Performance Optimization of Long Running Queries Using OLAP Cache

Applies to:

SAP BW 7.0. For more information, visit the Business Intelligence homepage.

Summary

This article explains how to improve performance of long running queries using Information Broadcasting by pre-filling the OLAP cache and the automation of the same by using process chain and an ABAP program.

Authors: Raghavendra Padmanaban T.S, Ananda Theerthan J

Company: Wipro Technologies

Created on: 21 October 2009

Author Bio

Raghavendra Padmanaban is working as BI Consultant at Wipro Technologies.

Ananda Theerthan J is working as BI Consultant at Wipro Technologies.

Table of Contents

Introduction

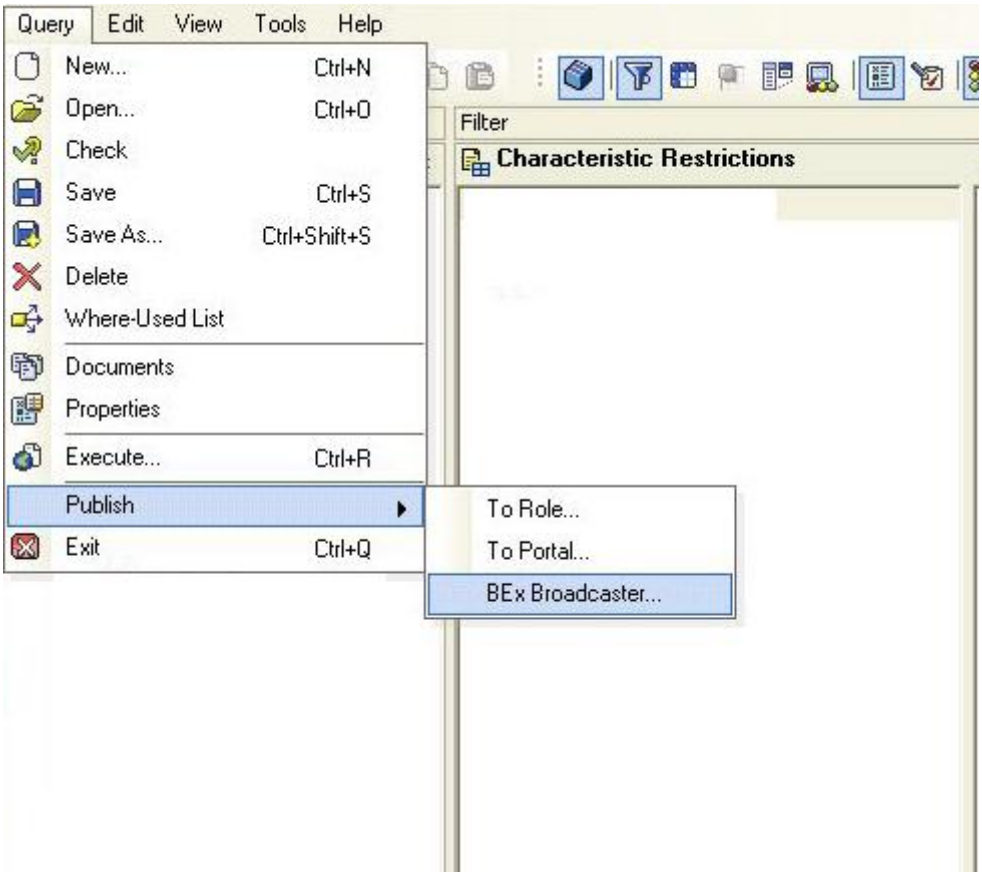
Increase query performance through Information Broadcasting of key queries by filling the OLAP cache where large number of users accessing a query or have query that access high volume of data. This article explains how to increase performance of long running BEx queries by filling the OLAP cache using information broadcasting which offers functions to optimize performance and for exception reporting. With the BEx Broadcaster, we can precalculate queries, query views, Web templates, reports and workbooks .The broadcasting function that we will be using for tuning query performance is "Filling the OLAP Cache".

The distribution type Fill OLAP Cache allows you to precalculate queries and to fill the OLAP cache with the generated data. If the users call Web Applications, queries, reports or workbooks that are based on this data, the access time is significantly reduced and the workload on the application server is considerably less. Whenever a query is run, it looks out for data in cache memory and if not found then hits database using select statements. This increases the time taken to execute a query. Hence the OLAP cache can be prefilled for the value for which the query will be run and next time when the query is actually run, it fetches the data from cache memory rather than database which will increase the performance. Here, we will start by creating a variant for query and use the created variant for filling the OLAP cache.

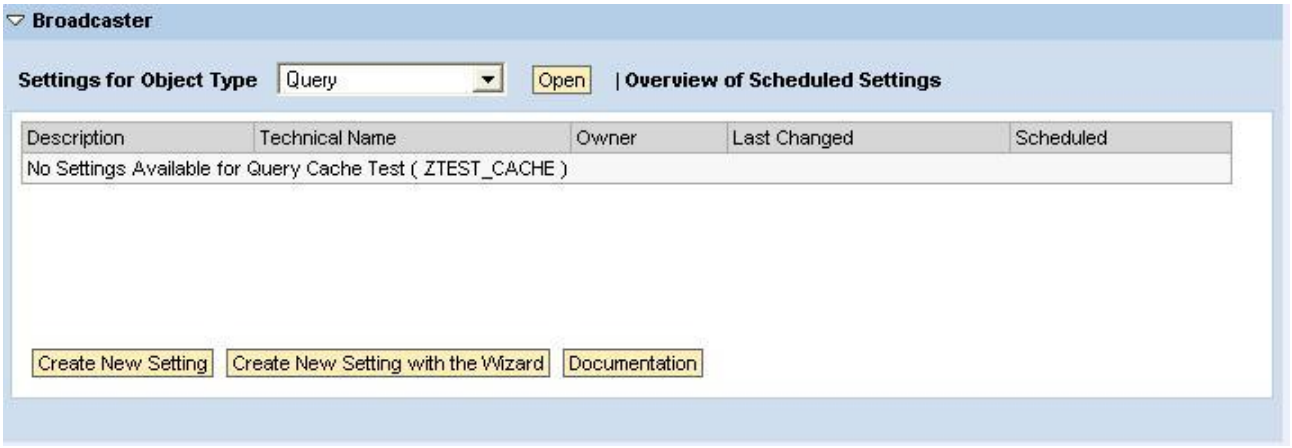
Creating Query Variant & Broadcaster Settings

Steps:

1. Open BEx Analyzer.
2. Run the query for which OLAP cache needs to be filled.
3. Create a global variant in query for the required selection.
4. Open BEx Query Designer and navigate to BEx Broadcaster.



After successful login, the below screen will be displayed.

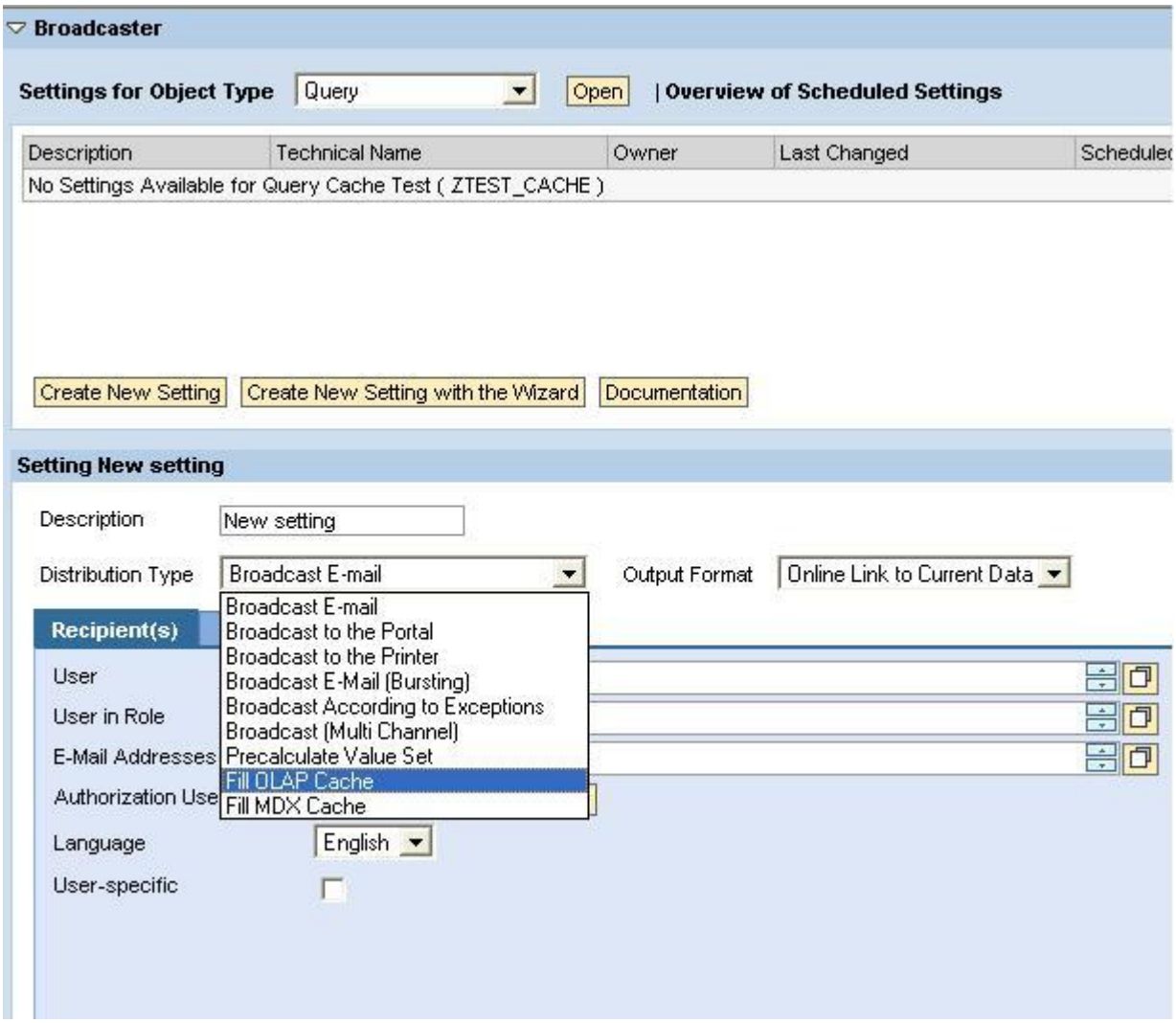


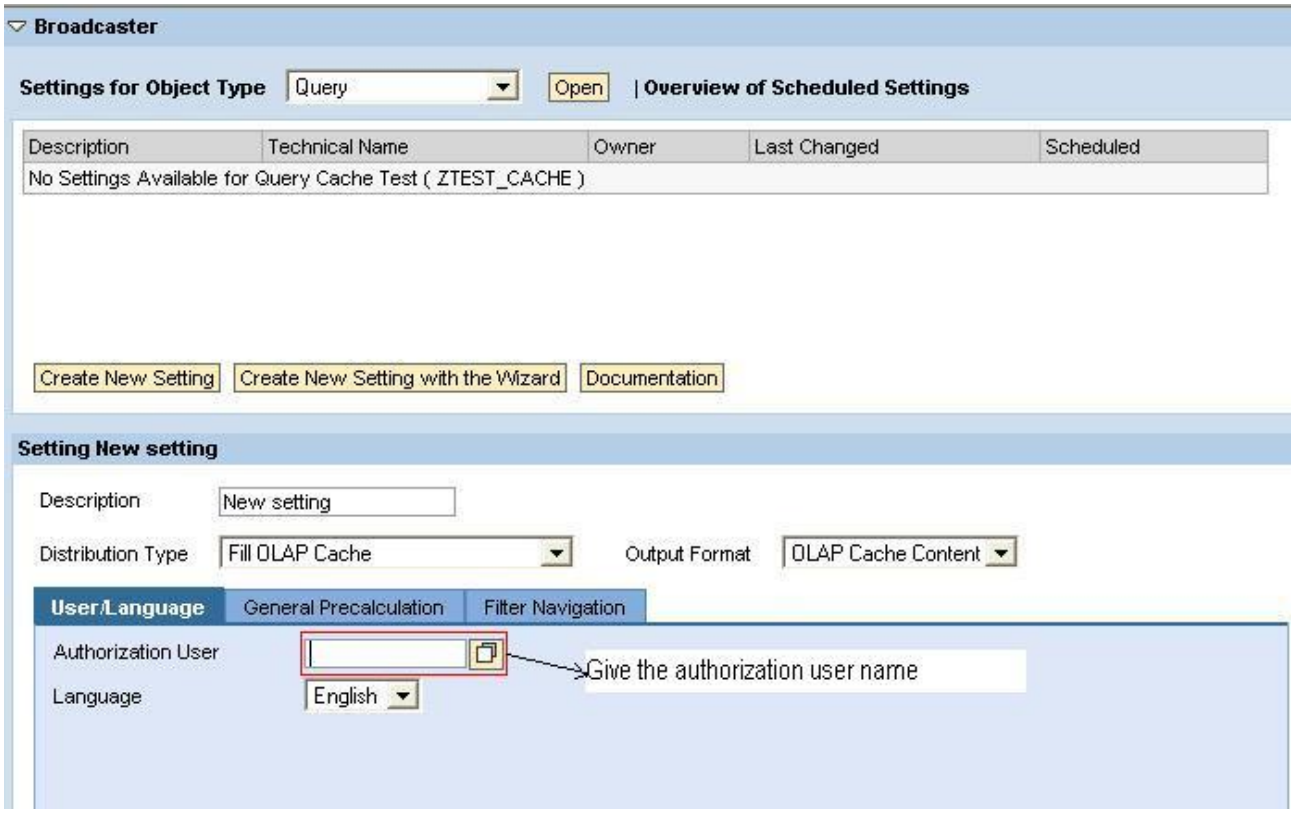
Click on "create new settings" to create new broadcaster setting.

In the distribution type menu select 'Fill OLAP Cache'

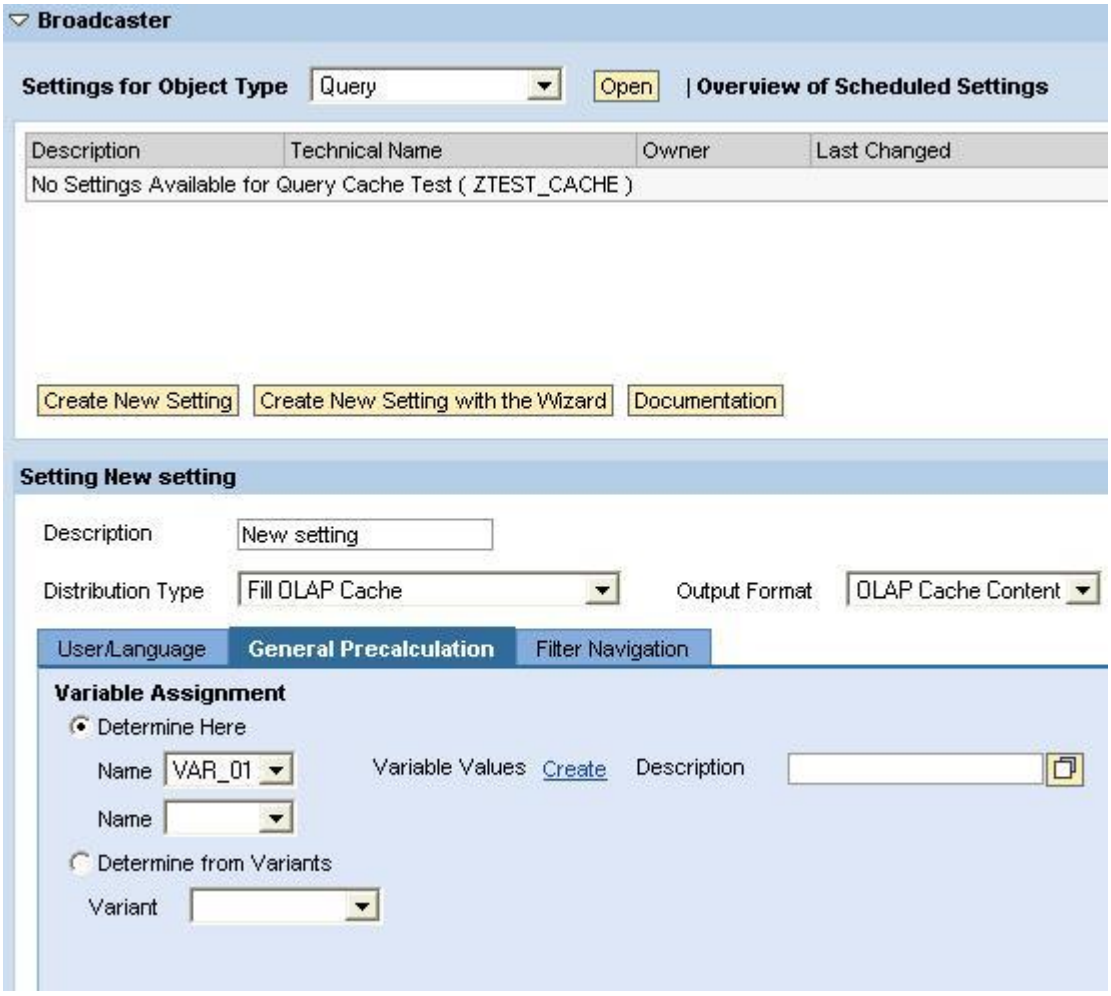
Next, click on "General Precalculation" tab.

Do the setting as shown below.



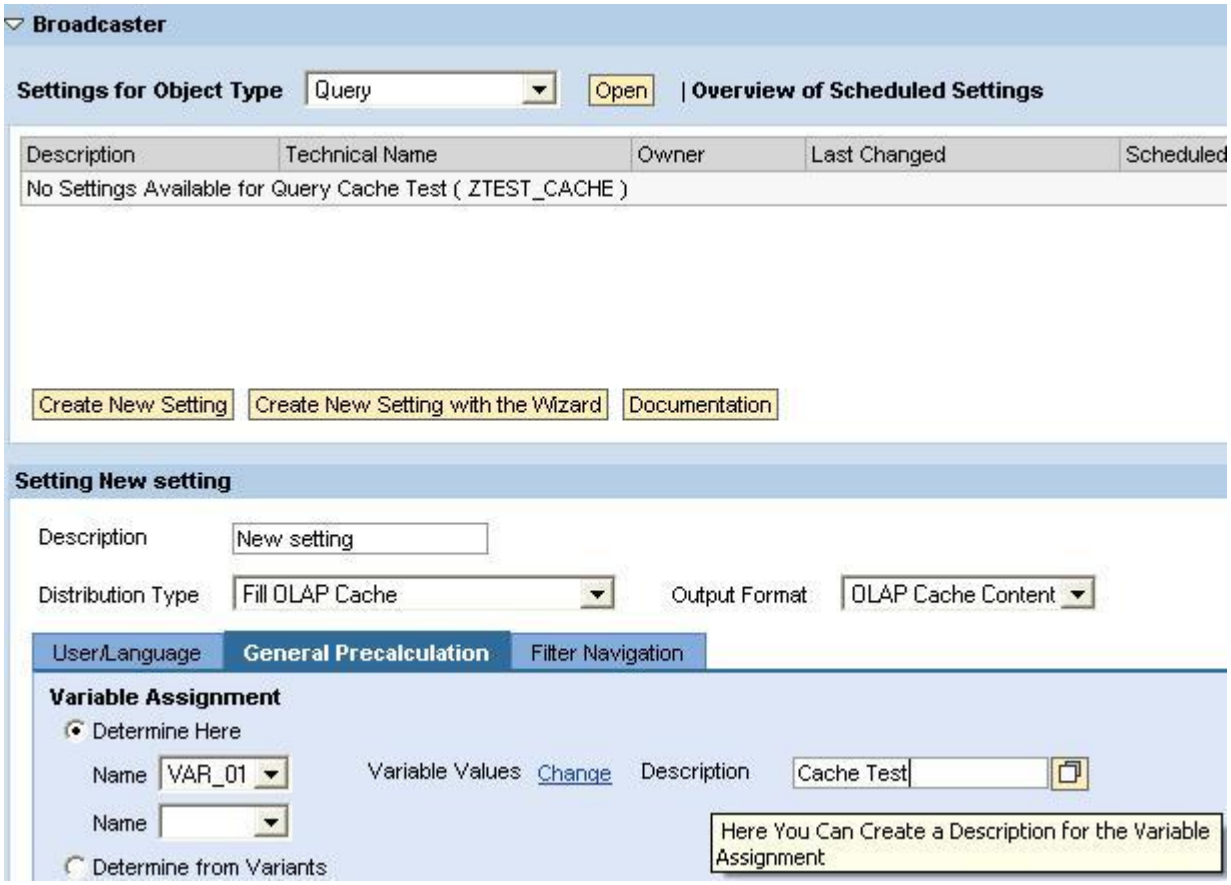


Give the username for background processing having required authorization. If the user doesn't have sufficient authorization, the cache process would fail.



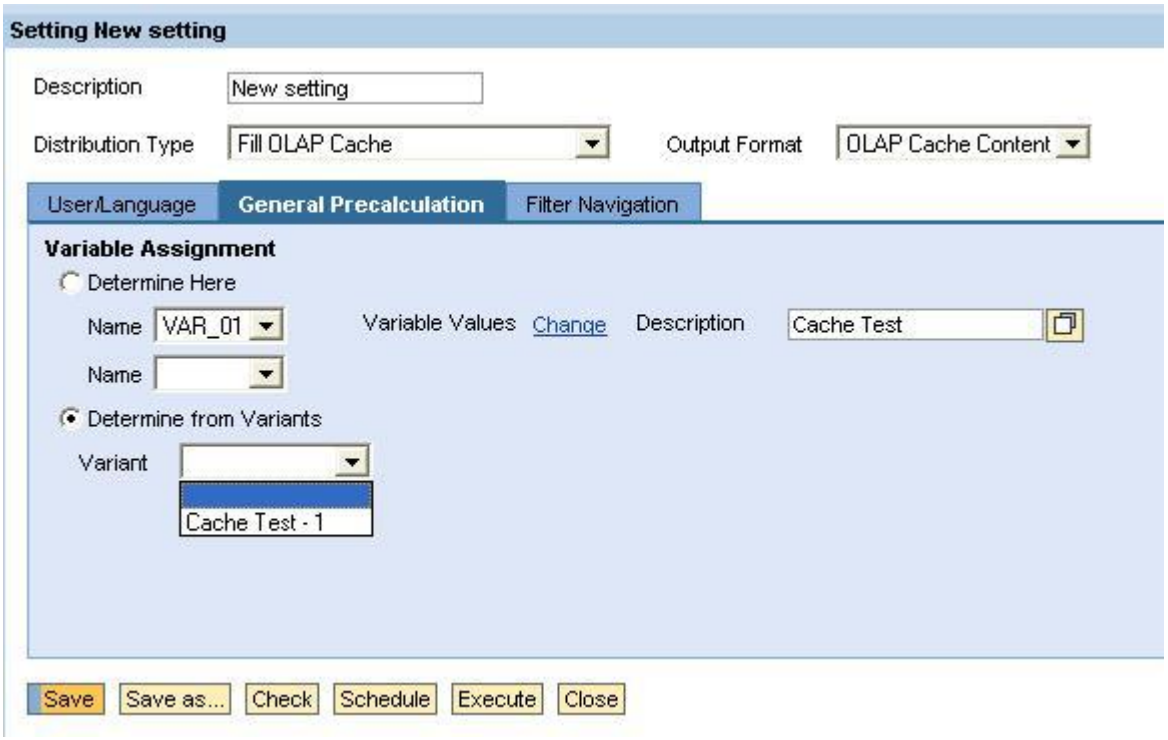
Select the first option to give a constant value to the variable VAR_01 for which cache will be filled every time the event is triggered. These VAR_01,VAR_02 are static and if this option is selected, cache will be filled for the same value every time the event is triggered. Click on create link to create variable values as shown below

We will be dealing with dynamic variant option as explained below.



Give suitable description for the variable.

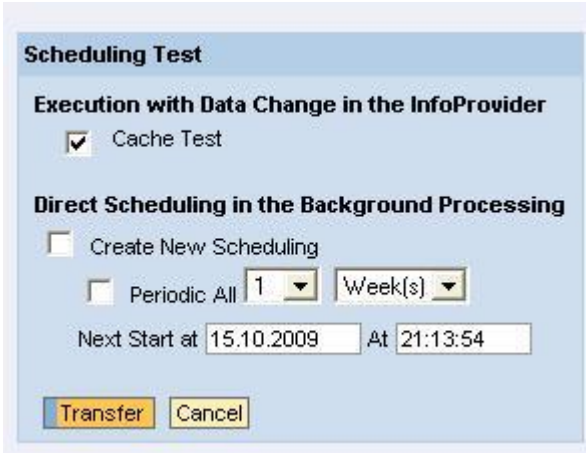
The next option 'Determine from Variants' is used to get values from variants created in query. So the pull down menu shows the list of variants created in query. Select the required variant in this screen as cache will be filled based on the variant value.



After selecting the required variants, save the setting created by clicking on the 'Save' button.

The filling of OLAP cache is scheduled using the "Schedule" option as shown below.

Once the setting is created for a query, the filling of OLAP cache can be scheduled either with an event trigger or on daily, weekly or monthly basis. The option "**Execution with Data change in the InfoProvider**" is used to fill OLAP cache whenever there is a change in data load status. The option "**Direct Scheduling**" can be used to schedule this job daily, monthly or weekly basis.



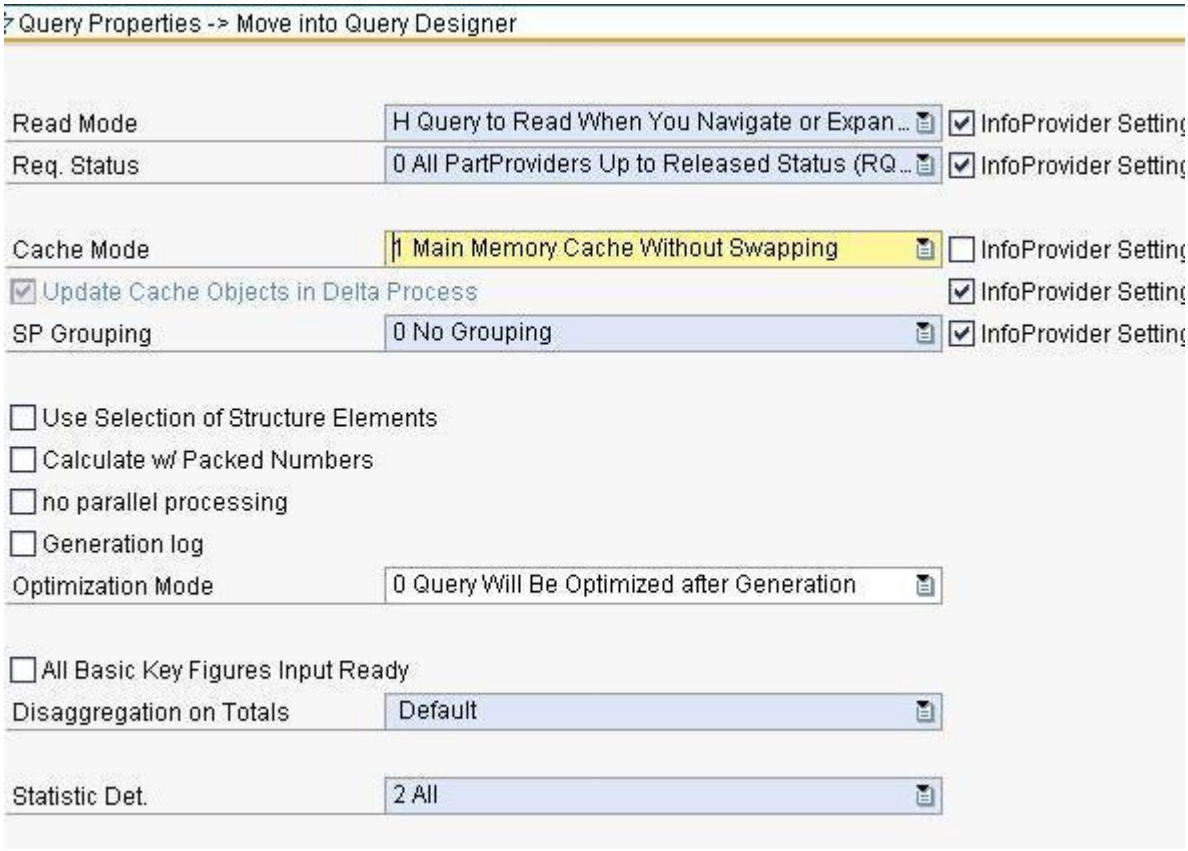
Enable Cache

The OLAP cache will be filled only if the cache property is enabled for a particular query or infoprovider.

To enable cache for a query, navigate to TCODE RSRT.

Input the query technical name and click on properties.

Change the **cache mode** setting as shown below.



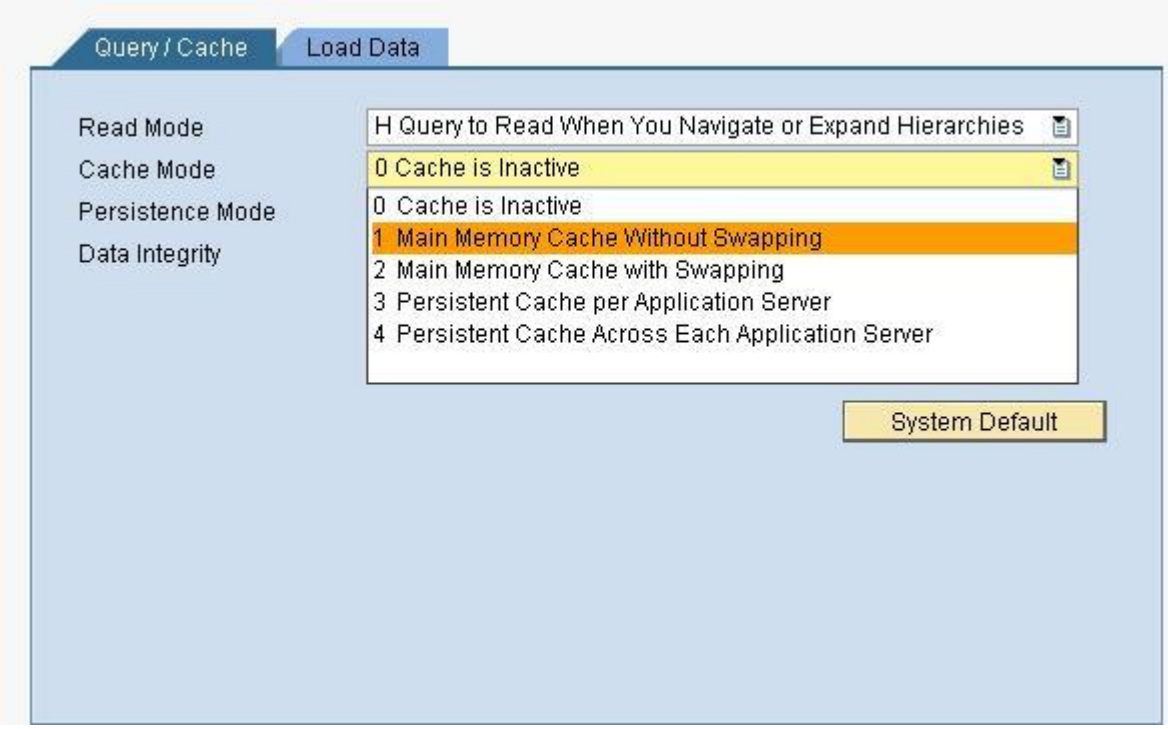
Note: Cache will be enabled only for the query for which this setting has been made. If there are multiple queries from an info provider, cache should be enabled for all queries individually.

To enable cache for an infoprovider,

Select the infoprovider

Navigate to Environment - > Infoprovider Properties - > Change.

Select the cache mode as shown below.



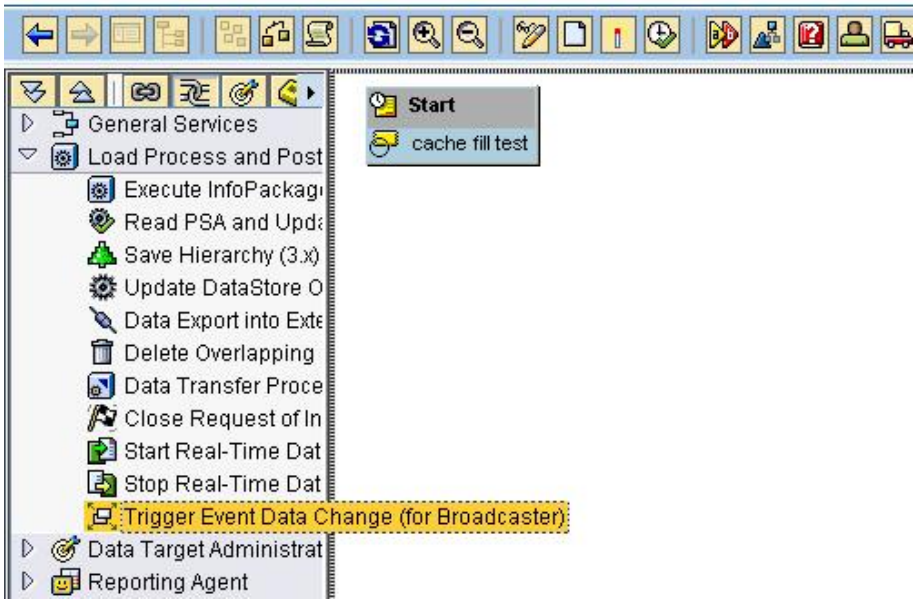
Note: If cache is enabled for an infoprovider, cache will be filled for all queries built on the infoprovider.

Automation of Event

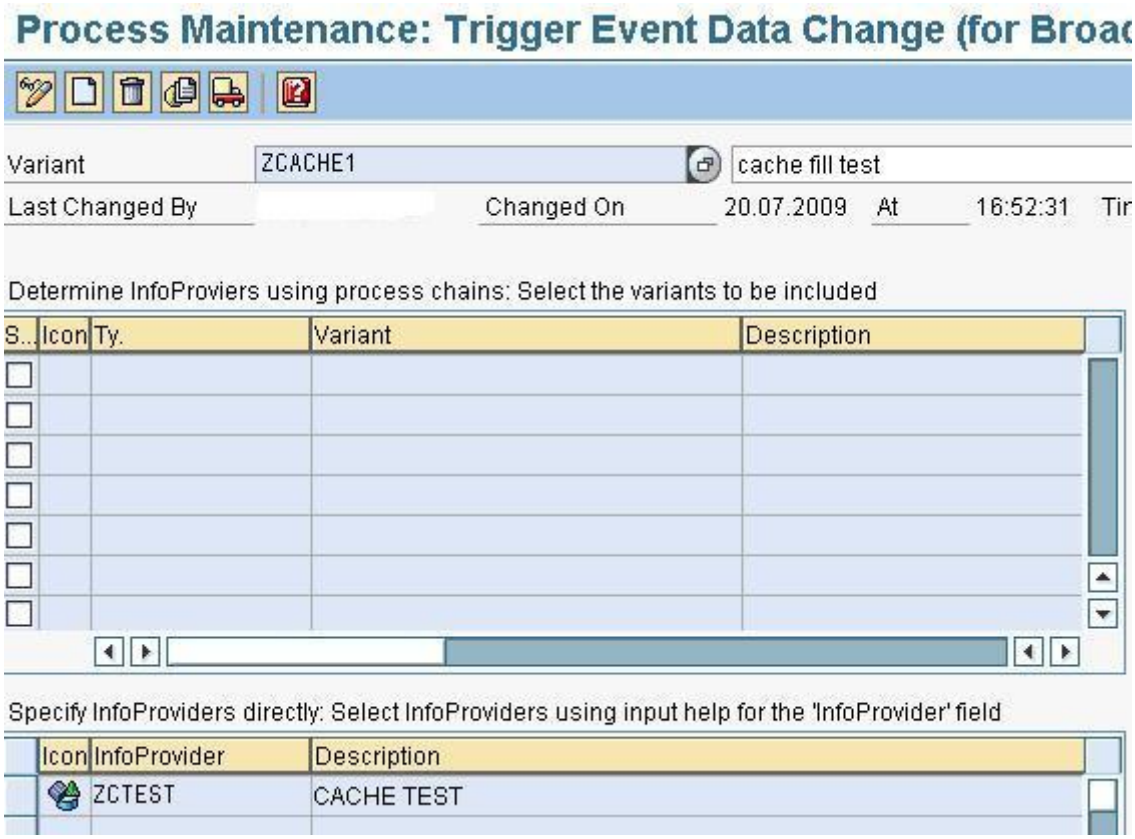
The OLAP cache is filled for the selected variant once the Event is triggered. So, we need to create a process chain to automate this event trigger.

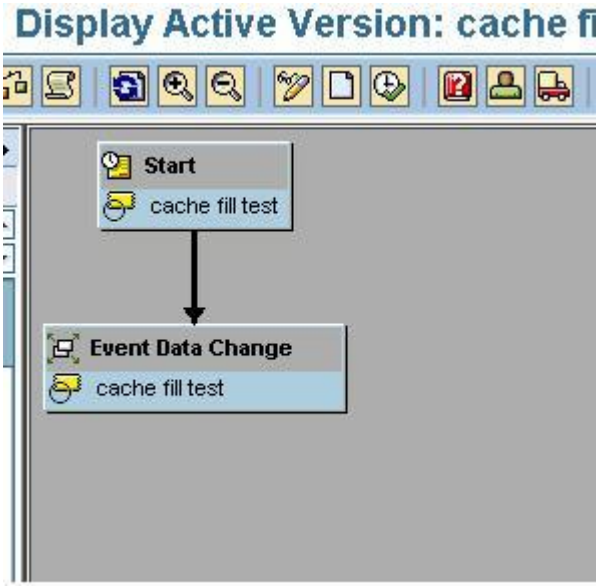
Goto RSPC

Create New Chain as shown below



After selecting the “Trigger Event Data Change” process, navigate to the process maintenance screen by double clicking on the process.

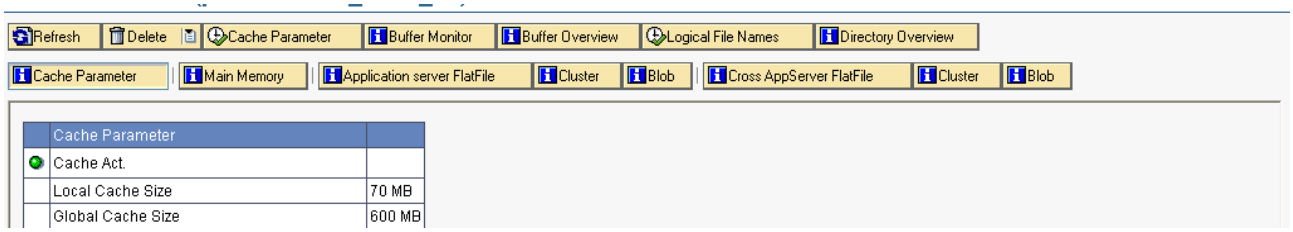




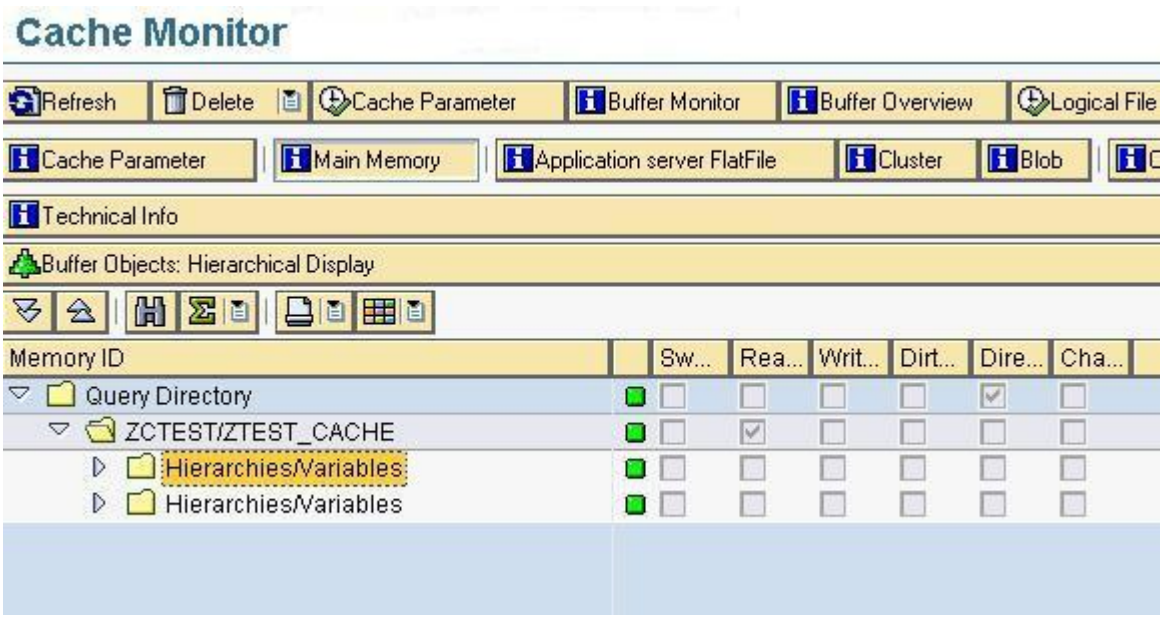
Select the required infoprovider and activate the chain after connecting the start variant and event trigger process. The chain can be scheduled as per the report frequency and the OLAP cache is filled accordingly. To check whether cache is filled for the selected variant, navigate Tcode RSRCACHE and check as shown below.

Cache Monitor

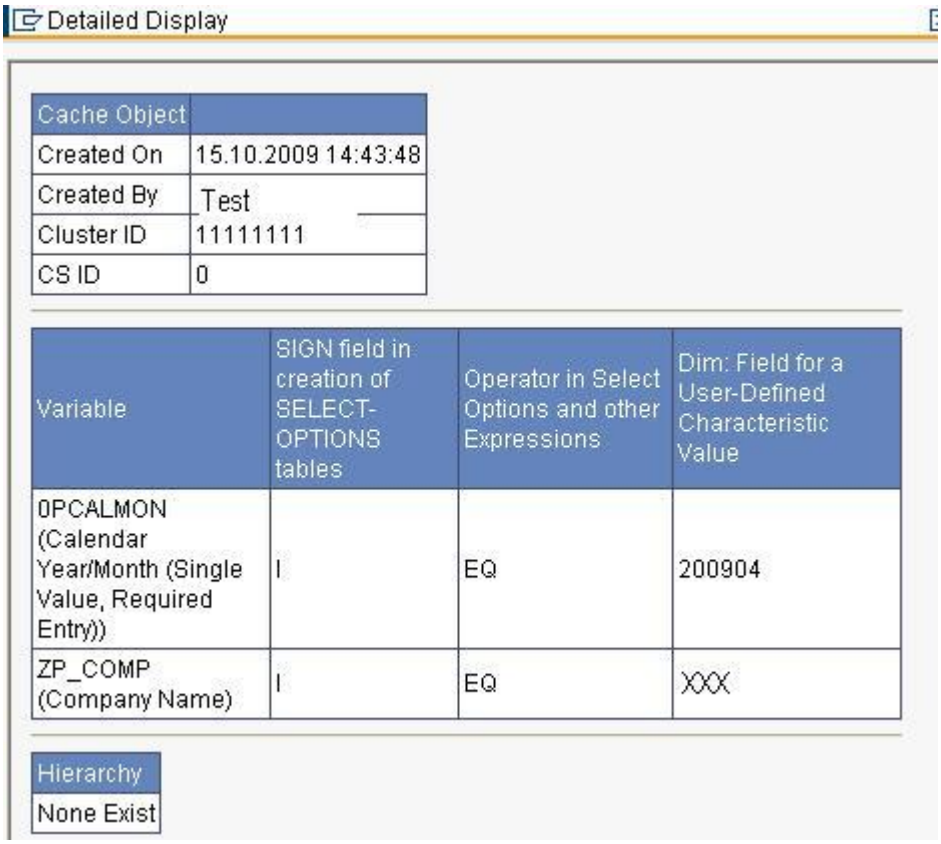
On the *Cache Monitor* screen you can get information on the global cache parameters, the amount of memory used by the query runtime objects and the current underlying cache structure.



Click on “Main Memory”



IF the cache is filled successfully, a folder below the query directory is created with the query technical name and the list of variants created for that query is shown. On double clicking the highlighted item, the selection for which the cache is filled can be seen.


Detailed Display

Cache Object	
Created On	15.10.2009 14:43:48
Created By	Test
Cluster ID	11111111
CS ID	0

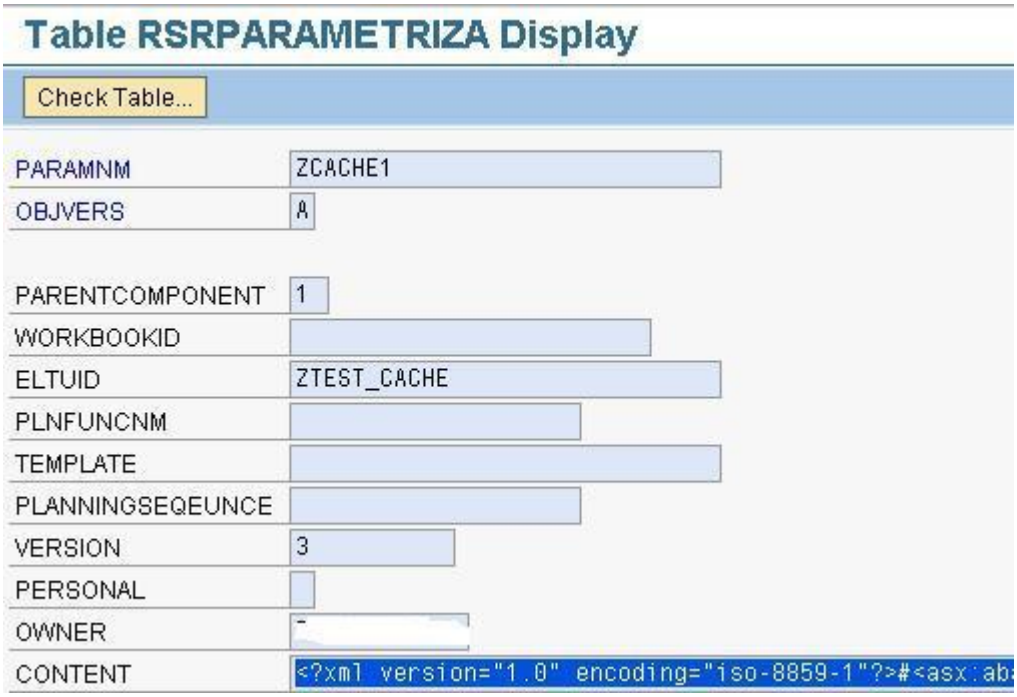
Variable	SIGN field in creation of SELECT-OPTIONS tables	Operator in Select Options and other Expressions	Dim: Field for a User-Defined Characteristic Value
0PCALMON (Calendar Year/Month (Single Value, Required Entry))	I	EQ	200904
ZP_COMP (Company Name)	I	EQ	XXX

Hierarchy
None Exist

Automation of Query Variant

The variant which is created in query is static and doesn't change dynamically. To make the query variant get dynamic values, we need to automate the process using an ABAP program. The variants created in query, workbook, web application etc are stored in database table **`rsrparametriza.`**

The field **PARAMNM** stores the variant technical name and **ELTUID** stores the query technical name for which the variant is created. The field **CONTENT** stores the values given for the variant in XML format as shown below.



Select the required entry and click on display which will take us to the above screen. Copy the entire contents from the field '**CONTENT**' and paste it in a notepad file which will look as below.

<?xml version="1.0" encoding="iso-8859-1"?>#<asx:abap xmlns:asx="http://www.sap.com/abapxml" version="1.0"><asx:values><PAGE><item><VNAM>0PCALMON</VNAM><DATA_PROV>DATA_PROVIDER_1</DATA_PROV><RANGE><RRRANGE><SIGN>I</SIGN><OPT>EQ</OPT><LOW>**200904**</LOW><HIGH/></RRRANGE></RANGE></item><item><VNAM>ZP_COMP</VNAM><DATA_PROV>DATA_PROVIDER_1</DATA_PROV><RANGE><RRRANGE><SIGN>I</SIGN><OPT>EQ</OPT><LOW>XXX</LOW><HIGH/></RRRANGE></RANGE></item></PAGE></asx:values></asx:abap>

The values between the tags <VNAM> & </VNAM> are the variable technical names used in query. The tag <RANGE> stores either the single or range values given for a particular variant within <LOW> and <HIGH> tags. We need to identify the LOW and HIGH values for a date field which needs to be changed frequently so that cache can be filled for the required selection. Now, the values between the tags <LOW> and <HIGH> needs to be changed and updated back to the table. A sample program is given on how to update the table

values having standard naming convention for the variants. This program finds and updates the date values for which a query variant is created.

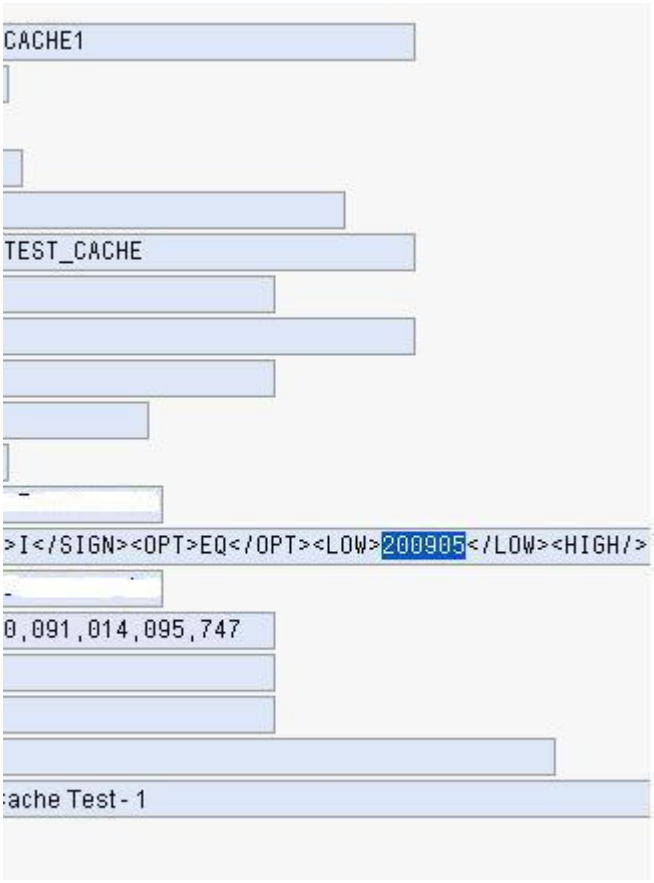
Note If the format of the XML code in **CONTENT** field is not maintained correctly, the query doesn't run and the entry from rsrparametriza needs to be deleted manually for the query to open.

Source Code of the Tool Used to Automate the Query Variant

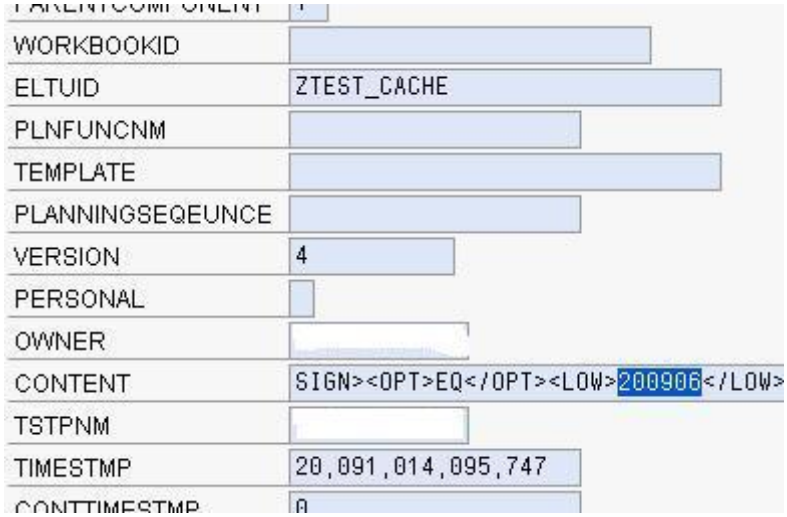
```
CONSTANTS : lc_pat TYPE c LENGTH 2 VALUE '20',
            lc_parameter TYPE c LENGTH 7 VALUE 'ZCACHE1' "Give variant name.


DATA : l_count TYPE i,
       l_off  TYPE i,
       l_moff TYPE i,
       l_mlen TYPE i,
       l_startdate_old TYPE sy-datum,
       l_enddate_old TYPE sy-datum,
       l_startdate TYPE sy-datum,
       l_enddate TYPE sy-datum,
       l_month_old type  /bi0/oicalmonth,
       l_month_new TYPE /bi0/oicalmonth.

DATA: l_year TYPE i,
      l_month TYPE i.

DATA : lt_variant TYPE TABLE OF rsrparametriza,
       lw_variant LIKE LINE OF lt_variant.


SELECT * FROM rsrparametriza INTO TABLE lt_variant
  WHERE paramnm LIKE lc_parameter AND objvers = 'A'.


LOOP AT lt_variant INTO lw_variant.

  sy-subrc = 0.
  CLEAR:l_off,l_moff,l_month,l_year,l_count.

  WHILE sy-subrc = 0.
    FIND lc_pat IN SECTION OFFSET l_off OF
         lw_variant-content
         MATCH OFFSET l_moff
         MATCH LENGTH l_mlen.

    IF sy-subrc = 0.
**- counter to track no.of hits
      l_count = l_count + 1.

**- fetch  value(low) , considering there are no ranges given in variant

      IF l_count = 1.
        l_month_old = lw_variant-content+l_moff(6).

        l_year = l_month_old+0(4).
        l_month = l_month_old+4(2).

        IF l_month = 12.
          l_year = l_year + 1.
          l_month_new+0(4) = l_year.
          l_month_new+4(2) = 01.
        ELSE.
```

```
            l_month_new+0(4) = l_year.
            l_month_new+4(2) = l_month + 1.
          ENDIF.
        ENDIF.

    **- set the offset for next search.
        l_off = l_moff + 6.

    **- if a range variable is used, l_count will be 2 for HIGH value and the above block
    should be modified.


      ENDIF.
    ENDWHILE.


    REPLACE ALL OCCURRENCES OF l_month_old IN lw_variant-content WITH l_month_new.

    UPDATE rsrparametriza FROM lw_variant.

  ENDLOOP.
```

Before running the program, the value of the variant is as shown



After execution of program,



This program can be scheduled based on the report frequency and once this is done, the event should be triggered using process chain so that cache is filled for the updated variant. So when the report is run, the data is fetched from cache memory which is prefilled and this increases the performance of long running queries.

Related Content

Information Broadcasting

Filling the OLAP Cache

For more information, visit the Business Intelligence homepage.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.