International Journal of Leading Research Publication (IJLRP)
E-ISSN: 2582-8010 ● Website: www.ijlrp.com ● Email: editor@ijlrp.com

Optimizing SAP S/4 HANA Reporting - Best practices using CDS views and Bex Queries

Kumail Saifuddin Saif

kumail.saif@gmail.com
SAP Technical Architect & Projects Delivery Manager, Accenture LLP, USA

Abstract
SAP S/4 HANA Embedded Analytics provides ability to report on the operational data directly using ABAP CDS views and BEx queries without using the traditional data warehousing design and ETL operations. However, it is very important to consider the best practices used for CDS view design and BEx query design, so that the performance of the report is not negatively impacted. This paper discusses in detail the design principles and best practices recommended for both of these objects and how they can help develop a performance efficient report.

Keywords: SAP S/4 HANA, Embedded Analytics, Operational Reporting, ABAP CDS Views, SET Hierarchies, CDS based virtual hierarchies, HRRP

1 Introduction: SAP S/4HANA Embedded Analytics leverages Core Data Services (CDS) views and BEx Queries to provide real-time reporting and analytics capabilities. These components are foundational in designing reports that enable business users to analyze data effectively. Performance is always a key aspect of any report development and hence it is necessary to consider the Best practices and guidelines while developing reports using CDS views and BEx Queries. We will discuss in detail the key design principles to be kept in mind to make sure the reports developed are efficient.

2 CDS Views Development Best Practices:
The ABAP CDS view framework was introduced to leverage the computational power of HANA DB. Data model turns into pure SQL when the CDS view is accessed from ABAP or HANA studio. During execution, the data being read is declared in a single SQL string that gets processed by the SQL optimizer to create an execution plan. Hence it is very important to understand how the SQL query is executed in the database and what are the design principles to be considered. Following sections discuss them in detail.

2.1 Avoiding Implicit Type Casting in Queries - If there is a comparison between a VARCHAR value and a DATE value, the system generates an implicit type-casting operation that casts the VARCHAR value into a DATE value. You can avoid implicit type casting by instead using explicit type casting or by adding generated columns. The system can generate type castings implicitly even if you did not explicitly write a type-casting operation. If two columns are frequently compared by queries, it is better to ensure that they both have the same data type from the beginning. One way to avoid the cost of implicit type casting is by using explicit type casting on an inexpensive part of the query. For instance, if

a VARCHAR column is compared with a DATE value and you know that casting the DATE value into a VARCHAR value produces what you want, it is recommended to cast the DATE value into a VARCHAR value.

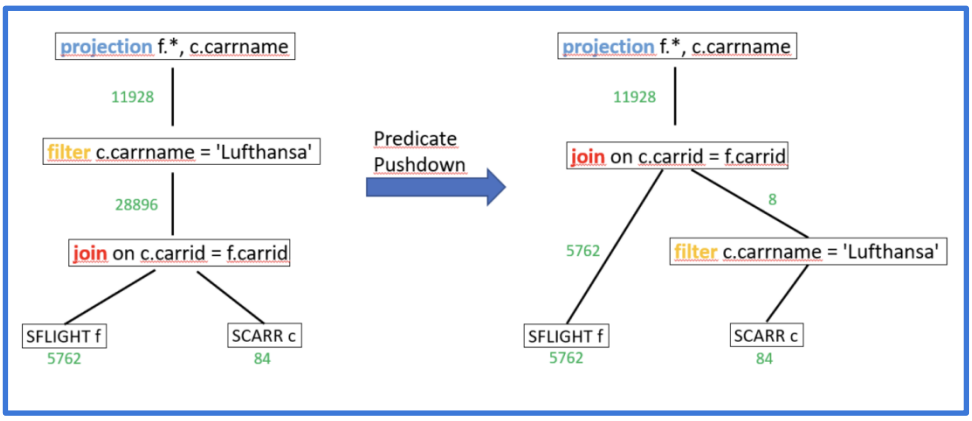
2.2 Calculated fields - Calculated fields should be avoided in general as it creates the materialization and intermediate storing of the records. Additionally calculated fields should be avoided in the WHERE-clauses and ON-clauses of CDS views, especially when tables with many entries are involved. Calculated fields result from functions like string functions (CONCAT, RTRIM) and CASE expressions. Note that all fields provided by a table function act as calculated. For a calculated field to be used (as filter or join condition) the values of this field for every row must be evaluated at this point of processing. That may be very time-consuming.

2.3 GROUP BY clause - Avoid GROUP BY clause as it can force the materialization of records which is expensive.

2.4 Constants - Avoid constants in ELSE branches within CASE especially for views to be reused in joins as it potentially limits DB optimization due to not preserving the null value. The same limitation applies to fields defined as constant if the view in which they are defined is on the right side of a LEFT OUTER JOIN.

2.5 ASSOCIATIONS instead of JOINS -
Use associations to improve performance: If they are just defined and exposed within the projection list without direct exposition of fields, they are pure metadata; only if used in a path expression they become standard joins.

2.6 Use FIlters as much as possible -
Using filters limits the number of records fetched from the database and helps improve the performance. Filters can be set via WHERE-clauses in CDS views. They are always propagated through projections. Filters are propagated through joins only via fields in ON-condition. However they are not pushed through calculations (functions).



2. 7 Intermediate Results - Minimizing intermediate results in ABAP CDS views is crucial for optimizing performance, especially when using JOINs. Applying filters as early as possible in the execution plan reduces the number of records processed during later stages, leading to faster query execution and reduced resource usage.

- Instead of filtering after the join, include the filtering conditions in the ON clause of the JOIN. This restricts the data fetched from each table before the join operation.
- Apply filters on key fields whenever possible. This helps the database optimizer use indexes effectively, reducing the dataset size early.

2.8 Materialization - Materialization of records refers to the process where intermediate or final query results are temporarily stored in memory or a physical location (e.g., temporary tables) during the execution of a CDS view or SQL query. This occurs when the query execution engine needs to persist partial results before continuing with the next operations. Materialization can lead to performance issues because it increases memory usage and involves additional I/O operations. Minimizing materialization is essential to optimize performance in ABAP CDS views. Below are some of the techniques to minimize Materialization in ABAP CDS Views:

- Apply filters at the source or in the earliest stage of query execution to reduce the number of records fetched and processed.
- Minimize the use of deeply nested JOINs or associations that force the database to materialize intermediate results. instead, use direct associations with path expressions.
- Avoid aggregation operations like SUM, AVG, or COUNT. They can force the materialization of records before the result set is finalized.
- Annotations like @Analytics.dataCategory or @DefaultAggregation guide the query engine on how to handle data efficiently, reducing unnecessary intermediate computations.
- Move complex calculations to a later layer (e.g., application layer or reporting tool) instead of performing them within the CDS view, as they may force intermediate materialization.

2.9 Annotations for performance - Following annotations should be used as a Best Practice. These are used in SAP standard CDS views as well and help proper execution of the CDS view by respective engines as per design.

- @ObjectModel.usageType.serviceQuality
- @ObjectModel.usageType.sizeCategory
- @ObjectModel.usageType.dataClass

General Guidelines -

- Expose only required fields to reduce the number of accessed tables, used joins and operations. SELECT * statement requires all columns to be accessed, which is costly for the HANA column store.
- Avoid cyclic associations or cyclic joins with involvement of left outer join, especially when tables with many entries are involved.
- To achieve a certain business purpose, choose the simplest CDS view possible. Do not create an all-in- one complex view covering all demands.

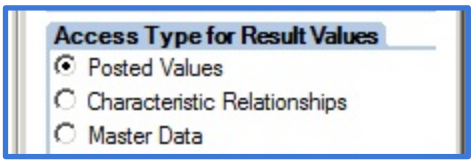
- Check if UNION clauses could be safely replaced by UNION ALL without changing the semantics.
- Prefer UNION ALL over CASE expressions in the models as they allow more room for preferable choices for the database optimizer.
- Extend standard SAP-delivered CDS views instead of creating entirely new ones.

3 BEx Query design Best Practices:

Designing effective SAP BEx Queries is crucial for delivering efficient, user-friendly, and insightful reports. Following best practices ensures that queries are optimized for performance and usability, meeting business needs effectively. Here are key best practices for designing SAP BEx Queries:

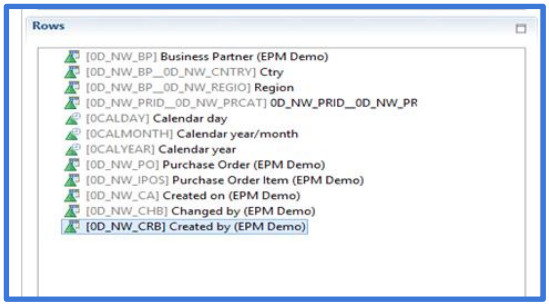
3.1 BEx Query Design Guidelines:

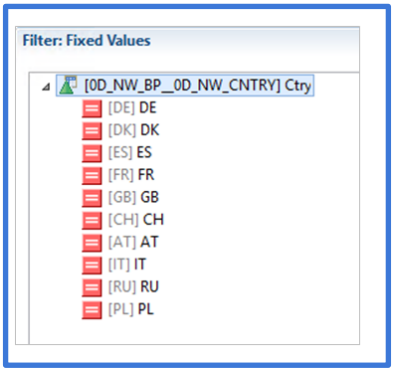
- Minimize Data Volume: Limit the dataset by applying filters and variables to retrieve only the necessary data.
- Mandatory Variables: Use mandatory input variables. Variables improve query performance by making data requests more specific. You can add a mandatory characteristic value variable so that reporting users must define a filter. This ensures that not all the detail values are read from the InfoProvider.
- Default Values: Provide default values for variables to guide users and improve query usability.
- Apply exception aggregation (e.g., averages or counts) only when necessary to improve performance.
- Use display attributes and suppress characteristics not needed in the report to simplify output.
- Enable the OLAP cache to store query results and reduce processing time for repeated executions.
- Remove unused query elements such as key figures or characteristics to streamline processing.
- Minimize Use of Calculated and Restricted Key Figures. These can increase processing time when overused.
- Avoid master Data Access Mode setting for the characteristics, specially the ones which are used in the default view. This creates a performance expensive operation.



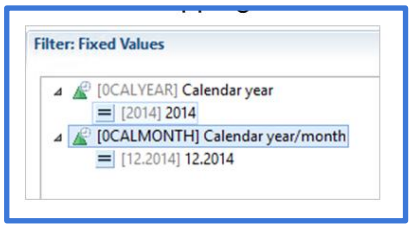
- Avoid too many drilldown characteristics in the initial view. Keep the number of characteristics in the rows or columns to a minimum and make use of free characteristics. Too many characteristics in the rows or columns means that the processor has to work hard to retrieve and format the extra data for all levels of the drilldown.
- Avoid exclude function - The Exclude function within characteristic filtering is useful when you want to eliminate certain values from a range of values in the query. However, use this function sparingly, because too many exclusion instructions can start to slow down the query runtime.

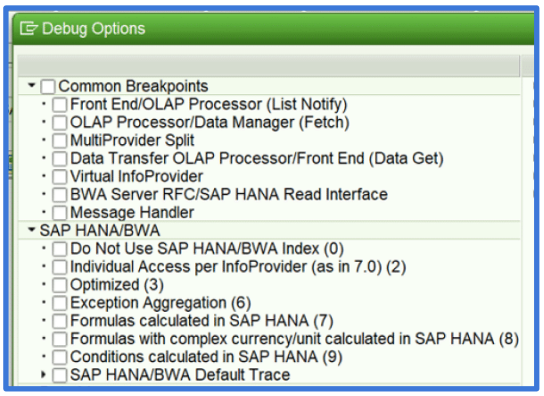
Consider using inclusions where possible, even if this means that the initial setup of the filters is more complex.





- Avoid overlapping filters - A common query design error is causing the OLAP processor to perform unnecessary steps to retrieve data. For example, if your query contains the characteristic Cal. Year/Month, filtered to the value 12.2014, and the query also includes the characteristic Calendar Year that is filtered to the fixed value 2014, you have a filter overlap that can affect performance.





- Read Mode Settings - A query read mode that does not meet the actual requirements can decrease performance. Unnecessary data may be loaded that is not used in the report. Sometimes it is difficult to find the best Query read mode by looking at the query definition and the key figures design. Hence it is always recommended to test the query performance in transaction RSRT using Execute + Debug option and selecting different runtime modes.
- Query Execution Settings - The query execution mode determines the push down of query calculation operations from the ABAP runtime to the SAP HANA database to improve performance significantly.

Conclusion:

Incorporating ABAP CDS view design best practices and BEx query design principles is essential for crafting performance-friendly reports that meet business requirements effectively. By minimizing intermediate results, reducing materialization, and leveraging efficient filtering, CDS views ensure that only the required data is processed, enhancing performance. Similarly, BEx query principles like

leveraging variables, filters, and query execution modes, access modes for characteristics contribute to efficient data retrieval and presentation. Together, these practices lead to scalable, intuitive, and high-performing reports that empower users with timely insights while optimizing system resources.

References

1 - What is SAP HANA? [Online]. Available at: https://www.ibm.com/topics/sap-hana
2 -SAP HANA Installing and administering. SAP TRAINING. [Online]. Available at: https://learning.sap.com/learning-journeys/installing-and-administering-sap-hana
3 - S/4HANA Embedded Analytics [Online]. Available at: https://help.sap.com/docs/SAP_S4HANA_ON-PREMISE/6b356c79dea443c4bbeeaf0865e04207/c53deb5765c7be12e10000000a4450e5.html
4 - 2414215 - Custom Hierarchy Replication using HRRP_REP [Online]. Available at: https://userapps.support.sap.com/sap/support/knowledge/en/2414215
5 - Replicate Runtime Hierarchy [Online]. Available at: https://help.sap.com/docs/SAP_S4HANA_ON-PREMISE/5e23dc8fe9be4fd496f8ab556667ea05/d477e555cd3b7b43e10000000a4450e5.html?locale=en-US&version=2020.latest
6 - SAP - ABAP CDS Development User Guide [Online]. Available at: https://help.sap.com/docs/SAP_NETWEAVER_AS_ABAP_752/f2e545608079437ab165c105649b89db/7c078765ec6d4e6b88b71bdaf8a2bd9f.html
7 - VDM Annotations [Online]. Available at: https://help.sap.com/doc/saphelp_nw75/7.5.5/en-US/ef/e9c80fc6ba4db692e08340c9151a17/content.htm?no_cache=true
8 - Financial Statement Versions [Online]. Available at: https://help.sap.com/docs/SAP_S4HANA_ON-PREMISE/651d8af3ea974ad1a4d74449122c620e/c064c2531bb9b44ce10000000a174cb4.html?locale=en-US&version=2020.latest
9 - Analytical Queries [Online]. Available at: https://help.sap.com/docs/abap-cloud/abap-data-models/cds-analytical-queries
10 - Virtual Data Model and CDS Views in SAP S/4HANA [Online]. Available at: https://help.sap.com/docs/SAP_S4HANA_ON-PREMISE/ee6ff9b281d8448f96b4fe6c89f2bdc8/8573b810511948c8a99c0672abc159aa.html
11 - ABAP Core Data Services | S/4HANA - Best Practice Guide https://www.sap.com/documents/2019/01/0e6d5904-367d-0010-87a3-c30de2ffd8ff.html
12 - Create an ABAP Project in ABAP Development Tools (ADT) https://developers.sap.com/tutorials/abap-create-project.html