How to…

Performance Tuning with the OLAP Cache

BUSINESS INFORMATION WAREHOUSE

Applicable Releases: 3.x

Release date: September 2004

1 Business Scenario

A successful SAP BW implementation requires that end user query response times are consistently running in the fast to reasonable range, in order to gain user acceptance and help drive overall productivity. There are many factors that influence the response time of a BW query, including its complexity, the volume of data involved, architecture of the data model, and the existence of pre-summarized or pre-calcultated objects. A sound OLAP performance strategy includes the utilization of several of the important tools that are available for SAP BW tuning efforts, such as multi-provider, aggregates, and pre-calculated templates. Introduced in SAP BW version 3.0, the OLAP cache offers a valuable compliment to these other useful techniques used for accelerating query response times.

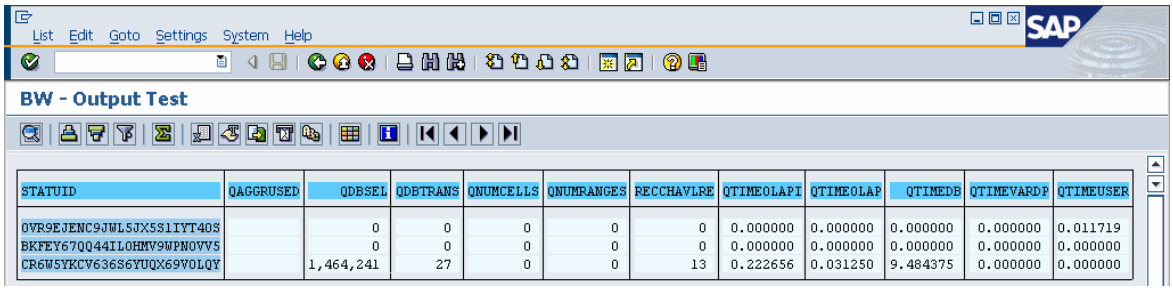
The OLAP cache is architected to store query result sets, and to give all users on an application server access to those result sets. If a user executes a query, the result set for that query's request can be stored in the OLAP cache. If that same query (or a derivative) is then executed by another user, the subsequent query request can be filled by accessing the result set already stored in the OLAP cache. In most cases, a query request filled from the OLAP cache is significantly faster than queries that receive their result set from database access.

A savvy performance tuning strategy could include scheduling background jobs to load specific query result sets into the OLAP cache. Therefore, initial user query executions against newly loaded data can "hit" and OLAP cache result sets in running queries, boosting performance significantly. For more information about performance tuning in general, see also SAP Service Marketplace alias /bw → Performance.

2 The Step-by-Step Solution

2.1 Description of the scenario

Choose a query to tune the response time. In this example, we will improve the query performance of query PM_PERFMEAS_01 using the OLAP cache. First, check the response time without OLAP Cache. In transaction RSRT, enter the query name and press 'Execute + Debug'. Selecting 'Display Statistics Data' and 'Do not Use Cache', will return a list that is comparable with the following:



STATUID	QAGGRUSED	QDBSEL	QDBTRANS	QNUMCELLS	QNUMRANGES	RECCHAVLRE	QTIMEOLAPI	QTIMEOLAP	QTIMEDB	QTIMEVARDP	QTIMEUSER
0VR9EJENC9JWL5JX5S1IYT4OS		0	0	0	0	0	0.000000	0.000000	0.000000	0.000000	0.011719
BKFEY67QQ44ILOHMV9WPNOVV5		0	0	0	0	0	0.000000	0.000000	0.000000	0.000000	0.000000
CR6W5YKCV636S6YUQX69VOLQY		1,464,241	27	0	0	13	0.222656	0.031250	9.484375	0.000000	0.000000

In this example, most of the overall query response time was spent in the database read operation, as the QTIMEDB was about 10 seconds. As the ratio of QDBSEL (number of selected records from database) to QDBTRANS (number of transferred records to OLAP) is very high, building an aggregate might be considered as an appropriate option for optimization. The maintenance cost of aggregates may be somewhat prohibitive, however, as rollup and change run require time and resources, and adds time to latency of data availiblity. Effective utilization of the OLAP cache can help minimize the number of aggregates that must be maintained in general, and offer a good alternative technique for boosting query performance. In this example here, an aggregate cannot be used as the query contains a key figure that is calculated before aggregation.

By simply configuring the cache, and then allowing random user activity to fill the cache with query result sets, general performance benefits will be realized for many queries. A managed approach, however, maximizes the resources of the cache and ensures consistently good end-user response times for many key queries. Active management of the OLAP cache entails using the reporting agent to run key queries and "warm up" the cache, after new data has been added to InfoProviders, but ahead of the first user query access to that updated data.

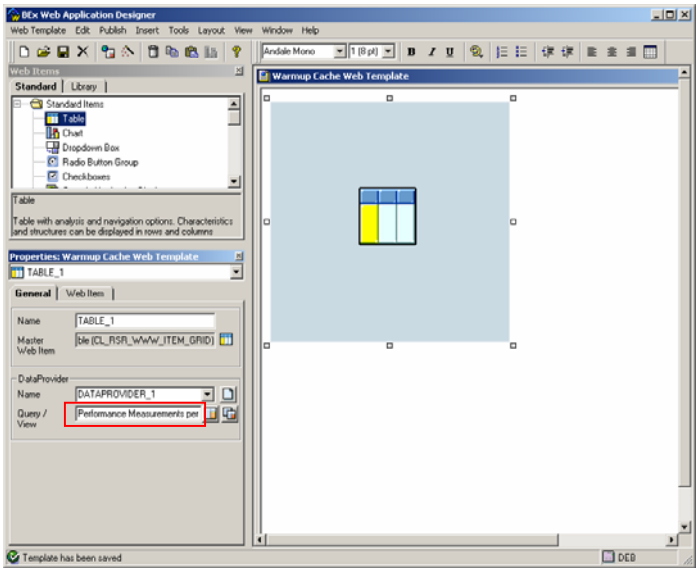
When building a strategy to warm up the cache, first identify which queries might be appropriate to include. These queries should be more general types, with minimal selection criteria, so that "superset" query result sets are stored in the cache. Other query navigations, where the query's request is more specific via selections, can be filled by accessing a "superset" cached query result set.

2.2 Warming up by Reporting Agent (Pre-calculating Web Templates)

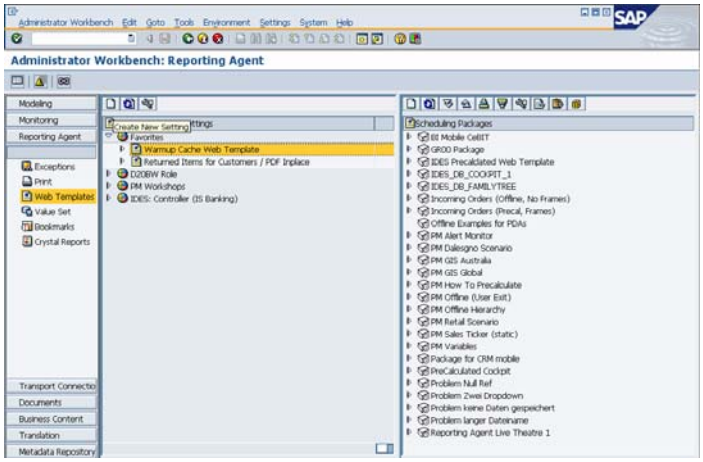
The technique in this example uses a simple web template, published for a query, and then it is scheduled to be run in the background using the reporting agent. The query has one variable. You can also use query views for your web template; this enables you to pre-calculate specific navigation paths.

Note, in SAP BW 3.5 you can also use Information Broadcasting instead of using the reporting agent.

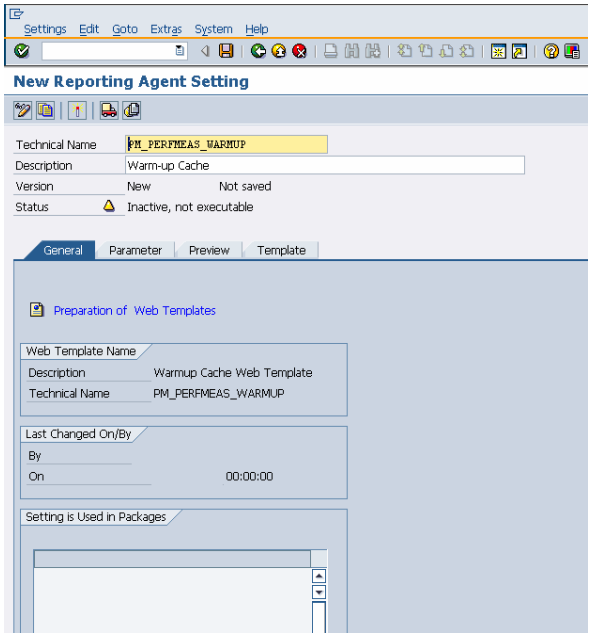
1. Open the Web Application Designer and define a web template. Drag & Drop a table item and assign your query/query view to the table. Save the web template (we use the name PM_PERMEAS_WARMUP).



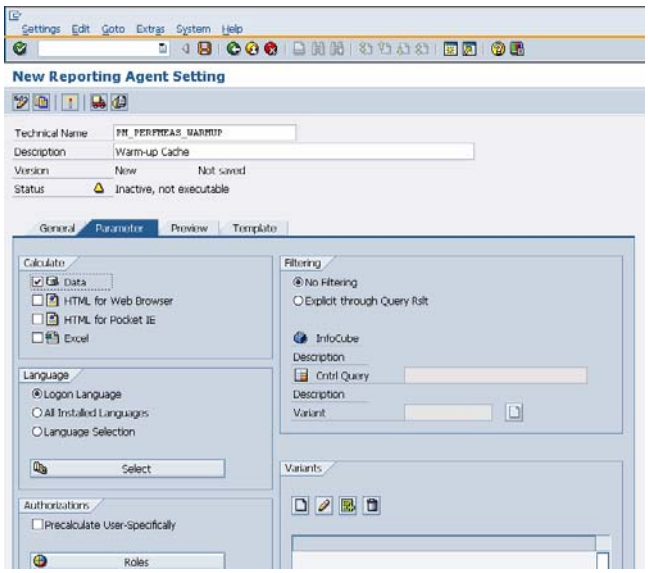
2. Now branch to the 'Web template' mode of the reporting agent, place the cursor on your web template and create a new setting.



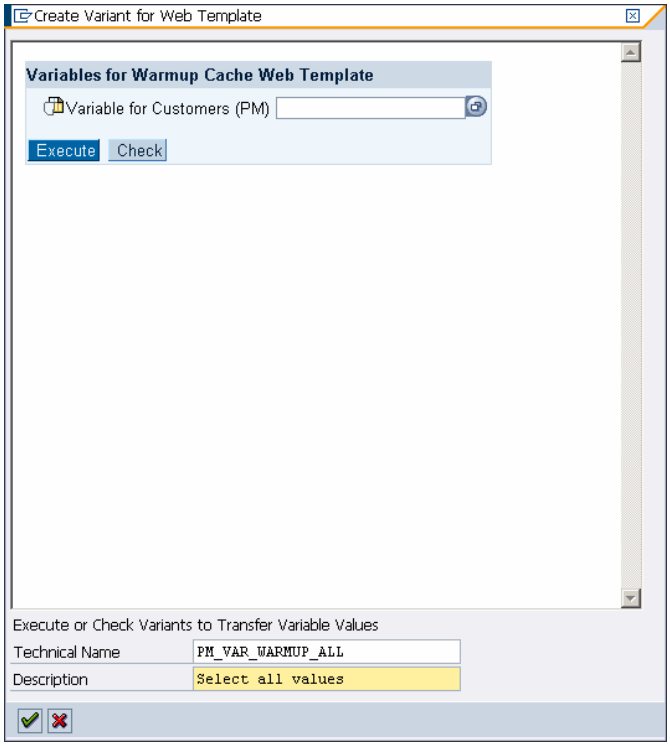
3. In the general screen, enter the technical name and description (we use PM_PERFMEAS_WARMUP)



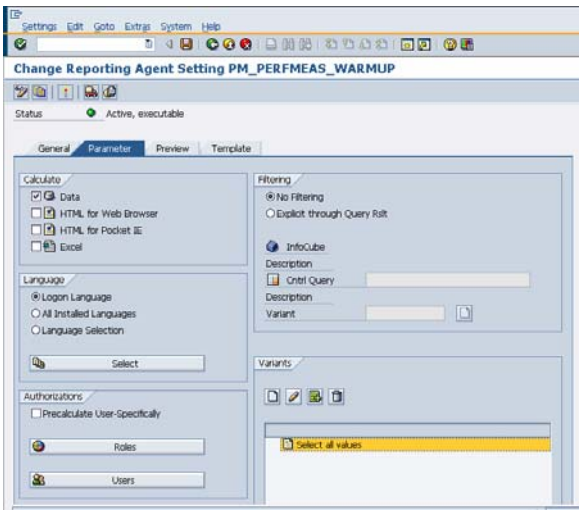
4. In the parameter screen, flag the 'data' field.



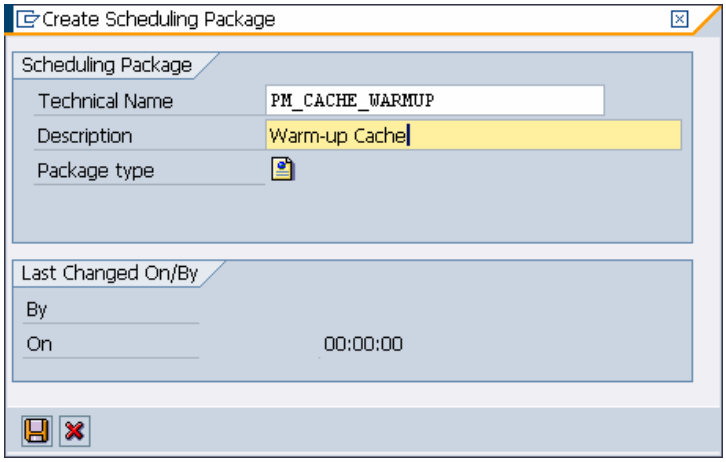
5. Create a variant that selects all values for the given variable.



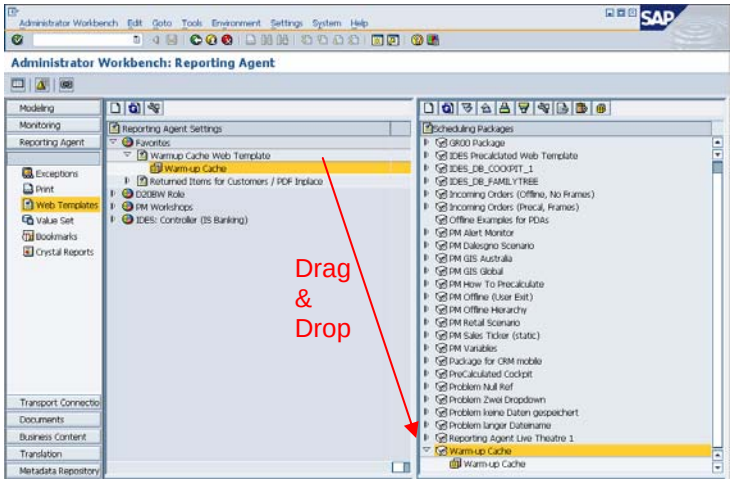
6. Activate the setting. For information on authorization and filtering settings, see the related How To paper on pre-calculation of web templates.



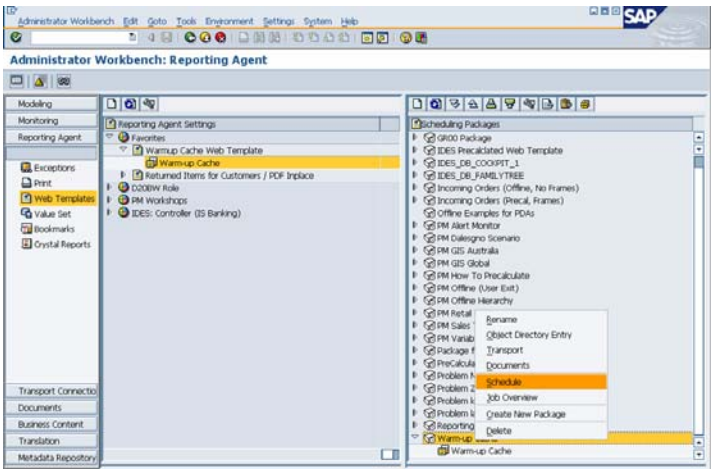
7. Now create a scheduling package in the right frame of the reporting agent. Enter a name (here: PM_CACHE_WARMUP) and save it.



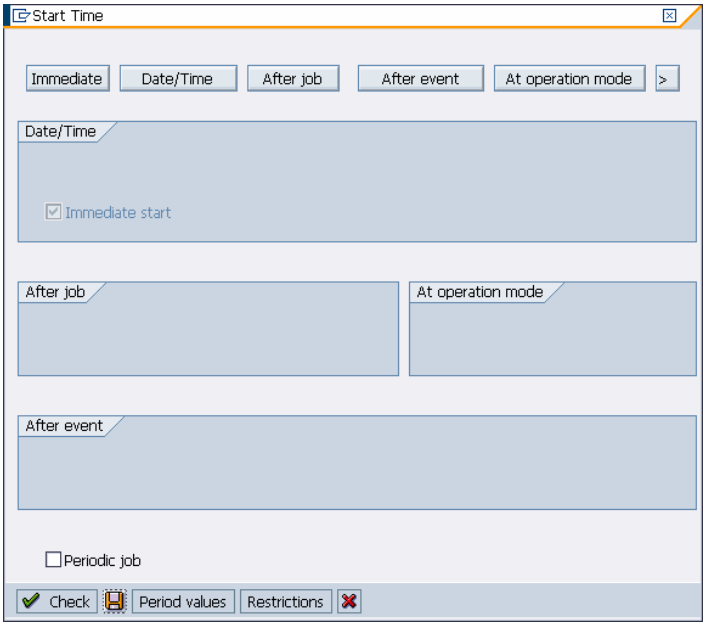
8. Now drag your reporting agent setting and drop it onto the scheduling package.



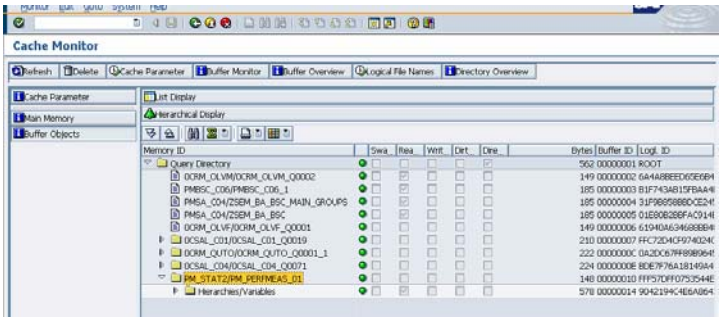
9. Place the cursor on the scheduling package and schedule the package via the context menu.



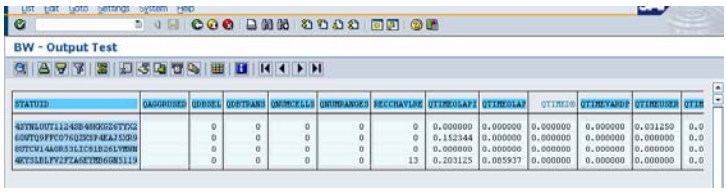
10. Enter the start condition (here: immediate) and save it. By saving the package, it will be scheduled.



11. After the pre-calculation, the query result is stored in the OLAP Cache. In order to check this, branch into the OLAP Cache Monitor (transaction RSRCACHE) and press 'Buffer Objects' on the left side. You should find your query as part of the buffer – of course, it could have been there before.

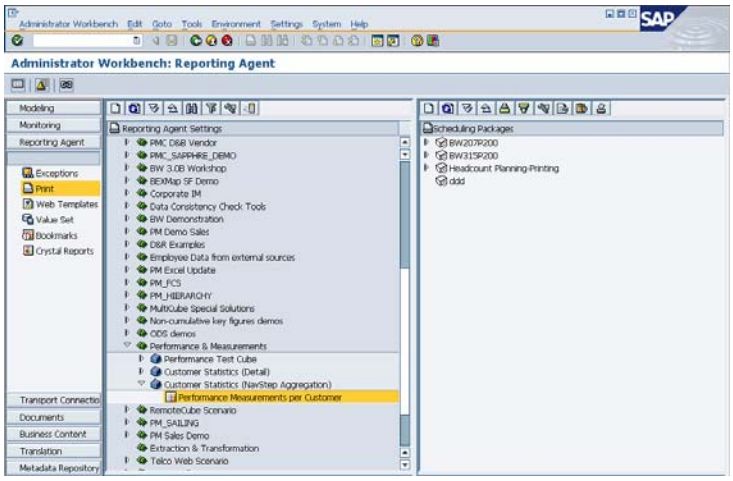


12. As a second check, call transaction RSRT again and execute your query in the debug mode in order to show the statistics for your query. The time spent on the database must be 0 now. In our case, the overall query response time has been reduced to less than 0.5 seconds.

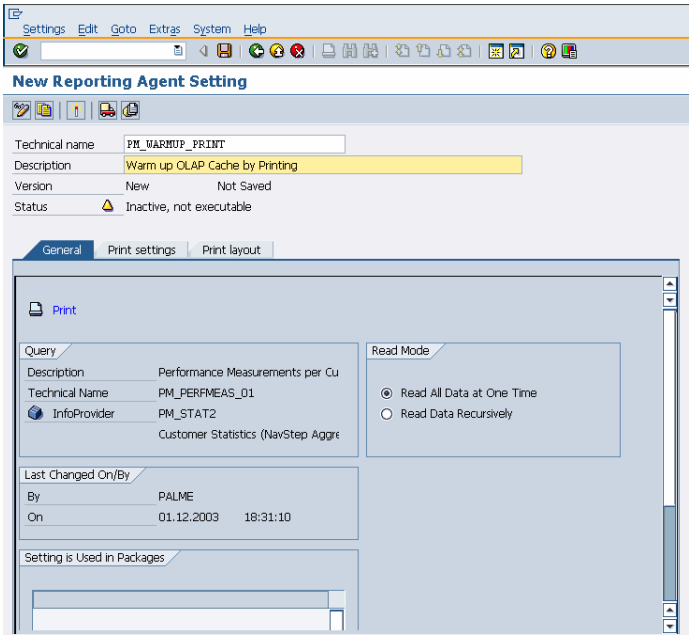


If you want to avoid defining web templates just for the OLAP Cache warm-up, you can also use the print function.

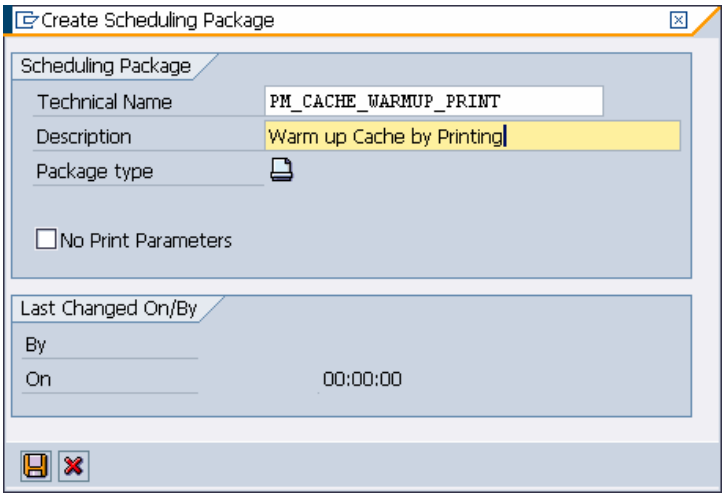
1. Go to the reporting agent's 'Print' mode and place the cursor on your query.



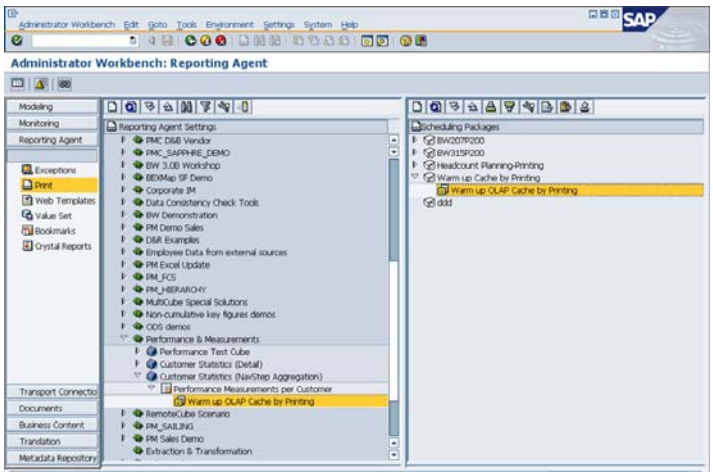
2. Enter a name (here: PM_WARMUP_PRINT) and activate your settings.



3. Now create a Scheduling Package in the right frame of the reporting agent, enter a name (here: PM_CACHE_WARMUP_PRINT) and save the package.



4. Drag your reporting agent setting and drop it onto your package.



5. Schedule this package as described above. Check the OLAP Cache Monitor.

3 Technical Information on the OLAP Cache

3.1 General

The OLAP Cache is used for caching transaction data and, as of SAP BW 3.0B SP22 / SAP BW 3.1 Content SP 16 / SAP BW 3.5 (SAP NetWeaver '04) SP Stack 4 also for caching master data.

3.2 Switching OLAP Cache on/off

The OLAP Cache can be switched off on four levels:

- Globally
  This can be done via the IMG: → Business Information Warehouse → Reporting-relevant settings → Global Cache Settings
- On InfoProvider level as default value for all subsequently created queries
  This can be done via the IMG: → Business Information Warehouse → Reporting-relevant settings → InfoProvider Properties
- On query level
  This can be done in transaction RSRT via the Properties button for a specific query
- For a unique execution of a query in the Query Monitor (RSRT)

If the OLAP Cache is not active, a local cache is used automatically. This local cache stores query results only within one session for the same user and will be used for operations like sorting. This cache was already available in SAP BW 2.x.

3.3 Configuration

The cache can be held in the application buffer/memory or/and in a BLOB table/cluster table/flat file which can be accessed centrally from all application servers.

The application buffer mode can be configured with and without swapping. The swapping mode stores displaced entries in a BLOB table/cluster table/flat file. If the cache is defined without swapping, displaced entries are just removed from the cache and this query (linked to the removed entries) must be read from the InfoProvider again. Usually you will only choose the 'without swapping' mode, if the I/O of your system is very slow; otherwise, the 'with swapping' mode is the recommended setting. The (global) OLAP Cache size in customizing only refers to the memory caching; a good setting to start with is between 100 and 200 MB – depending on the specific customer scenario. Note that the OLAP Cache is part of the export/import shared memory customized by the profile parameters rdsb/esm/buffersize_kb (size in KB) and rsdb/esm/max_objects (number of cache entries) respectively.

In system landscapes with several application servers, memory caching is in most cases only useful if some queries/user groups are restricted to one application server. In other cases, it might be advisable to centrally store the cache in a BLOB table/cluster table/flat file in order to enable queries on all application servers to utilize the cached entries.

A BLOB table can store larger records than a cluster table, so BLOB is usually faster for large OLAP Cache records; for smaller record lengths and small tables, cluster tables might result in less overhead.

See the following table for a collection of recommendations:

Cache Setting / Aspect	Cache inactive	Memory cache without swap	Memory cache with swap	Cluster / file cache for each application server	Cluster / file cache across application servers
Frequent changes of data	+	-	-	-	-
High amount of active users and query navigations	- - -	+ + +	+ + +	+ + +	+ + +
High amount of different queries	-	-	+ +	+ + +	+ + +
(Many) Ad-hoc queries	+ +	- - -	- -	-	-
Large result set in queries	-	-	+	+ +	+ + +
High load on InfoProvider database tables	- - -	+ + +	+ + +	+ + +	+ + +
Low query performance (without caching)	- - -	+ + +	+ + +	+ + +	+ + +
User groups assigned to application servers	- - -	+ + +	+ + +	+ + +	+
Slow I/O	- - -	+ + +	+ +	+	+

3.4 Re-use

Cached query results can be used only for the same query. In some cases, real subsets of cached query result sets can be used; this is possible, if the three following requirements are met:

- It is the same query ID
- All selection criteria of the second query must be contained in those of the first call, i.e. the query result set of the cached query is a superset of the second query
  Note that the selection variable must defined as '*changeable with query navigation*'! If the variable was not changeable, the variable value would be part of cache entry key and, hence, the cache entry could only be used for exactly the same variable values.
- The characteristics with differing selection criteria are included in the drill-down of the cached query result

The following example should make it clear:

The first query call has no restrictions for the time dimension. It shows up like this:

Cache Test

Cal. Year/Month	Plant	Key Figures

Plant		Cal. Year/Month	Orders
2000	Heathrow / Hayes	01.2002	68.000
		02.2002	40.000
		03.2002	118.000
		04.2002	28.000
		05.2002	108.000
		06.2002	58.000
		07.2002	58.000
		08.2002	40.000
		09.2002	96.000
		10.2002	22.000
		11.2002	70.000
		12.2002	22.000
		01.2003	78.000
		02.2003	51.000

The second query call is restricted to month 05.2002. It shows up like this:

Cache Test

Cal. Year/Month	Plant	Key Figures
05.2003		

Plant		Cal. Year/Month	Orders
2000	Heathrow / Hayes	05.2003	64.000
		Result	**64.000**
2010	DC London	05.2003	78.000
		Result	**78.000**
3010	Brussels	05.2003	69.000
		Result	**69.000**
4510	SAP Plant - Eastern Canada	05.2003	92.000
		Result	**92.000**
4520	SAP Plant - Central Canada	05.2003	30.000
		Result	**30.000**
5000	Tokyo	05.2003	70.000
		Result	**70.000**
6000	Mexico City	05.2003	87.000
		Result	**87.000**

For this second call, the OLAP Cache entries of the first call can be used, because it is:

- The same query ID
- The second query call is restricted to month 05.2002, which is contained in the first call
- Calendar Month is part of the drill-down of the first query

3.5 Invalidation

Cached results have a timestamp, which is compared with the timestamp of the last data load and meta data change.

If new data is loaded before a cached query is executed again, the OLAP engine states that the cached entry is out-of-date. It removes the old entry, loads the current query result from the InfoProviders and writes back this new OLAP Cache entry. The OLAP Cache entry is invalidated in the following cases:

- Transaction Data Load to the underlying InfoProvider(s)
- Deletion of Transaction Data from the underlying InfoProvider(s)
- Master Data Load (and hierarchies) and subsequent change run of InfoObjects contained in the query definition
- Currency conversion rate change

This invalidation is also done for meta data changes, i.e. changes to the query structure, and query generation.

Note that changes to key dates for time-dependent master data does not result in invalidation, but in an additional cache entry.

In addition, the OLAP Cache can be invalidated and deleted completely via the OLAP Cache monitor (transaction RSRT → Cache Monitor or transaction RSRCACHE). One query entry in the OLAP Cache can be invalidated by simply re-generating the query.