Creating a universe on SAP HANA: Best Practices

Applies to:

This article applies to SAP BusinessObjects BI 4.0 (and its service packs up to SP2) and SAP HANA 1.0 (and its service packs up to SPS3)

Summary

This document presents best practices on creating universes and queries on a SAP HANA in memory computation engine. All the recommendations put forth in this document are driven by the intention to achieve best performance.

The target audience is users who have at least a basic understanding of the universe technology, and have read some general documentation about universe design. Links to the available documentation is provided at in the Related Links section.

Universes treat SAP HANA mainly as a relational database with some specific functionality. Detailed explanations are provided in this document when HANA's usage differs from a “typical” RDBMS and hence requires particular attention in the definition of a universe.

Due to the fast rate at which enhancements are being added in SAP HANA, this document is continuously updated. To be sure you are referring to the most up to date version review the document history and related links.

Authors: Pierpaolo Vezzosi, Patrice Le Bihan, Didier Mazoué, Elizabeth Imm

Company: SAP

Created on: 24 January 2012

Authors Bio

Pierpaolo Vezzosi is a solution manager for business intelligence on HANA and the semantic layer. Since joining the company in 2000 he’s held a variety of roles such as alliances manager and offshore manager for India. Pierpaolo holds a master’s of science in Aeronautics and Space Engineering. He currently resides in France and works out of the SAP Labs Paris.

Patrice Le Bihan is a Solution Architect within the CSA group (former RIG team) at SAP, delivering implementation best practices for SAP BusinessObjects BI products in general, and specifically when integrating with SAP NetWeaver and HANA. He currently resides in Canada and works out of the SAP Labs Vancouver.

Didier Mazoué is an expert product Manager for business intelligence on the semantic layer, specialized in most major OLAP databases, SAP Netweaver BW and relational databases. He has a background as a strategic pre-sales field representative specializing in closing large database deals. He currently resides in France and works out of the SAP Lab Paris.

Table of Contents

SAP HANA Basics

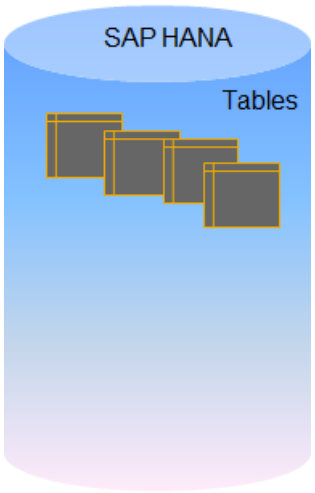
This section presents SAP HANA basic information which is of interest for a universe designer.

This is not intended as a general presentation of HANA. It is rather a simplified overview to help existing universe designers who are already familiar with building universes on relational databases, so they may quickly grasp key concepts to enable them to easily create a universe on HANA.

HANA as a relational database

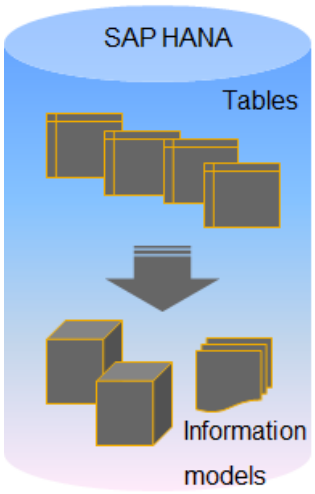
SAP HANA is a relational database whose tables can be stored in memory in a columnar format. This particular storage method is completely transparent for the designer who will see the usual logical representation of tables as rows and columns of data.

As a normal relational database, HANA, has the concepts of tables, joins, keys and SQL views (not to be confused with OLAP views -see HANA information models). HANA can be accessed via ODBC and JDBC drivers and its tables can be defined and queried with SQL language. Tables are managed with a tool called HANA Studio.



HANA information models

In HANA, it is possible to build business or calculation models on the tables. These models are called "Information models" and they are also known under the name of "column views", "HANA views", "HANA models" or “HANA cubes”. These models can provide a dimensional representation of the data in HANA and, in some circumstances, can be compared to OLAP cubes.



There are three information model types: attribute views, analytical views and calculation views.

Information model type	Description
Attribute view	Describes the dimensions of a data model and can be hierarchical (e.g. the "Customer" dimension, the "Geography" dimension; dimensions might have zero to many associated hierarchies).
Analytic view	Describes the facts related to some dimensions. Those facts come from a single fact table (e.g. the "Actual" revenue and cost)
Calculation view	Renders complex models that can combine multiple analytic views (i.e., to create a multi-fact cube) or that can be defined programmatically using the SQL script language.

Note:
The SQL Script language allows you to define complex calculations on data and can contain calls to low level functions (the 'CE" functions), to SQL, to R and L languages used for statistical and predictive analysis.

- All information models are represented as tables when used in a universe.
- Information models are found in the "Content" folders in HANA studio, they are found in the _SYS_BIC folder of the catalog in the information design tool.
- The hierarchical information is not yet available when connecting to information models via a universe.
- Information about languages, currency, client number is available but the functionality has to be recreated in a universe.

HANA engines

The HANA architecture is made of multiple engines, each of them is optimized to handle a specific task.

In a reporting scenario and as far as read queries are concerned, there are three main engines you should know about:

Engine name	Description
JOIN engine	Used to process joins across multiple tables
OLAP engine	Used to process the aggregation of large amounts of data with great performance. The OLAP engine works on Analytical Views
CALCULATION engine	Used to process more advanced models designed graphically or defined using programming languages such as R and L.

For best performance, SAP recommends leveraging the OLAP engine for aggregating large volumes of data.

Depending on which **database object** you query against, the engine that runs at query execution time varies. The engine selection does not depend on which **query language** you use. Whether you use MDX or SQL in your query, the choice of engine is not affected. In the following table, an example of an SQL query is used.

SQL Query	Invokes
SELECT… FROM TableA <some_join> TableB…	JOIN engine only
SELECT… FROM "_SYS_BIC"."AnalyticViewA"…	OLAP engine only If AnalyticViewA does not have any calculated attribute or measure
SELECT… FROM "_SYS_BIC"."AnalyticViewB"…	CALC engine for calculated attribute or measure OLAP engine for aggregation If AnalyticViewB has calculated attribute or measure
SELECT… FROM "_SYS_BIC"."CalculationViewA"…	CALC engine and OLAP engine if CalculationViewA is defined graphically and using analytic views
SELECT… FROM "_SYS_BIC"."CalculationViewB"…	CALC engine and JOIN engine If CalculationViewB is defined in SQLScript and using tables

Best practices for creating universes on SAP HANA

The following section includes some recommendations for creating universes on SAP HANA.

As the HANA technology is rapidly evolving, these recommendations are subject to change over time. Please ensure you are referencing the latest version of this document.

Version History

Version	Date	Remarks
V 1.0	November 28, 2011	Internal version published and based on SAP BusinessObjects BI 4.0 SP2 and SAP HANA 1.0 SPS3
V 1.1	January 24, 2012	Updated and published as white paper on SCN. Edited by Elizabeth Imm

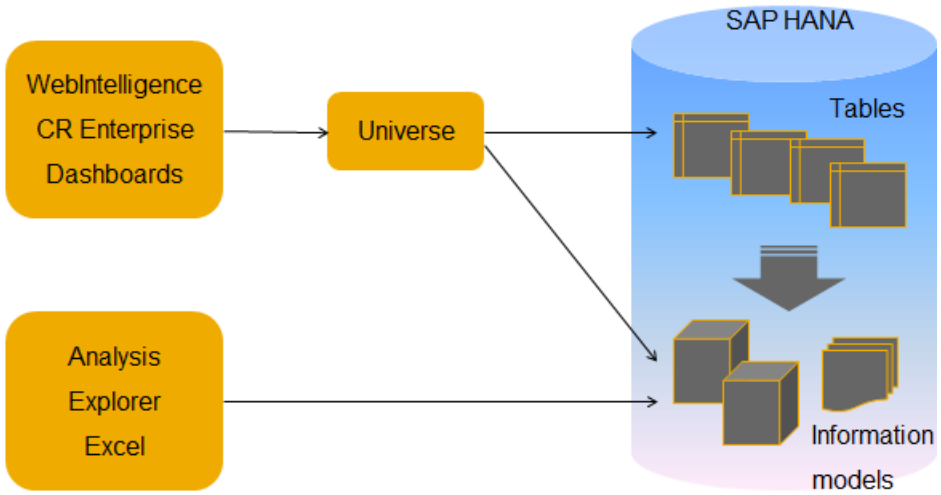
Choosing to create HANA information models or universes

When starting a new business intelligence project on HANA, the first question to ask yourself is whether you want to (or need to) create information models or universes.

Today, your choice mainly depends on the client tool that you want to use to access HANA:

Client tool	Requires a Universe	Access HANA data via
SBO Analysis Office **SBO Explorer (*)** **Microsoft Excel (PivotTable)**	No	an information model
SBO Web Intelligence **SBO Dashboards** **SBO Crystal Reports for Enterprise**	Yes	a universe built on tables or information models
SBO Crystal Reports 2011 / 2008	No	Tables (as of BI4.0 SP2)

(*) Explorer has the option to connect to universes; however, SAP recommends using the native Explorer connectivity to HANA views rather than using universes when accessing SAP HANA with Explorer.



When you create information models you benefit from:

- Availability to Analysis, Explorer, Excel, reusability in Universes
- SQL Script, L, R languages for programmatic calculation view
- Modeling in HANA Studio

When you create universes you benefit from:

- Higher customization of the output for business users
- Query time prompts and variables
- Embedded multisource technology to federate data from HANA with data from other sources
- Modeling in the information design tool

Creating a universe on HANA

As seen above in HANA, there are two main of database objects that you can query from a universe: tables and information models.

Information models are business or calculation models already defined into HANA and as such already contain a lot of intelligence which could be reused. Information models are more than simple relational tables, and come with additional functionality and constraints when queried with SQL.

Information models can be reused in a universe and take advantage of the business or calculation algorithms already defined into HANA Studio.

When building a universe on information models a particular attention should be given to the SQL constraints described in the section "Best practices when creating universes on HANA information models".

Decision criteria on whether to define a universe on HANA tables or information models

Build a universe on	If you
Columnar HANA **tables**	• Require full flexibility for an ad-hoc query experience (SQL on information models has some constraints listed below) • Want to avoid joining multiple information models (joins between information models might not be performing) • Need to extend the query space defined in an information model (you will have Information models and tables in your universe, you use aggregate awareness to select data either from the tables or from the models)

HANA **information models**	• Want to use the OLAP engine on Analytical views • Already have information models and you want to reuse them in your universes (taking into account the constraint in the "Best practices when creating a universe on information models") • Need complex calculations (for example, statistical, predictive, procedural) that are difficult to achieve in a universe with SQL but fit well into a calculation view • Need information models to be built for other client tools which do not access universes (for example SAP Analysis, Microsoft Excel)

Setting the correct parameters in the universe definition

The following parameters could improve the performance of the query when building a universe on HANA (both on tables and information models)

Set the correct Array Fetch Size value

Set this parameter in the connection definition (CNX) file where the default *Array Fetch Size* value is 10.

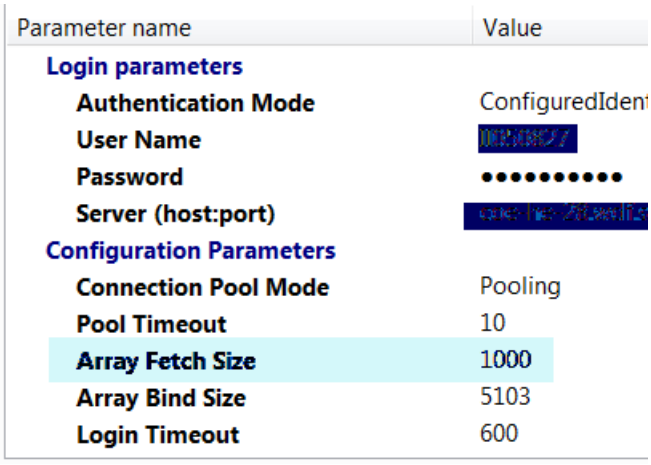
This means that, the BI query will fetch 10 rows at a time from HANA.

Increase this value for faster response times.

The higher the parameter, the more memory will be consumed on the machine that processes the query (the BI platform server or the client tool).

For example, tests run with a value of Array Fetch Size of 1000, showed a good performance increase while maintaining reasonably low memory consumption.

This parameter should, however, be lowered when retrieving BLOBs or long text fields; the exact value depends on the average size of those fields.

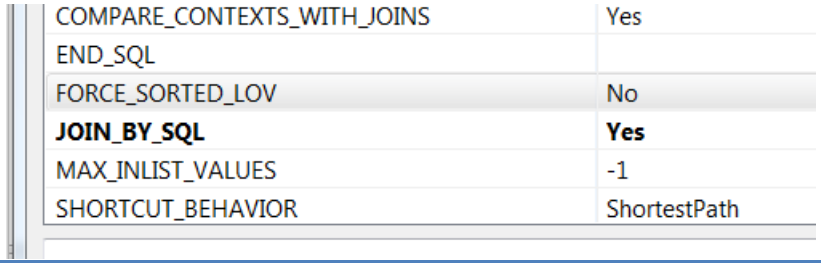


Ensure that query synchronization happens in HANA and not in the client tool

For some queries (e.g. when you query on multiple fact tables), multiple SQL statements might be generated. The results of those statements need to be synchronized in a single result set.

By default, this synchronization happens in the client tool. It is possible to combine the multiple statements in a single one which is pushed to the database. With this technique the database and not the client tool, takes care of the synchronization.

In order to push the single synchronized query to the database you have to set the Data Foundation (DFX file) parameter **JOIN_BY_SQL** to **YES**.



COMPARE_CONTEXTS_WITH_JOINS	Yes
END_SQL	
FORCE_SORTED_LOV	No
JOIN_BY_SQL	**Yes**
MAX_INLIST_VALUES	-1
SHORTCUT_BEHAVIOR	ShortestPath

Best practices when creating universes on HANA information models

Information models (analytical views and calculation views) will have the same look as tables in the universe data foundation.

Note: The "data foundation" concept in a universe is the set of all tables that are used to define your data source. This is different from the concept of data foundation found in HANA studio.

Information models often contain the business or calculation algorithms needed for the query. In this scenario, if possible, **perform all the modeling you need at the HANA Studio** level to avoid adding functionality which may impact the query performance.

Moreover, joining multiple information models has a strong negative impact on the optimization engines.

General recommendation for using information models in a universe
Information models **should not be joined to anything**. You can either use you single information model per universe or, if you have multiple views in a universe use the Aggregate_Awareness and Incompatible Objects functionality to make sure that queries run on a single view (unless performance is not an issue).
If possible, avoid adding calculated dimensions or measures in the universe and add them into the information model itself.
In a data foundation that is based on information models, if the attributes of the views have set description mappings, do not set index awareness for the dimension objects. Use the <attribute_name> description field of the view in the definition of the dimension object and let HANA handle the key mapping internally.

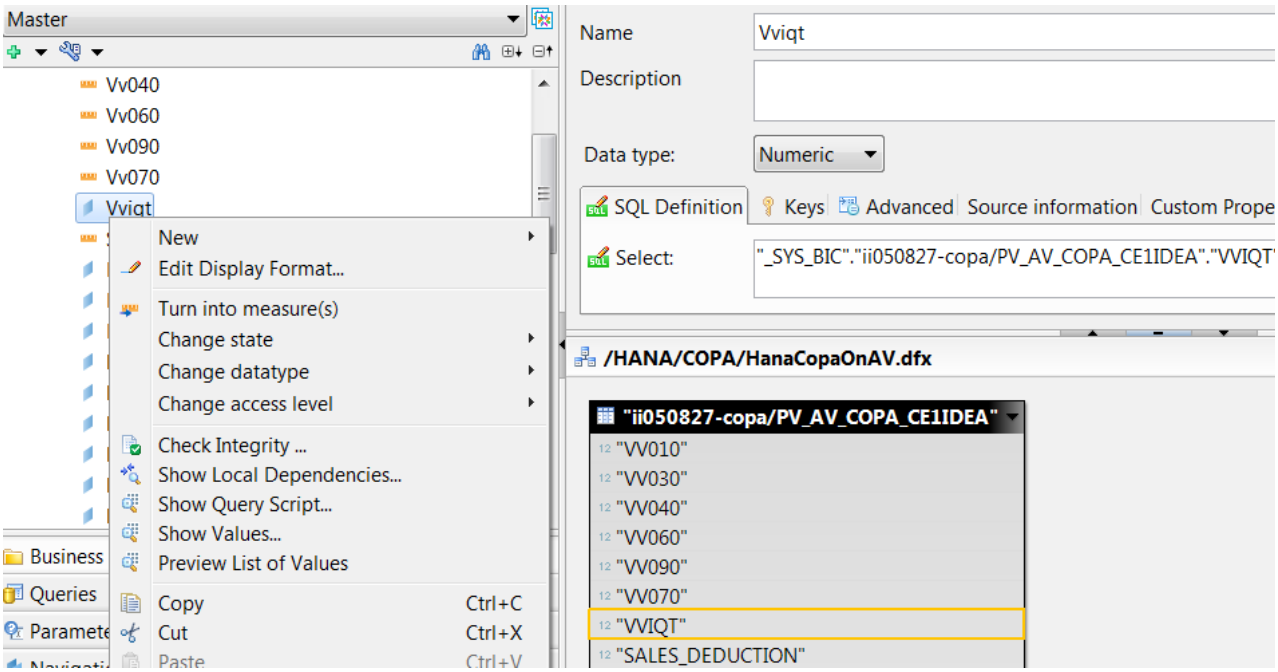
General recommendations for universes on calculation views
Calculation views provide access to SQL Script, L- and R- languages and calls to low-level CE functions. This access is generally not available in a universe
Universes built on the tables defining the Calculation view will retrieve only the data necessary in the query. While calculation views will provide the whole set of data from the tables, universes will work only with the data actually needed for the query.
Universes built on a programmatic calculation view (as any other tool accessing using SQL) will execute the whole calculation view algorithm. Data not needed in the BI query will be anyway calculated in the HANA engine.

Pre-requisite knowledge needed when using information models in a universe
The information models appear as tables in the information design tool but they are not tables! The **"Show Table Values" command will fail** on an Analytical View because of the HANA constraints when querying an information model with SQL. When you query Analytical Views a `group by` or a `select distinct` statement is required. The **Show Tables Value** command sends instead a plain Select statement with no groupings.
Measures that are defined in the information model, are seen as dimensions in the information design tool (as any other field of the table). Make sure you change them to measures, wrap them around the correct aggregation functions and the correct projection function. SAP recommends that you use the same aggregation function defined in the information model measure via the HANA studio.

Example workflow

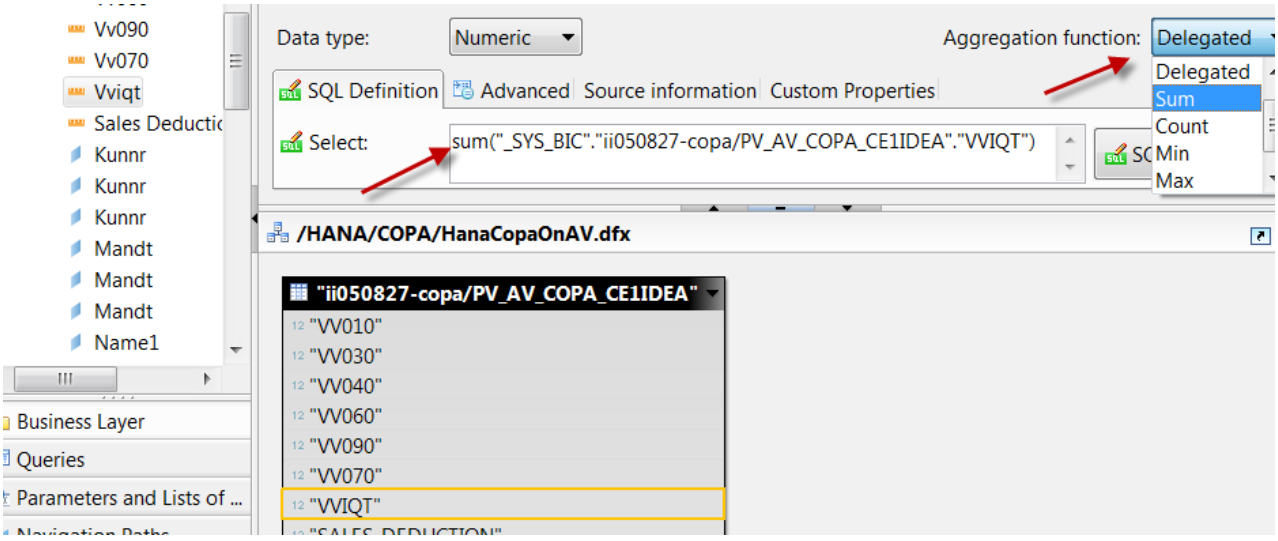
In this example, the **Vviqt** object is defined as a measure in the HANA information model but is interpreted as a dimension in the information design tool. To make sure it is correctly defined as a measure follow these steps:

1. In the information design tool business layer, select the object, then **right click** and select "**Turn into measure(s)**"



2. Set the correct aggregation function in the SQL definition (sum() in this example, and set the correct projection function in the drop down box.

Note: the 'projection function' is labeled 'Aggregation function' in the user interface. This label will change in an upcoming release of the information design tool.



What is the projection function?

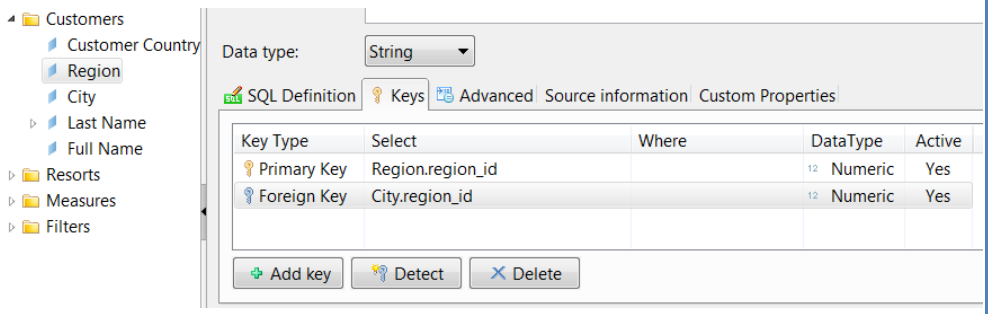
The projection function tells the client tool (i.e., Web Intelligence) how to aggregate the data locally. When the query runs on the database, the data is aggregated by the aggregation function used in the SQL definition of the measure. Once the data is in the Web Intelligence report, it is possible to change

the aggregation level to a coarser one (for example, you query on Country, City, Revenue, then in the Web Intelligence report you show only Country and Revenue). SUM runs a local sum; the DELEGATED projection function pushes the calculations down to the database instead of executing it locally.

Best practices when creating universes on HANA with either tables or information models

In the list of recommendations below, keep in mind all the items from the previous section about information models, especially that SAP strongly recommends that you do not join information models with anything else.

Best Practice	Explanation
Prefer columnar tables for reporting	Tables in the column store provide a faster read, filtering and aggregation time than tables in the row store. When creating your HANA schema, if possible, define the tables used for reporting as Column Based. In your universe project, whenever possible make use of columnar tables instead of row stored tables
Single source and multi-source universes	Use multi-source enabled data foundations on HANA **only if** you really need to federate data from multiple sources. The processing of a multi-source enabled query is complex and will decrease the performance. This performance cost is justified if multi-source queries are a necessity but is to be avoided when you just need to get data from HANA and no other database.
Avoid calculations in the query generation outside of the select statement	Avoid calculations in filters or GROUP BY For example, `group by T1.”user_ID”+T1.user_code”` For example, `where a*b=10` Prefer defining table columns which already contain the calculated values
Type casting	Avoid implicit type casting e.g.: (slow) `where date_string > Now()` (fast) `where date_string > TO_CHAR(Now(), 'YYYYMMDD')`
Joins optimization	Avoid non equi-joins, when possible Avoid calculation in joins, when possible Example, `T1.”user_ID”+T1.”user_code” = T2.” user_global_”` Define table columns that already contain the calculated values
Predicates optimization (applies mainly to the definition of universe derived tables)	Avoid using Exists and OR in the same filter Use Not Exists instead of Not In predicates Avoid Union, Intersect, Except predicates Usually those predicates are found in derived tables Prefer Coalesce rather than Union Except and Intersect can sometimes be substituted with better filters

Index awareness	When creating a universe on tables using Index Awareness may improve the performance as queries will be run on fewer tables and be filtered on keys rather than labels. To set index awareness for an object, in the business layer you must define its primary key and all of its foreign keys, found in the data foundation.  ***Warning: Index Awareness might not be a good choice if the universe is based on HANA information models (see the previous section)***
Apply data filtering techniques in the Universe	To reduce the quantity of data retrieved by a user, try to set compulsory filters which will limit the query result set. Row-level security mechanisms in a universe allow reducing the visible data for specific users based on their profiles

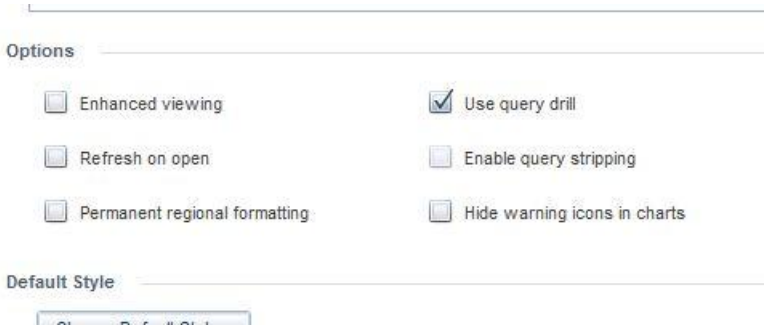
Best practices when creating queries based on universes on HANA

The power of HANA is in manipulating, aggregating, and filtering large quantity of data.

Large quantities of data should not be transferred to the client tool.

To benefit from HANA's calculation performance, retrieve the least quantity of data in your report. HANA can easily provide sub-second response times while handling millions or records but the client tool on your laptop won't be able to provide the same performance.

In your query retrieve only what you need
Always retrieve the least possible quantity of data in your client tool using filters or restricted list of values
Retrieve aggregate data, not detailed, whenever possible
Retrieve only the columns you need: HANA being a columnar database, if you don't ask for a column, the system will not spend any time on it (unless the column is used in an internal calculation)
Build your universe to avoid the need of multiple queries on the same schema from the client tool. 'Query on query' functionality could be avoided by table joins.
In Web Intelligence, select the "online query-drill" option to delegate the data display to HANA. Rather than retrieving all columns in the document, you will be able to drill down and up in the document whereby retrieving only the needed amount of data through automatic query filtering as you drill. From the Web Intelligence report properties page make sure you select the 'Use query drill' option:



In your report, consider using multiple small and very narrow queries rather than a single huge query. In some situations you might retrieve less data with multiple targeted questions.

Related Content

Semantic Layer forum on SDN: http://forums.sdn.sap.com/forum.jspa?forumID=308

BI on HANA basic tutorial: http://www.sdn.sap.com/irj/scn/bi-suite-tutorials?rid=/library/uuid/703298ef-02f9-2e10-2691-cfef154fd920

Information design tool on HANA viewlets: http://www.sdn.sap.com/irj/scn/info-design-tool-elearning?refer=main

Copyright