Smarten
Advanced Data Discovery
Powered by ElegantJ BI

Impact of Cube Design on Performance

Business Intelligence & Advanced Data Discovery

Document Version: 6.0
Product Version: 5.0 and above

Document Information	
Document ID	Smarten-Impact-of-Cube-Design-on- Performance
Document Version	6.0
Product Version	5.0 and above
Date	10-December-2018
Recipient	NA
Author	EMTPL

Statement of Confidentiality, Disclaimer and Copyright

This document contains information that is proprietary and confidential to EMTPL, which shall not be disclosed, transmitted, or duplicated, used in whole or in part for any purpose other than its intended purpose. Any use or disclosure in whole or in part of this information without the express written permission of EMTPL is prohibited.

Any other company and product names mentioned are used for identification purpose only, may be trademarks of their respective owners and are duly acknowledged.

Disclaimer

This document is intended to support administrators, technology managers or developers using and implementing Smarten. The business needs of each organization will vary and this document is expected to provide guidelines and not rules for making any decisions related to Smarten. The overall performance of Smarten depends on many factors, including but not limited to hardware configuration and network throughput.

Contents

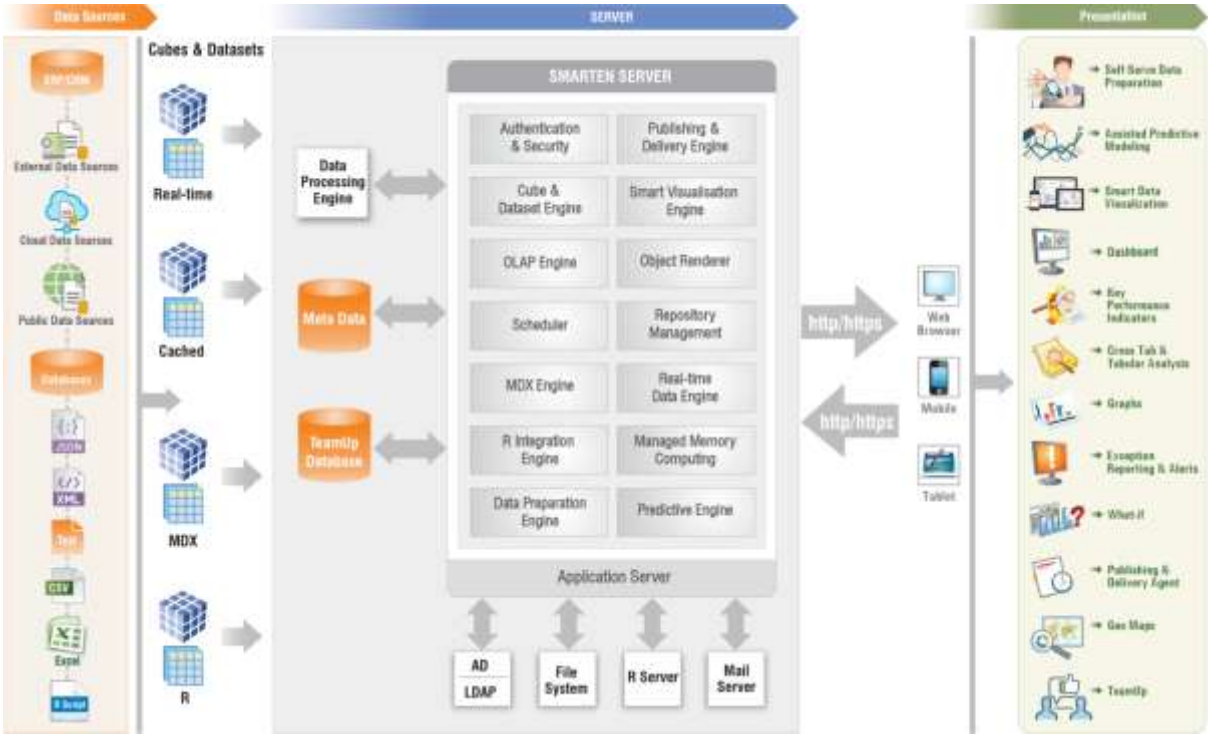
1 Introduction

The purpose of this document is to elaborate on the impact of cube design decisions on the performance of Smarten.

2 Smarten Concepts

Before proceeding to discuss the recommended design practices, a few Smarten technical concepts should be understood. They relate to how cubes are created and stored, and to how the server processes the data.

2.1 Smarten architecture diagram



HIGHER LEVEL ARCHITECTURE—SMARTEN

This document focuses on the design of the data cubes and its impact on performance.

Smarten supports both real-time and cached cube architecture. There is also an option for aggregation in cache cubes, and user can choose if user wants to perform aggregation for cached cubes at cube level or not.

Cached cubes will store indexed, pre-aggregated data in the cache files. For MDX and Real-time cubes, only metadata information is stored in Smarten.

Building these data cubes is a key process. The design decisions made in building these cubes have significant impact on the performance of Smarten.

2.2 Unique Aggregated Result Set (UARS)

The number of unique aggregated result sets (UARS) is the number of unique records in a data cube built by the Smarten cube engine.

The number of unique aggregated records is related to the dimensions and measures used in building the cube. A combination of dimensions makes up a unique record. Any change in the number of dimensions or measures changes the number of unique rows and the physical size of the data cube.

Understanding UARS

UARS is explained with an example.

In the sample table below, the Country dimension has 2 values and the City dimension has 5 values. These data are for a measure called Sales Value. This results in 5 unique records (UARS). Each record is a unique combination of Country and City for the Sales Value.

Resultset Raw Data		
Country	**City**	**Sales**
India	Mumbai	$200.00
India	Delhi	$400.00
USA	New York	$5,000.00
India	Chennai	$2,000.00
India	Chennai	$500.00
USA	Atlanta	$7,500.00
India	Mumbai	$200.00
India	Mumbai	$2,500.00
India	Delhi	$400.00

UARS		
India	Mumbai	$2,900.00
India	Chennai	$2,500.00
India	Delhi	$800.00
USA	Atlanta	$7,500.00
USA	New York	$5,000.00

2.3 Impact of adding dimensions on UARS

Consider the table below, with multiple dimensions, where the last column indicates the maximum number of unique aggregated records that these dimensions will create.

The sample table takes the simplistic view that each country has the same number of regions, each region has the same number of cities, and each city has the same number of dealers etc. This is a hypothetical example used to demonstrate the impact of adding dimensions with large number of values.

	Dimensions					
	Country	**Region (Zone)**	**Town/City**	**Dealer**	**Retailer**	**UARS**
Recordset 1	0	4				4
Recordset 2	1	4				4
Recordset 3	2	4				8
Recordset 4	2	4	25			200
Recordset 5	2	4	30			240
Recordset 6	2	4	40	80		25,600
Recordset 7	5	4	25	80		40,000

Recordset 8	5	15	25	80		150,000
Recordset 9	1	4	25	80	Average 1,200	9,600,000
Recordset 10	1	4	25	80	Average 1,200	9,600,000

In the example above, Country, Region and City are dimensions with fewer values, and will not increase the UARS substantially.

In the Recordsets 1-5, the maximum number of UARS does not increase dramatically, as the increase in the values of Dimensions (the numbers in Countries, Region and City) is not large. For example, if the number of cities increases from 25 in Recordset 4 to 30 in Recordset 5, the number of UARS will not increase substantially.

However, when a dimension like Dealer with a value of 80 per city is added, the increase in UARS is higher. The number of dealers is assumed to be 80 for each city, so for each combination country-region-city-dealer, there will be a unique row. In the above sample, with the addition of dealers in Recordset 6, the unique records increased from 240 to 25,600. This is a substantial increase.

Adding a dimension like Retailer with larger number of values has a significant impact on the UARS, as can be noticed from the aforementioned table. Average 1200 retailers for each dealer in each town in a region for a particular country will generate many unique rows. For example, in Recordset 9, the increase in unique records in comparison with Recordset 8 is very high with the addition of the Retailer Dimension. The unique records jump from 150,000 to almost a million records. When there is design decision to increase the number of rows by such a large number, the value vs performance, as discussed in the ensuing sections, will be a key factor.

The impact of adding a large number of dimensions with fewer values is usually lesser than adding a dimension with a very large number of dimension values.

A practitioner of good design will ask whether we really need to analyze data at retailer level for a regular business decision, as this will lead to some trade-off in terms of performance. An alternate design decision could be to separate the cubes.

Increasing the size of the UARS is not a good design practice. One has to consider business needs for the design along with the alternative design strategy of multiple cubes while making an overall design plan.

In general, a dimension increases the number of rows and hence, the size of the cube. This is different from adding a column to the existing UARS which are not dimensions.

Some columns do not add value to the UARS but just increase the size. For example, having both a product code and a product name in the Resultset will add to the size but not to the uniqueness of the UARS. Selection of columns which do not add value for analysis is avoidable.

2.4 Custom Cube Measure

A value computed based on data available in the existing cube measures is called **Custom Cube Measure.**

For example – a cube has sales quantity and unit price measures. To compute the sales value, we have to multiply the product quantity and the unit price to create another measure called sales value.

A measure can be created in the front end (for example, in the cross tab analysis) or in the cube itself. If this measure is used frequently, it is prudent to add it to the cube as a custom measure rather than adding it to the front end where it gets calculated each time the user requests for the data.

A custom measure can be added by one of the three options below:

- This computed field can be created while creating the data in the ETL. This will provide a ready computed field for Smarten cubes
- A calculated field in the cube – custom cube measure, which is the product of unit value and quantity
- Make a UDDC (in the user front end) which multiplies the unit value of the product with the quantity

The first option is the best practice, the second option is a good alternative, and the third option is not a desirable method.

2.5 Custom Cube Dimension

A Custom Cube Dimension column is a new cube dimension column created on the basis of existing cube dimension columns. The cube dimension columns not found in the data source (database, CSV or any other data source used for creating the cube), can be created on the fly by the administrator.

Administrators can create a new Custom Cube Dimension by performing various string, arithmetic, date, statistics, trigonometry or conditional functions using arithmetic operators (like +, -, /, etc) or comparison operators (like =, >, < etc.) on either two or more existing cube dimension columns.

For example, if we want to make a dimension called Employee Name by concatenation of last name and first name together from existing cube dimension columns, it is called a Custom Dimension. The impact of adding a few custom dimensions is not significant as it is based on existing dimensions and does not dramatically increase the UARS.

There are two paths to add a custom dimension:

- Complete this operation at the ETL level and provide a ready computed field for Smarten cubes. This will eliminate the need for a custom cube dimension
- A computed field at the cube level, Custom Cube measure

2.6 Time Dimension

Data Cubes usually need a **Time Dimension** for time-period-related operations that look at periods, i.e. weeks, months, quarters or years.

Using the Time Dimension, the hierarchical levels for the drill down path are Year > Quarter > Month > Week > Day > Hour > Minute > Seconds.

For example,

If the time dimension is applied on the following record set, 8 additional columns are added to the cube resultset. This will increase the size of the cube as well as the time required for building the cube.

Country	City	Sales	Date / Time
India	Mumbai	$200.00	01-01-2013 01:05:15
India	Delhi	$400.00	02-01-2013 01:10:22
USA	New York	$5,000.00	03-02-2013 02:00:33
India	Chennai	$2,000.00	01-04-2014 01:15:45
India	Mumbai	$1,000.00	01-01-2013 01:05:15
USA	New York	$5,000.00	03-02-2013 02:00:34
USA	New York	$5,000.00	03-02-2013 02:00:35

UARS

Country	City	Sales	Date/Time	Date/ year	Date/ Quarter	Date/ Month	Date/ Week	Date/ Day	Date/ hour	Date/ minute	Date/ second
India	Mumbai	$1,200.00	01-01-2013 01:05:15	2013	Q1	1	1	1	1	5	15
India	Delhi	$400.00	02-01-2013 01:10:22	2013	Q1	1	27	2	1	10	22
USA	New York	$5,000.00	03-02-2013 02:00:33	2013	Q1	2	27	3	2	0	33
India	Chennai	$2,000.00	01-04-2014 01:15:45	2014	Q2	4	27	1	1	15	45
USA	New York	$5,000.00	03-02-2013 02:00:34	2013	Q1	2	27	3	2	0	34
USA	New York	$5,000.00	03-02-2013 02:00:35	2013	Q1	2	27	3	2	0	35

Values get aggregated based on the time dimension. If the date and time are the same for a record, along with the other dimensions, a unique record will be created. For example, in the above example, the first record for India-Mumbai is unique in terms of Country, City and Time, and hence the measure is added up, while New York has 3 records with just a second of difference and keeps the independent rows.

Had it been no date time stamp on the above data, the recordset with the dimension would have been as follows:

Country	City	Sales	Date / Time	Date/ year	Date / Quarter	Date / Month	Date / Week	Date / Day
India	Mumbai	$1,200.00	01-01-2013	2013	Q1	1	1	1
India	Delhi	$400.00	02-01-2013	2013	Q1	1	27	2
USA	New York	$15,000.00	03-02-2013	2013	Q1	2	27	3
India	Chennai	$2,000.00	01-04-2014	2014	Q2	4	27	1

If only date values are considered instead of date-time, in the above case, New York, which had 3 distinct rows when the Time Dimension was considered is now down to a single row as all the three transactions took place on the same day.

A date without time dimension increases the UARS by 5 columns and a date-time dimension will increase the UARS by 8 columns. This is valid for each date-time field. If we were to consider 3 date fields used, this would result in 9 more dimensions. This can have medium to high effect on UARS, depending on data.

Depending on the level of drill-down operation required by your business, one can choose to change the date-time field to a date field during the ETL operation. If you do not need to drill down below day level, do not choose to have a date-time field as it adds to the UARS in terms of rows and columns.

If the date-time fields are not to be used for business analysis, these need not be a part of the dataset.

Thus, if you do not need any date or date-time field for business analysis, it is prudent to not use it, as these date or date-time fields have more impact on performance than normal dimension columns.

If the business requirements are not for drilldown up to seconds level, it is obvious and prudent not to have the time component in the data.

3 Design Decision and its Impact on Performance

In this module, the design decisions will be explored and their impact on corresponding performance metric will be explained. These involve:

- (D1) Adding a dimension having a small number of values (0-50 unique values) to a cube
- (D2) Adding a dimension having a medium number of values (51-500 unique values) to a cube
- (D3) Adding a dimension having a large number of values (above 500 unique values) to a cube
- (D4) Adding a measure to a cube
- (D5) Adding a dimension hierarchy level to a cube
- (D6) Adding a Custom Cube Measure
- (D7) Adding Custom Cube Dimension
- (D8) Adding Time Dimension (automated time series) to the cube
- (D9) Creating a link cube - union type
- (D10) Creating a link cube - join type
- (D11) Creating a cube with drill through data
- (D12) Adding pre-defined data operations for measures

The following performance metrics will be explored against each of the above design decisions.

- (M1) Impact on front-end user access speed (response time)
- (M2) Impact on server user process RAM requirement
- (M3) Impact on server user process CPU usage
- (M4) Impact on front-end user network bandwidth usage
- (M5) Impact on cube build/rebuild time performance
- (M6) Impact on cube build/rebuild process RAM requirement
- (M7) Impact on cube build/rebuild process CPU usage
- (M8) Impact on storage (disk space) requirement on server

	M1	M2	M3	M4	M5	M6	M7	M8
D1	Low	Low	Low	Low	Low	Low	Low	Low
D2	Medium	Medium	Medium	Low	Medium	Medium	Medium	Medium
D3	High	High	High	Low	High	High	High	High
D4	Low	Low	Low	Low	Low	Low	Low	Low
D5	Low	Low	Low	Low	Low	Low	Low	Low
D6	Low	Low	Low	Low	Low	Medium	Medium	Low
D7	Low	Low	Low	Low	Low	Medium	Medium	Low
D8	Low	Low	Low	Low	Medium	Medium	Medium	Medium
D9	High	High	High	Low	High	High	High	High
D10	High	High	High	Low	High	High	High	High
D11	Low	Low	Low	Low	High	Medium	Medium	High
D12	High	Low	High	Low	High	Low	High	Medium

Note:
The network bandwidth can be impacted by the size of the front-end BI object.

4 Cube type selection recommendations

Here are higher level recommendations for cube type selection.

Smarten Cube selection			
Factors to consider	Type of Smarten cube		
	Cached Cube		**Real Time Cube**
	With Aggregation	Without Aggregation	
If non real time Aggregated data is required in Front End Analytics and Source data is aggregated		✔	
If non real-time Aggregated data is required in Front End Analytics and Source data is not aggregated	✔		
If real-time data is required in Front End Analytics and Source data is aggregated/not aggregated			✔
If non real-time transactional [non-aggregated] data is required in Front End Analytics and Source data is transactional		✔	
cube should store data	✔	✔	
cube should act only as meta data and should not store any data			✔
Cube rebuilding / refreshing / processing time	Higher	Lower	NA
Zero latency for data (from source data to front-end)			✔
Latency for data (from source data to front-end) – Depends on cube refresh scheduler and time required to refresh the cube	✔	✔	

5 Smarten – Design Practice for Effective Performance

5.1 Design considerations on using ETL for achieving performance

One of the key purposes for recommending the use of a good ETL tool is to prepare data in an optimal way before it reaches Smarten. A complete document on ETL concepts and their importance in implementing a BI solution is available on request. The document provides an overview of ETL concepts and details good practices to be followed when using an ETL tool along with the Smarten.

With a proper use of ETL:

- The data quality can be improved with data cleanup at initial levels, in order to remove unwanted fields, errors in the data, and eliminate duplication.
- The data from multiple sources can be consolidated.
- It is prudent to perform all data operations, calculations, consolidations and string operations at the ETL level. This will also reduce the load on the cube engine and eliminate the need for custom cube dimensions and measures. This will enable Smarten cube engine to focus on its job of building a UARS rather than doing transformations and related calculations.

5.2 Design considerations for defining the cubes

Based on the organization structure, the security level and the periodicity of the analysis are important considerations in designing cubes. Usually, having multiple cubes (created reasonably, by avoiding to have too many redundant cubes) based on user needs and business functional needs is a good practice rather than building one big massive cube serving all needs. For example, in HR management, having a separate cube for each of the performance management and compensation analysis functionalities is prudent. Similarly, cubes for operations and purchase are best kept separated, even if they have some dimensions in common.

Another example would be corporations which operate with many regional offices. Each regional office is usually limited to working with its own operational data at the regional level.

Several design path choices in such a situation:

- Have a large cube for all regions and provide security rules which restrict the access of data to each region. In this case, the size of the UARS increases dramatically and all users who only have to access a fraction of these data reach for the same UARS.
- The other choice is to have cubes for each region: Have regional cubes where each region can do regional analysis, and have a Corporate / HQ cube where only aggregated data from regions is compiled, and that serves the purpose of strategic and tactical level corporate analysis.. Regional cubes access can also be given to corporate HQ users, in case they need to tap into operational data for any region.
- Cubes can be based on the period for which the data is being analyzed. Short term decisions which need quick and regular analysis could be based on a cube with 3 years of data. Larger cubes covering a longer period but having aggregation at a higher level could be for more tactical decision-making. For example, one does not need details of sales per retailer when analyzing 7 years but needs it for analyzing 3 years of data. This is a choice based on business needs. Having less detailed data for longer periods reduces the size of UARS and is a good design practice.

5.3 Design considerations for link cubes (JOIN and UNION)

In Smarten, one can link data from two cubes to create Union or Join cubes. It is prudent not to deal with cubes in the same way as one deal with an RDBMS. Avoid Join and Union cubes unless it is absolutely necessary.

These Joins increase the size of UARS and add to the cube building time, and this in turn has an adverse impact on performance.

As a general practice, it is best to avoid Joins and Unions at the cube level. Combining data and transforming is best possible at the ETL level.

Another alternative to this is to use subviews or dashboards with sections having BI objects from different cubes at front end.

5.4 Design considerations for adding dimensions

Design considerations for adding multiple dimensions and dimensions with large number of values have been discussed earlier.

Dimensions with large number of values need not be a part of the cube but can be retrieved as flat data from data stored in Smarten cubes. One does not need to put such dimensions in the cube structure, but the Smarten cube engine will store flat data from data source resultset, and will

retrieve flat data from front-end operations such as drill through. If an analysis is made which needs to display data that is not a part of the UARS, it can be pulled from this data resultset stored in the cube.

For example, Country, State, and City are essential dimensions in the abovementioned example which are to be used in regular business analysis, but the invoice details are not. Country, State and City will be a part of the cube dimensions, and will form UARS, while data for invoice will not be a part of the UARS but can be retrieved from the resultset data part of the cube as and when required through a drill-through operation from analysis.

5.5 Design considerations for selecting dimensions

Selecting dimensions which provide meaningful hierarchy and analysis is important. While making decisions on dimensions, it is important to consult the consumer of the data and explain the implications of adding dimensions. An irrelevant dimension with a very large number of values will bloat the UARS. A good path to follow will be to classify the needs and select dimensions based on:

- Regular analysis requirement – This is what the user checks on a standard basis.
- Detailed irregular requirements – This has to be in the resultset data and pulled when required through drill through. This has been explained in the earlier section when presenting the design considerations for adding dimensions.

In the abovementioned example of UARS, analysis is not possible without Country, Region, City as dimensions. This is a regular analysis. At times, one has to analyze how some dealers are performing with help of invoice-level analysis. In such case, there is no need to keep the Invoice field as a dimension of the cube, but it can be stored in the data result set part of the cube, and can be retrieved with drill-through features in analysis as and when needed. Or, if other analysis operations are queried, a separate operational cube with the Invoice dimension can be created with different hierarchy, and user can tap in analysis from that cube whenever that irregular need arises.

5.6 Design considerations for Security Access Rights implementation

Smarten provides security at granular level – up to dimension value level. E.g. if you have country as a dimension, you can associate users with a value of this country dimension. E.g. user X has access to India and USA, while user Y has access only to USA and user Z has access only to India.

This level of security is an important and convenient feature, but has to be applied with appropriate design decisions so that it does not impact the performance.

Adding access rights at column data level hinders the overall performance of the cube, but this is avoidable unless necessary. It is better to design cubes in a way that the entire cube is accessible to a set of users rather than design a large cube where security is to be applied at each level for proper access.

In such case, if UARS is large, separate cubes are recommended along with a central cube with aggregated data. This has been discussed in depth in the section on Design consideration for defining the cubes.

5.7 Design considerations while adding custom measures or dimensions

The design choice of adding custom measures or dimensions should be based on necessity. Creating a custom dimension or measure when the results are obviously visible is not a good practice. For example, do we need a column showing the age when the date of birth is visible? Alternatively, rather than having the cube engine do this job of computing the age by including a custom dimension, it is prudent to do the same at design level.

5.8 Design considerations for the Time Dimension

Most of the data which needs to be analyzed will have a date field and/or a date-time stamp field in it. Depending on the business requirement, a design decision to add a date or date-time field as a time dimension is to be made. There are several trade-offs in terms of performance and advantages in making prudent choices:

- In case of data sets with multiple date/date-time fields, applying the date dimension increases the resultset by 5 columns, while applying time dimensions to all will increase the size of the UARS by another 3 columns.

  If one chooses to apply time dimensions on three date / time fields, then 24 new columns for analysis will be created.

  If you have an invoice date and a date-time stamp for invoice print date, it is prudent to choose the invoice date. If there are multiple date-time fields, then it is necessary to choose the select few date fields that matter for analysis. Choose account date or date time fields only if they are required. Otherwise they are overhead on performance.

- There is a clear difference between a date field which stores the date and a date time stamp.

  For example, a date field like invoice date will have just the date stored in it and will generate 5 more dimensions (from year, quarter, month, week and day) while the date time stamp will generate 8 more dimensions (from year to second).

  If your business is in telecommunication or retail or aviation, the time will play a role, and you may want to have the date time field as a dimension, but if you are analyzing sales data for bulk drugs, having time as a dimension just adds to the size of UARS, and should be avoided.

  If your data source has a date time stamp from which you only want to use the date part, it is best to cover it at the ETL level.

5.9 Design considerations for storing drill through data

The design choice of storing drill through data should be based on necessity. If there is no requirement to see transaction level data (via drill-through), one should not select this option. It will reduce cube creation/rebuild time as well as disk space usage.

5.10 Design considerations for storing aggregated data

The design choice of storing aggregated data should be based on nature of data. If aggregation is already done at data source or query level and there is no need to do another level of aggregation at cube level, one should not select “Perform aggregation” option. It will reduce cube creation/rebuild time as well as disk space usage.

5.11 Design considerations for selecting pre-defined data operations for measures

You can select data operations for each measures while creating the cube. Data operations available for cube are: sum, min, max, count and effective count.

One should select only required data aggregation operations for measures. Selecting few data operations (compared to all 5 available data operations) will reduce data processing time, increase response time and reduce CPU processes.

For example, for measure such as SalesAmount, if you do not need count or effective count, do not select these two data operations.

6 Object Cache

Cubes can be enabled with Managed Memory for optimised caching and performance. Similarly, front end objects such as analysis, graphs, KPI, reports and dashboards can also be enabled with Managed Memory.

For objects with Managed Memory configuration, object data will be stored in object cache. Object cache contains dimensions and measures used in the object, and object UARS (Unique Aggregated Result Set) is typically much smaller than the UARS of the cube from which the object is derived. Object cache is a subset of cube data and the Managed Memory configuration stores the aggregated result set of selected dimensions and measures of a particular object. When a user loads the object with Managed Memory configuration, data will be loaded from the object cache instead of the cube.

For example, if a cube is having 10 Million aggregated result-set, the front end report derived from this cube could have 100,000 aggregated result set. In that case, loading of front end object from the object's Managed Memory or cache will be much faster than reading cube data from the managed memory or cache, and processing that cube data to generate object UARS.

Object cache will be refreshed automatically after the cube is rebuilt. Administrators can put object cache into Managed Memory by selecting "Enable managed memory" for an object.

Reference: **Managed Memory Computing Concept document**
Reference: **The Workings of Managed Memory Computing document**
Reference: **Admin Manual > Application Configuration Settings > Managed Memory Configuration**

7 Product and Support Information

Find more information about ElegantJ BI-Smarten and its features at www.smarten.com
Support: support@smarten.com
Sales: sales@smarten.com
Feedback & Suggestions: support@smarten.com
Support & Knowledgebase Portal: support.smarten.com