from the aggregate definition. Technically speaking, an aggregate is actually a separate BasicCube with its own fact table and dimension tables. Dimension tables that agree with the InfoCube are used together. Upon creation, every aggregate is given a six-digit number that starts with the figure1. The table names that make up the logical object that is the aggregate are then derived in a similar manner, as are the table names of an InfoCube. For example, if the aggregate has the technical name 100001, the fact tables are called: /BIC/E100001 and /BIC/F100001. Its dimensions, which are not the same as those in the InfoCube, have the table names /BIC/D100001P, /BIC/D100001T and so on.

When you fill an aggregate you load it with data. This action can only be triggered from the aggregate maintenance. Also note that an aggregate can be filled from the data of a larger aggregate that is already filled. This means that very highly summarized aggregates, as a rule, can quickly obtain data from other aggregates. In contrast, it can take a long time to build aggregates from the InfoCube. Because of this, all aggregates are filled in background processes. If there are several aggregates scheduled to be filled in one process, the program first determines a hierarchy sequence for the aggregates, which is processed sequentially. This process guarantees that very highly summarized aggregates are built from the more detailed aggregates.

In BW version 2.0B it is possible to determine the data type and therefore the table space for the aggregates of an InfoCube (in the aggregate maintenance: *Extras > Change Data Type for Aggregates*). If the data type is not determined, the data type of the InfoCube is used, (as with the previous release) and the InfoCube’s tablespace will be used when the aggregate’s tables and indices are created. Assigning an aggregate’s objects to a user-defined table space can improve performance depending on the database, and according to disk layout considerations. With some databases, user-defined buffer pools (caches) can be created for specific tablespaces; this allows the aggregate tables to be buffered separately. Furthermore, separate tablespaces can be useful for back-up, administration and reorganizations.

3.2 Roll-up

New data packets / requests that are loaded into the InfoCube cannot be used at first for reporting if there are aggregates that are already filled. The new packets must first be written to the aggregates by a so-called “roll-up”. In other words, data that has been recently loaded into an InfoCube is not visible for reporting, from the InfoCube or aggregates, until an aggregate roll-up takes place. During this process you can continue to report using the data that existed prior to the recent data load. The new data is only displayed by queries that are executed after a successful roll-up. See the attachment for more details on the technical process of a roll-up.

To be able to use data recently loaded into the InfoCube in Reporting, you must carry out the following steps:

1. Firstly, the system automatically runs a technical check on the loaded data packet. At the same time, the dataset requested is matched with the set that was actually transferred.

2. In 2.0B you can check the quality of the data that was loaded. To do this, in the “Administrating” screen (context menu *InfoCube Administration* in the Administrator Workbench), choose *Environment* > *Request Handling*. In the next screen, remove the flag on “Set Quality Automatically”. Before you release the new data for Reporting, you can check it with specific queries, for example. These queries typically contain a variable for the number of the data packet/request as a filter for the characteristic “0REQUID”. You can also attach these queries to the exception Reporting.

3. Finally, the roll-up of this packet is scheduled. This step can also be automatic.

There are three different strategies for rolling up data:

1. Start the roll-up manually. To do this, choose *Roll-up* from the context menu *Administration* of the InfoCube in the Administrator Workbench[4]. This process is appropriate if the data of several packets form a logical unit and only make sense if they are released together. An example would be if different plants deliver their data at different times, but the data should only be visible if all plants have loaded their data into the InfoCube.

2. You can also set up the InfoCube so that every data packet is automatically rolled up into the aggregate if it is technically correct and the quality has been ensured. To do this, in the screen where you can schedule the manual roll-up, choose *Environment* > *Request Handling*. In the next screen, select “Automatic Roll-up”. However, this mechanism does not work if you are loading several data packets in parallel. Use the next process in this case.

[4] You can get to the same screen with Transaction *RSICUBE* when you select an InfoCube there.

3. You can also run the roll-up with the program RSDDK_AGGREGATES_ROLLUP. You can either schedule this program periodically or attach construct an event chain to include aggregate roll-up[5].

3.3 Change Run

If you change the master data, navigation attributes or hierarchies usually change, too. It is therefore recommended that you adjust the data in the aggregates after you load the master data. So that Reporting delivers consistent results, the master data and hierarchies are in two versions: in the active version where you can see the query, and in a modified version, which at some point will become the active version. The change run (also called hierarchy-attribute realignment run) adjusts the data in the aggregates and turns the modified version of the navigation attributes and hierarchies into an active version. In almost every phase of the change run, you can carry out Reporting on old master data and hierarchies.

You can either start the change run manually in the Administrator Workbench (Admin. Workbench: *Tools > Hierarchy/Attribute Changes*) or with the program RSDDS_AGGREGATES_MAINTAIN. You can give the program a list of characteristics and hierarchies that are to be taken into account for the change run. You can also schedule this program periodically or attach it to event chains. You can also save the list of characteristics and hierarchies as variants and treat them as variants when scheduling. If you do not enter anything, all those characteristics are taken into account whose master data you loaded or changed manually, and all the hierarchies that were marked for the realignment run. If you changed a hierarchy, you must activate this change. When you load hierarchies you can set whether you want the hierarchy to be directly activated or marked. If there are no aggregates that contain these hierarchies, the hierarchy is directly activated. If there are aggregates, a popup appears asking you if you want to delete the aggregates in question and directly activate the hierarchies, or whether the changes should just be marked. When you load and at the same time activate the hierarchy, it is marked for the change run as long as aggregates exist.

The change run takes places in different phases:

1. Firstly, all navigation attributes and hierarchy levels are generated that have actually changed[6]. The relevant aggregates that are to be adjusted are generated from this list.

[5] Event chains can only be defined as of 2.0.

[6] Not every change necessarily leads to the modification of an aggregate. For example, if you change a hierarchy by shifting a leaf to another node, it does not mean it affects nodes that are very high up in the hierarchy. Consequently, you do not always have to adjust the aggregate on a high level.

2. The system adjusts the aggregates and calculates the percentage of change per aggregate. If the percentage is small, the changes are calculated explicitly. If the percentage is great, the aggregate is rebuilt in a temporary table. In 2.0, you can set after what change rate the aggregate should be changed, in BW Customizing (*Tools > Business Engineer > BW Customizing*) (*SAP Reference IMG > BW Customizing Implementation Guide> Business Information Warehouse > General BW Settings > Maintain Aggregate Realignment Run*). Previous tests have shown that this value should be between 10 and 20 percent.
   The delta process was implemented later in System 1.2B. To use it, see the OSS note 181944.

3. Master data and hierarchies are activated[7].

4. The temporary aggregate tables are renamed. With databases that do not support the renaming of indices, the indices are rebuilt at this stage. In this very short phase, if the indices are not rebuilt, you cannot execute a query.

4 Maintenance Optimization

You can monitor the performance of the maintenance functions with BW statistics. The statistical data on the aggregates is stored temporarily in the table RSDDSTATAGGR, before it is loaded into the BW statistics InfoCube. You also look in this table (transaction *SE16*) if you want a quick analysis of how a particular roll-up or change run has behaved. If performance worsens over time, it may be that the amount of data has increased, or that indices have degenerated. In the latter case, you must reorganize the secondary indices of the aggregate fact table using databases.

4.1 Filling

If your system has adequate resources (number of processors, number of background processes, enough temporary table space PSAPTEMP, enough main memory) you can fill several aggregates at the same time in several background processes. To do this, select part of the aggregate you want to fill, schedule it for filling, select another part, and schedule this for filling too. Repeat the process. As a result, the subpackets are filled in parallel. Think about the hierarchical structure of the aggregate beforehand, and schedule only complete branches.

Test whether the database option “Parallel Query” speeds up the filling process.

[7] With databases that only have the “Dirty Read” mode, Reporting is not possible in this phase of the change run.

4.2 Change Run

You optimize a change run by defining basic aggregates for aggregates that contain hierarchies or navigation attributes that alter frequently. Instead of attributes or hierarchies, these basic aggregates contain the relevant characteristic grouped according to values. The advantage is, that the change run must not adjust itself to the basic aggregates, but the data for the aggregate that needs to be adjusted, can be read in an already summarized form from the basic aggregate. It is only useful to use a basic aggregate, however, if it is considerably smaller than the InfoCube. You can also use hierarchies and attributes that do not alter frequently in a basic aggregate. This can, however, mean long run times for the change run, if these attributes or hierarchies do actually change at some point.

5 Finding Efficient Aggregates

You can and should define a lot of aggregates for an InfoCube. However, when you create aggregates, you must always maintain a balance between cost and use. On the one hand, aggregates improve query performance, but on the other hand, they have a negative effect on the loading times due to roll-up and change runs. Optimal aggregate structure depends on how often a query is executed, how important performance is to the execution of a query, and how often transaction data arrives in the system. The questions of how up-to-date the master data and hierarchies have to be, and how often the change run is carried out (daily, weekly, monthly, ...) are also important.

Optimization of an InfoCube consists of several steps. Create the first aggregates after you have created the InfoCube and the queries, to ensure adequate performance for the first test. Continue to check whether additional aggregates are needed, or whether aggregates that you have created are no longer used at all. Delete aggregates that are no longer used to minimize maintenance costs.

5.1 Rules for Efficient Aggregates

From 2.0, the “Valuation” column evaluates each aggregate as either good or bad. The valuation starts at “+++++” for very useful, to “-----” for delete. This valuation is only meant as a rough guide. For a more detailed valuation, refer to the following rules:

1. An aggregate must be considerably smaller than its source, meaning the InfoCube or the aggregate from which it was built. Aggregates that are not often affected by a change run have to be 10^8 times smaller than their source. Other aggregates have to be even smaller. The number of records contained in a filled aggregate is found in the "Records" column in the aggregates maintenance. The "Summarized Records (Mean Value)" column tells you how many records on average have to be read from the source, to create a record in the aggregate. Since the aggregate should be ten times smaller than its source, this number should be greater than ten.

2. Delete aggregates that are no longer used, or that have not been used for a long time. The last time the aggregate was used is in the "Last Call" column, and the frequency of the calls is in the "Number of Calls" column. Do not delete the basic aggregates that you created to speed up the change run. Do not forget that particular aggregates might only not be used at particular times (holidays, for example).

3. Determine the level of detail you need for the data in the aggregate. Insert all the characteristics that can be derived from these characteristics. For example, if you define an aggregate on a month level, you must also include the quarter and the year in the aggregate. This enhancement does not increase the quantity of data for the aggregate. It is also only at this point, for example, that you can actually build a year aggregate from this aggregate, or that queries that need year values are able to use this aggregate.

4. Do not use a characteristic and one of its attributes at the same time in an aggregate. Since many characteristic values have the same attribute value, the aggregate with the attribute is considerably smaller than the aggregate with the characteristic. The aggregate with the characteristic and the attribute has the same level of detail and therefore the same size as the aggregate with the characteristic. It is however affected by the change run. The attribute information in the aggregate is contained in the aggregate only with the characteristic using the join with the master table. The aggregate with the characteristic and the attribute saves only the database – join. For this reason, you cannot create this kind of aggregate. If they are ever going to be useful, since otherwise the database optimizer creates bad execution plans, you can create an aggregate of this kind in the expert mode (in 2.0B: In the aggregate maintenance select an aggregate: *Extras* > *Expert Mode*, otherwise enter "EXPT" in the OK code field).

[8] The factor 10 used in the following, is only meant as a rule of thumb. The exact value depends on the user, the system, and the database. If, for example, the database optimizer has problems creating a useful plan for SQL statements with a lot of joins, aggregates with smaller summarization are also useful, if this means that joins are saved.

5.2 Estimating the Size of Aggregates

It makes sense to have an idea of the size of an aggregate directly when you are defining an aggregate. In this way you avoid defining aggregates that have only a very small summarization factor, and are as large as the InfoCube, or aggregates that already exist. The size of an aggregate that is built from another aggregate is smaller than or equal to the size of the source. If there are aggregates in the system that have already been filled, and you define a new aggregate, you get a rough estimate of the size: From the size of the aggregate from which the new aggregate is built, you get an idea of the maximum possible size, and from the size of the aggregate that the new aggregate can use as a source, you get an idea of the minimum possible size.

You get a good estimate of the size from knowledge of the data in the InfoCube and its distribution. This is shown in the following example:

Imagine you have an InfoCube, the size of which is determined by the characteristics ‘customer’ and ‘article’. There are 100,000 customers and 100,000 articles, but not every customer buys every article. The fact table does not therefore contain more than 100,000 x 100,000 = 10,000,000,000 records. The articles can be grouped into 100 article groups and 10 distribution lanes. The 100,000 customers are divided between 100 regions and 10 countries. A rough estimate of the size of the possible aggregates is given in the following table.

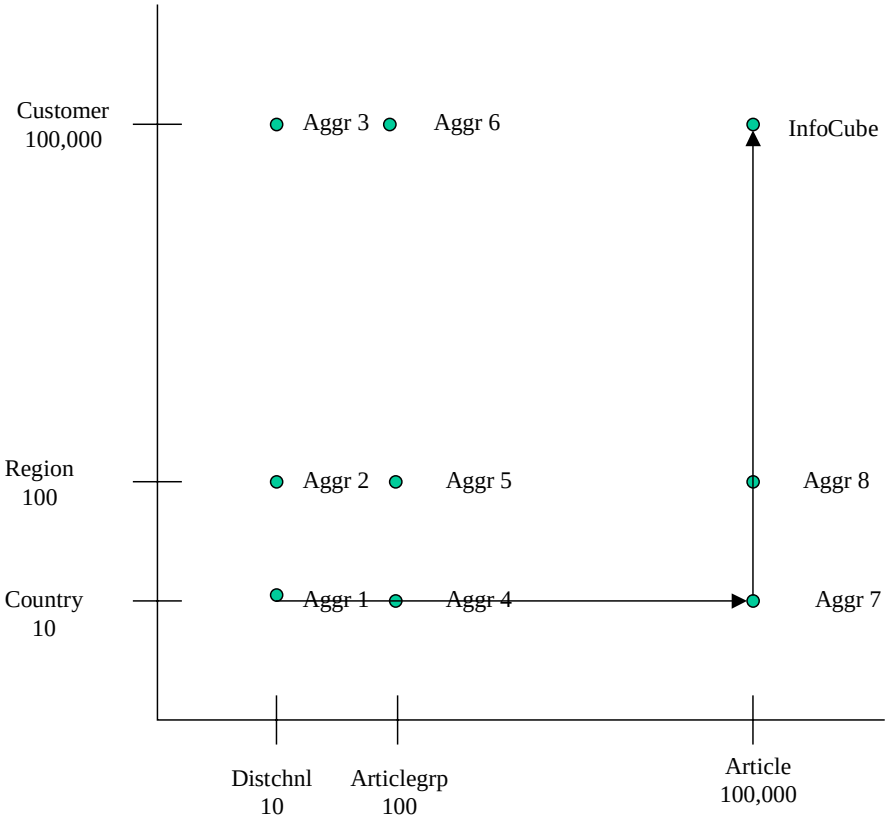
Aggregate	Combination	Max. Entries
1	Distrib.chnl: ‘*’, Country: ‘*’	100
2	Distrib.chnl: ‘*’, Region: ‘*’,	10,000
3	Distrib.chnl: ‘*’, Customer: ‘*’,	1,000,000
4	Article group: ‘*’, Country: ‘*’	1,000
5	Article group: ‘*’, Region: ‘*’	10,000
6	Article group: ‘*’, Customer: ‘*’	10,000,000
7	Article: ‘*’, Country: ‘*’	1,000,000
8	Article: ‘*’, Region: ‘*’	10,000,000
9	Article: ‘*’, Customer: ‘*’	10,000,000,000

Estimating the Size of Aggregates

Note that the InfoCube and the large aggregates are considerably smaller than the estimates, since not every combination is realized. The more summarized an aggregate is, the closer the values are to the true size, since the probability increases that, for example, every distribution channel is used in every country. Enhance all size estimations with correction terms resulting from knowledge of your concrete scenarios.

You get more exact size estimations if you take suitable select statements that determine the number of records.

You display the relationship between potential aggregates in the following diagram. The dots correspond to the potential aggregates. The volume of the rectangle given by the dot as the upper right corner and the origin as the lower left corner corresponds to the size of the aggregate. (Notice that the diagram has a logarithmic scale). If you draw the probable drilldown path of the queries on the diagram, you can use it to predict during which steps you might expect to encounter performance problems and which aggregate you should define.



Aggregate & Drilldown Paths

5.3 Efficient Aggregates for an InfoCube

The previous section described how you use knowledge of data distribution and possible drilldown paths to identify efficient aggregates. The tool also acts as a pointer for finding aggregates after you have created the InfoCube and the queries. In the aggregate maintenance (context menu *Maintain Aggregates* for an InfoCube in the Administrator Workbench) you get the tool to generate a proposal for aggregates from the existing queries.[9] The tool proposes two aggregates for each query. The query needs the first aggregate, with the name "MIN" and a number, to process the start list. The second aggregate, with the name "MAX" and a number, is designed so that the data can be read from this aggregate for every navigation step. The danger with the "MAX" aggregates is that they are exactly the same size as the InfoCube. This is particularly the case if a query contains a lot of free characteristics. If a query does not have any free characteristics, then the aggregates "MIN" and "MAX" agree. The system identifies similar aggregates, and counts how often a particular aggregate is used. The number of potential calls gives an initial indication of how important an aggregate is.

If the system has been running for some time already, analyze which aggregates are really being used, and where exactly there are performance problems. Delete aggregates that are not being used, because they cause an unnecessary overload during the loading process. To optimize the system, create any extra aggregates that are needed. You must repeat this process at regular intervals, since the behavior of the end user may change. Queries that are important today may not be interesting tomorrow. If you collect BW statistical data[10] for the InfoCube, the tool can support you with this process. Aggregates can also be proposed from collected statistical data.[11] These proposals are all given a name that starts with "STAT" and a number.

---

[9] Execution path of the function: Aggregate proposal from existing queries
2.0B: *Proposal* > *Proposal from Query* in the aggregate maintenance
2.0A: Dropdown menu *Proposals* > *Analyze Queries*
1.2B: *Aggregates* > *Propose Aggr.* and *Queries* pushbutton in the following screen

[10] You turn on the BW statistical data collection function in the Administrator Workbench using *Tools* > *BW Statistics* for InfoCubes. You must select the "OLAP" column to retain data on the executed queries.

[11] Execution path of the function: Propose aggregates from statistical data
2.0B: *Proposal* > *Propose from BW Statistics (DB)* or *Propose from BW Statistics (InfoCube)* in the aggregate maintenance. With the first function, the records that are in the collection tables directly on the database are referred to. With the second function, only the statistical data that has already been loaded into the BW statistics cube is referred to.
2.0A: Dropdown menu *Proposals* > *BW Statistics (Database)* or *BW Statistics (InfoCube)*. The function works in the same way as for 2.0B.
1.2B: *Aggregates* > *Propose Aggr.* and the *Statistics* pushbutton on the following screen. The statistical data from the BW statistics InfoCube are taken, meaning that you must have already loaded the data from the collection tables into the InfoCube before you use this function.

Regardless of which of the two methods you used to generate proposals, you must always check the proposals, and only activate or deactivate/delete the ones where you think it would be useful to do so. Avoid deleting basic aggregates that you created to optimize the change run. When you finish working, think again about the basic aggregates – which extra basic aggregates are required, which are no longer used? When you are analyzing the proposals, remember that the proposals minimize only the reading costs. You have to minimize the update costs (roll-up, change run) yourself, by summarizing the various proposals into a single, perhaps larger, aggregate, or by not activating some of the proposals. The aggregate hierarchy that tells you which aggregate can be built from which other aggregates, is important in helping you with the analysis. In Release 2.0, you see this directly in the system (in the aggregate maintenance button with the hierarchy symbol). In a 1.2B system you build the hierarchy graphically yourself on a sheet[12]. If you think the list of aggregates is too long, and you have no idea, which aggregates you do not want to activate, use the "optimize" function (*Optimize the aggregate hierarchy* button in the aggregate maintenance, or in the "Optimize aggregation selection" screen for 1.2B). This function deletes the 20% of the proposals that are used the least, as long as they can be replaced by a slightly larger aggregate. The number of calls is added to the slightly larger aggregate.

6 Queries and Aggregates

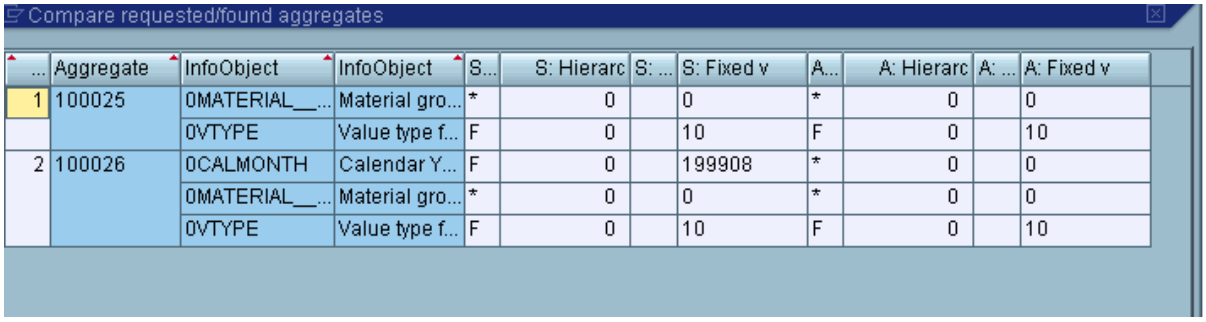
The previous chapter discussed the options for optimizing an InfoCube globally. There is also the issue of creating an aggregate for an individual query – for an individual navigation step. In this chapter, you are introduced to the tools that can support you with this. When you are analyzing individual queries, you do not want to lose the overview of all the aggregates for the InfoCube, since you might otherwise get high update costs. If you have several aggregates resulting from the query analysis, consider the points mentioned in the previous chapter again, and summarize several aggregates into a single, larger aggregate if you think it is necessary.

[12] You can determine the hierarchy of the data that has already been filled by looking at the information for every aggregate in the aggregate maintenance (*Information* button). The technical key for the source aggregate is in the "PARENTUID" field. You find the technical key of an aggregate in the "Aggr. UID". If the "PARENTUID" field is empty, this means that the aggregate is filled from the InfoCube.

6.1 Analyzing a Query – a Navigation Step

There are two tools that are used to analyze which aggregate is suitable for which navigation step. You optimize the start screen of a query with the query Monitor (transaction *RSRT*) if you execute the function *Execute + Debug*. With the trace tool (transaction *RSRTRACE*) you can draw several navigation steps, and process them again using the function *DB Debugging* (exact path: Transaction *RSRTRACE* > *Goto* > *User Logs* or *All Logs* > double-click on the log you want > *DB Debugging*). The OSS note 99010 gives you a more detailed description of the trace tool. Both tools behave basically in the same way. With the query monitor, you optimize the start screen easily. With the trace tool, you optimize several navigation steps or queries from third parties (using the MDX interface). To do this, you need to have a log first.

After you have started the query Monitor or the trace tool as described above, a popup appears in which you must select the row “Show found aggregate”. From the screen, determine the aggregates that are needed for this step, and which aggregates are actually taken. In 2.0 the screen looks like this:


Compare requested/found aggregates

...	Aggregate	InfoObject	InfoObject	S...	S: Hierarc	S: ...	S: Fixed v	A...	A: Hierarc	A: ...	A: Fixed v
1	100025	0MATERIAL__...	Material gro...	*	0		0	*	0		0
		0VTYPE	Value type f...	F	0		10	F	0		10
2	100026	0CALMONTH	Calendar Y...	F	0		199908	*	0		0
		0MATERIAL__...	Material gro...	*	0		0	*	0		0
		0VTYPE	Value type f...	F	0		10	F	0		10

On the database, a navigation step is divided up into several subqueries by the OLAP engine. The characteristics and navigation attributes that are needed for the subqueries are in the rows. The columns that are labeled with “S:” describe the optimal selection, and the columns that are labeled with “A:” describe the found aggregate. The aggregate name is found in the “Aggregate” column. In this example, two subqueries are sent to the database. The first requires an aggregate (0MATERIAL_GROPUP * ; 0VTPYE F = 10), and it reads the data from the aggregate with the technical name 100025. This aggregate is the same as (0MATERIAL_GROPUP * ; 0VTPYE F = 10). The second subquery needs the aggregate (0CALMONTH F = 8.1999; 0MATERIAL_GROPUP * ; 0VTPYE F = 10) and it reads from the aggregate 100026 that is the same as (0CALMONTH * ; 0MATERIAL_GROPUP * ; 0VTPYE F = 10).

The OSS note 166433 describes how this information is contained in a 1.2B system.

If you also selected the row ‘Display statistical data’ on the first popup, you then get the BW statistical data for this query. The columns “QDBSEL”, “QDBTRANS” and “QTIMEDB” are important here for optimization. “QDBSEL” shows how many data records the database has read and “QDBTRANS” shows how many data records have been transferred from the database to the application server. In the field “QTIMEDB” the time is entered, that was required by the database for the selection, aggregation and transport. All other fields of the BW Statistics are described in OSS note 130696. The relationship of QDBSEL to QDBTRANS shows how many records the database still had to aggregate. If this relationship is large (greater than 10) and the database time is long, then the navigation step can be accelerated by an aggregate. If the relationship of “QDBSEL” and “QDBTRANS” is close to 1, then this query is not accelerated by an aggregate.

6.2 Testing the Efficiency of Aggregates[13]

If you created and filled aggregates, but are unsure as to whether it makes sense to have a particular aggregate, then you can switch off the aggregate in the aggregate maintenance (select aggregate: *Aggregate > Switch On/Off*). With aggregates that are switched off you see a gray light in place of a green one. The effect of this is that the aggregate is no longer used by queries. Aggregates that are switched off, however, are still kept consistent. They can thus be used again directly as and when they are switched back on again. The advantage over deactivating or deleting is that the aggregate must not be refilled again, and switching off is thus particularly well suited to simple testing. As described in the previous section, you can now execute a query or trace which would really use the aggregate. Compare the time that is required by the database to the time that is required by the query using the aggregate. If the query is not considerably slower then you can deactivate or delete the aggregate.

6.3 Read Mode of a Query

The read mode determines which data the OLAP Engine gathers from the database and at which time. There are three different modes that you can set in the query monitor (transaction *RSRT > Read mode*), whereby the third mode is the default setting for all queries:

[13] This function does not exist in Release 1.2B. In this instance you would have to deactivate the aggregate.

1. "Total Read of the Data":
   When executing the query in the Business Explorer all the data, that would be required for all possible navigation steps of this query, is read into the main memory area. The dataset to be read is determined by the smallest level of detail of the query. When navigating within the query the required data is then read from the main memory and no longer from the database. The OLAP Engine aggregates the data. This read mode should only be used if the costs of a database access are extremely high. This read mode does not use the full potential of the aggregates since a great deal of internal aggregation must take place.

2. "Reading Data on Demand":
   For every navigation step of the query in the Business Explorer, the OLAP Engine requests only the required data for each step. The data is reused from the main memory for each navigation step, or new data is read from the database. By doing this, the best aggregate table is used for each step and is already aggregated as far as is possible on the database.

3. "Reading on Demand when Expanding the Hierarchy":
   When "reading the data on demand" (2), the data for the entire hierarchy is requested for a hierarchy drilldown – i.e., the data is read at leaf level. When "reading on demand when expanding", the OLAP Engine gathers only the values for nodes from the database, that are also actually displayed. When expanding a hierarchy node the children of the node are then read on demand respectively. Otherwise this mode behaves just like the simple "read on demand" (2).

The read modes from 1 to 3 differ in that more and more database accesses take place, for which, however, the amount of read data becomes smaller and smaller. The mode "reading on demand when expanding" thus supports the concept of aggregates the best. Also note that, with the third mode, the total data read is generally smaller than with the other two modes, since a user rarely looks at all the data that goes into a query.

You can understand the way the different read modes work if you include a trace for navigation in a particular query, and then run this trace with the different read mode – settings of the query, and look at the aggregates used in the process.

6.4 Effects of the Query Design on Aggregates

To give aggregates optimal use, the queries should be created in such a way that they start at an aggregated level. The user then has the option of filtering certain interesting characteristic values and of drilling down according to other characteristics. If efficient aggregates exist in such a scenario then, for each navigation step, a dataset of roughly equal size must be selected from the database. With the more and more restrictive filter when drilling down, it is of little significance to the performance that the data sources (aggregates or InfoCube) become larger and larger. In this scenario the reply times thus remain roughly the same.

6.4.1 Query Definitions with Particular Effects on Aggregates

The following constructs in the query definition have the effect that other, bigger aggregates are required than you might first suppose:

6.4.1.1 Key Figures with Exception Aggregation

If a key figure is used in a query with exception aggregation then the data is always read from the database at the level of detail of the exception reference characteristic since the OLAP Engine executes the exception aggregation. This means that an aggregate must be grouped by values of the exception reference characteristic. The reference characteristic must therefore be included with '*' in the aggregate.

For this reason the reference characteristic is automatically included in all aggregates if the InfoCube contains a key figure with exception aggregation. If not all queries of the InfoCube contain key figures with an exception aggregation then you can also create aggregates that are summarized using the exception reference characteristic. For this you have to use the expert mode (in 2.0B: select an aggregate in the aggregate maintenance: *Extras > Expert mode*, otherwise enter "EXPT" in the OK-Code field).

6.4.1.2 Calculating with Attributes in Calculated Key Figures[14]

If a query contains a calculated key figure with a formula variable that is to be replaced by an attribute value, then the data is read by the database grouped according to the characteristic belonging to the attribute. This is necessary since the calculation takes place in the OLAP Engine before aggregation[15].

6.4.1.3 Virtual Characteristics and Key Figures

If the query uses the exit "virtual characteristics and key figures" (enhancement RSR00002) then the data is read at the level of detail, that the data requires in the exit.

6.4.1.4 Calculating before Aggregation

If the query contains a key figure that is calculated before aggregation, then no aggregate at all can be used.

---

[14] This feature exists as from 2.0B.

[15] If a formula variable such as this is used in a formula, then the data is not grouped extra for each respective characteristic, since the calculation is first made in this case after aggregation. For this, the formula is not calculated in rows in which aggregation is carried out with the characteristic.

6.4.1.5 Time-Related Currency Translation

If a time-related currency translation is active then the result is used by the database at the level of detail of the corresponding time characteristic.

6.4.1.6 Presentation Hierarchy and Filter

If a characteristic is drilled down with a presentation hierarchy, and if there is - at the same time - a filter on this characteristic from another hierarchy, an interval filter, or another complex selection, then the OLAP Engine drills down the nodes of the presentation hierarchy into leaves. In such an instance no aggregates can therefore be used with the presentation hierarchy, even if the read mode of the query is set to “read on demand when expanding the hierarchy”.

6.4.1.7 Hierarchies with Multiple Leaves; Hierarchies without Remaining Nodes[16]

If an aggregate contains a hierarchy that has multiple leaves or does not have any remaining nodes, then this aggregate cannot be summarized with the hierarchy for the total sum using the characteristic.

6.4.2 Recognizing Particular Query Definitions

This section describes how you can recognize the constructs above on the application server, without being able to use the BEx Analyzer. For this you go into the Query Monitor (transaction *RSRT*). If you execute the function *Import*, then you run a breakpoint. At this stage you can use all fields and tables described further below. From 2.0B, this information can be viewed directly under *Technical Info.*

6.4.2.1 Key Figure with Exception Aggregation

The reference characteristic is in field: *l_sx_report-rkb1f-aggrcha*. If the field is empty then the query does not contain any key figure with an exception aggregation.

6.4.2.2 Calculating with Attributes in Calculated Key Figures

The formula variables are in table *l_sx_report-var_lrech.* If the column *attrinm* is filled in a row, then the calculation is made with an attribute. The respective characteristic is located in the column *chanm*. However, by using this table, you cannot tell whether the variable is used in a formula or in a calculated key figure.

[16] “Hierarchies without remaining nodes” is a new feature for 2.0B

6.4.2.3 Virtual Characteristics and Key Figures

In table *l_sx_report-sfc* the characteristics that are required for the exit are marked with 'X' in the column *user_exit*.

6.4.2.4 Calculating before Aggregation

If calculation is made in a query before aggregation then the field *l_sx_report-rkb1f-vor_aggr* is equal to "X".

6.4.2.5 Time-related Currency Translation

The currency conversion types are in column *cttnm* of table *l_sx_report-cell.* If all cells in this column are empty then no currency translation is active. If a translation is active then you can find out the modalities of the currency translation type with the translation type maintenance (*BW Administration > Translation types > Display*).

6.4.2.6 Presentation Hierarchy and Filter

The selection conditions are in table *l_sx_report-sel*. In table *l_sx_report-startlist-dim* you can see the presentation hierarchy in field *hienm* and whether it is active in field *hry_active*. Note that this can be changed at runtime.

6.4.2.7 Miscellaneous

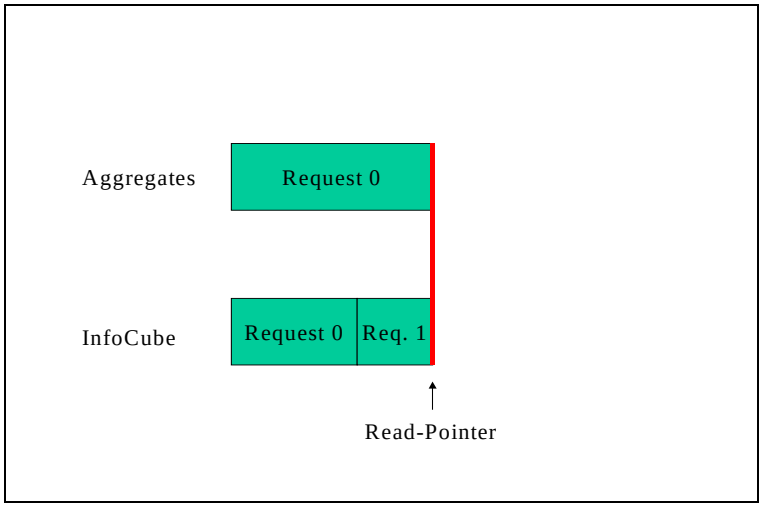
All characteristics that the OLAP Engine requires are marked with "X" in the *necessary* column of table *l_sx_report-sfc*. If the field *l_sx_report-rkb1f-summabel is equal to "X"* then no aggregates can be used.

7 Appendix

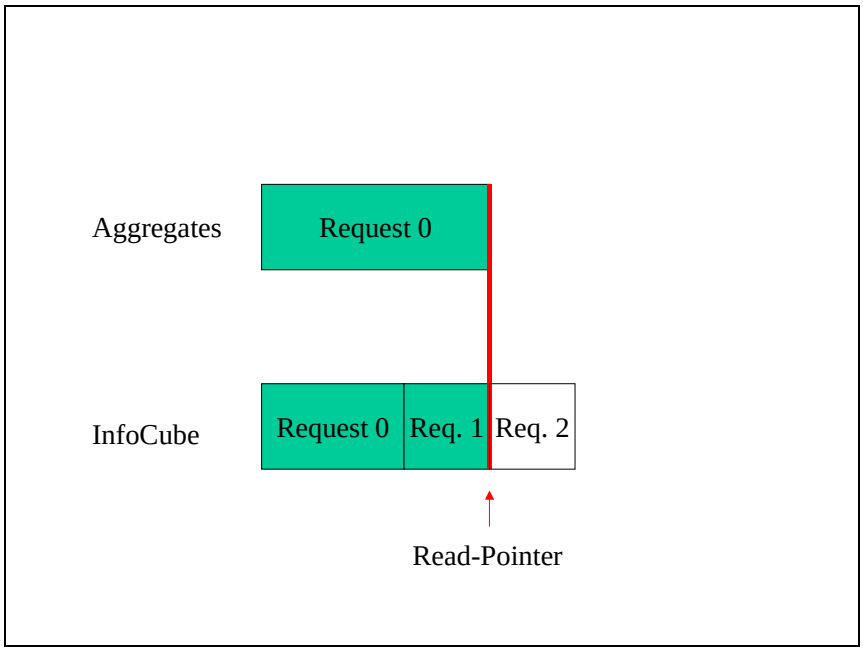
7.1 Roll-up and Consistency

The individual steps in the roll-up will be discussed in this section, and an explanation will be given as to how the consistency of the visible data in the InfoCube and in the aggregates is guaranteed.

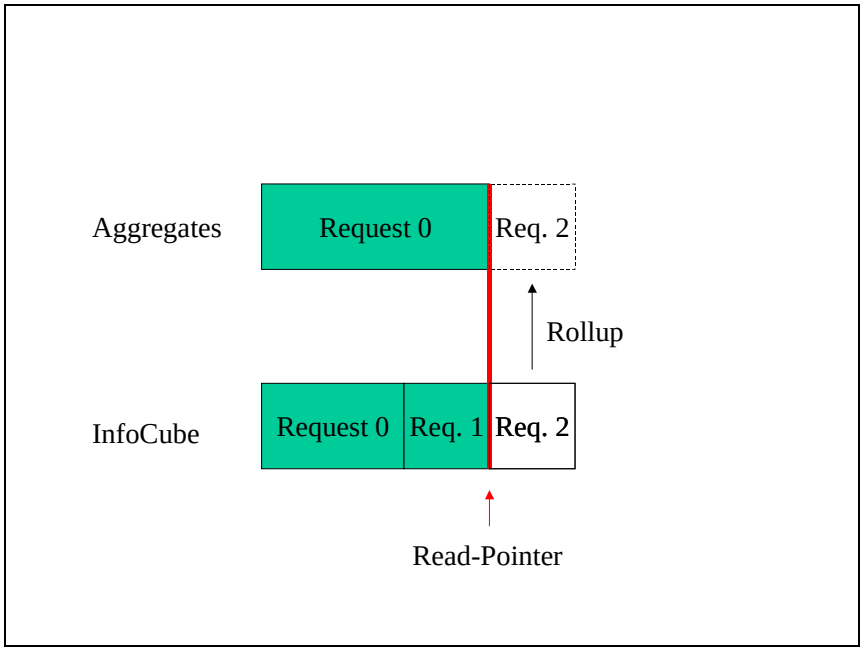
At the starting point of the analysis, the InfoCube and the aggregates contain the same data packets. The data is visible up to the Read Pointer. Technically this is a filter on the characteristic 0REQUID. This is located in the technical packet dimension that is contained in every InfoCube.



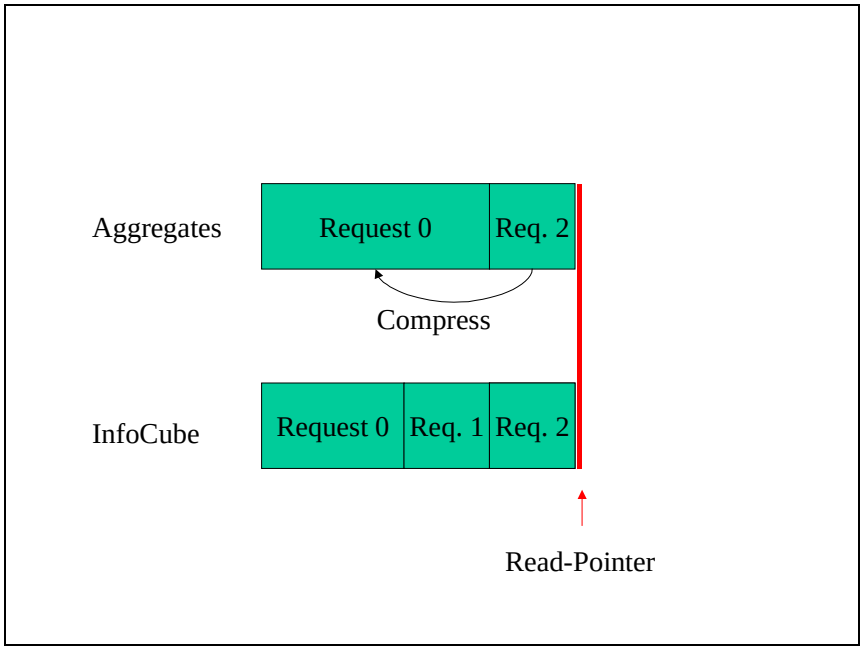
If new data is loaded into the InfoCube then it is written in the fact table without it being made available for Reporting, since the Read Pointer has not yet changed.



If the data was loaded successfully into the fact table of the InfoCube, it can then be rolled-up. Since the Read Pointer has not changed the new data is still not yet visible. Queries thus still show the result of the start of the process.



If the data packets have been rolled-up successfully into all aggregates of the InfoCube then the Read Pointer is set to the last rolled-up data packet. Queries started from this point display the new data.



Since the characteristic “0REQUID” also exists in every aggregate, records that logically belong together (records that have the same key in all non-technical dimensions) are saved physically in several records. So that the aggregate tables do not get too large, the data for the aggregates is compressed automatically after the successful roll-up, meaning that it is summarized over characteristic “0REQUID”. No query can be executed in this compression phase for databases that only support the “Dirty Read” mode.

It should be understood that, during a roll-up, further data could be loaded into the InfoCube.

7.2 Aggregates and Most Recent Data from the InfoCube[17]

In certain scenarios a query requires the use of aggregates, but should still display the most recent data from the InfoCube. You can achieve this if, in the query, you restrict the characteristic 0REQUID in the global filter to the variable 0S_RQMRC. You can find characteristic 0REQUID in the packet dimension. This restriction results in the data, already in the aggregates, being read from the aggregates. The data that is technically correct in the InfoCube, but that is not yet rolled-up, is read directly from the InfoCube. For MultiCubes the splitting is made for each sub-Cube.

[17] This feature is available as from 2.0B

7.3 Transporting Aggregates

In BW Release 1.2B aggregates cannot be transported. They must, therefore, be created manually in all systems that are subsequent destination systems of the development system. With BW 2.0 you can transport aggregates. With this, however, always bear in mind that how the user behaves in the test system does not have to correspond to how the user behaves in the development system. You should therefore not transport aggregates indiscriminately.

7.4 Checktool for Aggregates

If you would like assurance that the data in the aggregate is consistent to the InfoCube, then you can execute individual queries without using aggregates in the query monitor by marking in the aforementioned debug mode that the query should not use any aggregate. You can compare the result with that which appears when using aggregates.

This process can also be automated. You can include logs of queries with the trace tool and let it run twice: once using aggregates and once without aggregates. A program checks the results for similarity. The exact process is described in OSS note 202469.

7.5 Additional Information

You can find additional information on aggregates in the BW online documentation. In addition, there are various training courses offered by SAP, in which the topic of aggregates and the performance optimization of a BW system are addressed. Currently the training courses are as follows:

- BW205: Business Information Warehouse Analysis
- TABW90: SAP BW - Technical Administration
- A training course, offered by TCC, that focuses especially on optimizing queries by aggregates, is being prepared. It will be on offer from fall 2000

7.6 OSS Notes

Look occasionally at the SAPNET R3-Frontend OSS for notes with keywords such as aggregates and performance. Also look for notes that belong to the component BW-BEX-OT-AGGR. Here is a selection of useful notes.

- 099010: Description of the Trace Tool for RFC Connections
- 125681: Aggregates and Exception Aggregation
- 130696: Performance Trace in BW 12a (also valid for 1.2B and 2.0)

- 166433: Options for Finding Aggregates
- 176616: BW Statistics
- 181944: Accelerating the Hierarchy-/ Attribute-/ Change Run
- 192658: Basic Parameterization for BW Systems
- 202460: BW Statistical Data for Aggregates
- 202469: Using the Aggregate Check Tool