International Journal of Management, IT & Engineering
Vol. 14 Issue 06, June 2024,
ISSN: 2249-0558 Impact Factor: 7.119
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gate as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Strategy to Handle Large Tables in SAP S/4 HANA

Saurav Mago

Abstract

Database tables grow significantly over the period of time which can cause significant performance issues. In this paper, we will discuss the ways to handle large tables in SAP S/4HANA.

Keywords:

SAP.
SAP S/4HANA.
Data Archival.
Table Partitioning.

Author correspondence:

Saurav Mago,
ERP SAP Manager &
Technology Lead
Email:
saurav.mago@gmail.com

1. Introduction

There are two type of tables which SAP HANA database supports – Row-wise (Row Tables) & Column-wise (Column Tables). As the name suggests, row tables store complete row of a table in an immediate memory whereas column table stores entries of a particular column in an immediate memory. SAP HANA database is optimized for column tables for faster read operations (due to data compression) with good performance for write operations also. Column Tables are suitable for heavy table with write operations whereas Row Tables are more suitable for small tables with frequent update/insert operations. Partitioning of tables is only feasible for column tables.

There are scenarios where SAP tables grow significantly over the course of time and can cause performance issues. Example: Characteristic (AUSP), IDOC (EDIDD), Finance (ACDOCA, BKPF, BSEG) etc.

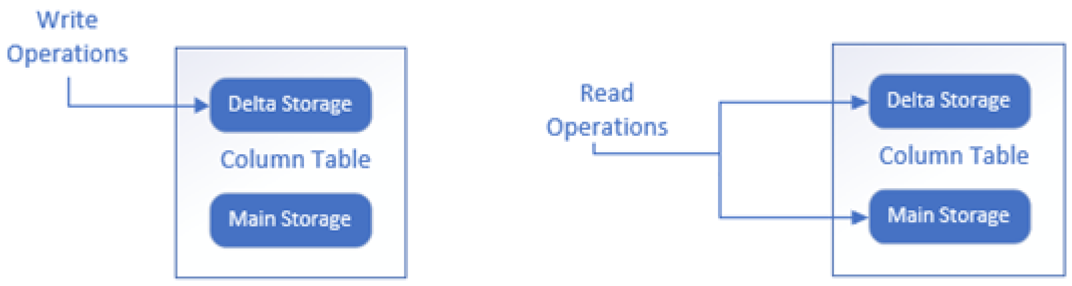
To control large tables, data archiving & partitioning techniques can be leveraged which is explained in subsequent sections.

2. Partitioning of Large Tables

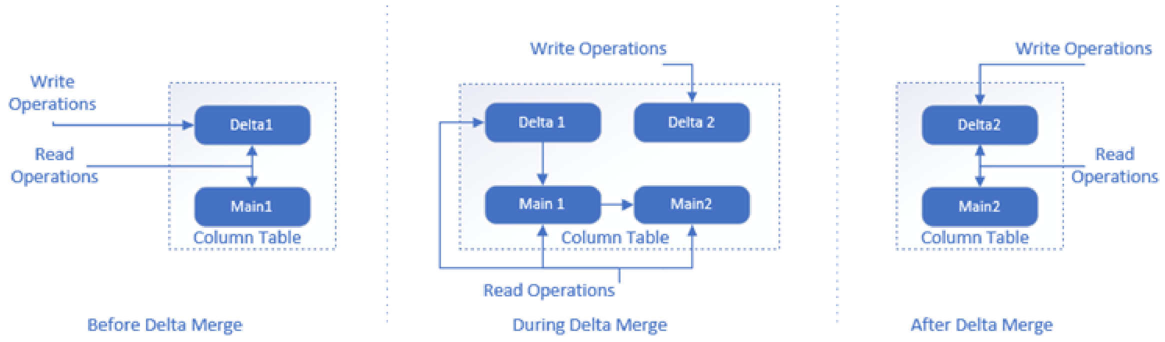
Partitioning breaks down a huge table into smaller parts which are easy to manage. There are multiple ways to partition a table like Range Partition, Hash Partition & Round-robin partition. Depending upon the use-case, analysis by database administration & system architect, we should finalise the type of partition to be used as it requires downtime and repartition will be more difficult exercise.

Before we proceed on to learn about different types of partitioning, we should understand how read/write operations typically happens in a column table. Column tables are optimized for read procedures and provides good performance for write procedures. This is achieved via Main storage & Delta Storage.

- Main part of the data is stored in Main storage where data compression happens to save memory and improve read performance
- Write operations on Main storage will be performance intensive hence avoided
- All changes to be written are stored in Delta Storage which is optimized for write operations
- All the data will be read from both the storage to get latest information
- All the changes are moved from Delta storage to Main storage which is called as Delta Merge
- Once Delta Merge starts –
  - All data from Delta1 is copied into Main1
  - All write operations go into new delta storage i.e., Delta 2
  - All read operations go from Delta1, Delta 2 & Main1



- Once Delta Merge is complete –
  - Delta1 & Main1 is deleted
  - Main1 is copied into Main2 which causes temporary increase in Memory & utilize CPU resources. This potentially negative impact on performance can be mitigated by splitting tables into smaller parts (partitions), which will allow each part to have its own main & delta storage. Delta merge happens only for that partition which requires it, and this ensures less data is merged & improves the performance
  - Optimal compression happens in the Main storage



Delta Merge & Compression are ongoing activity for a column table. Once we perform the partition, each set of partition will perform its own merge & compression with a smaller dataset thus enhancing the performance

Types of Partition:

Hash	Range	Round Robin
Used to distribute records to partitions equally for load balancing	Not suited for load distribution	Used to distribute records to partitions equally for load balancing
Utilizes Hash algorithm for record distribution	Used to distribute records to partitions based on range specified	Used to distribute data equally in a rotation basis
No in-depth knowledge of actual content of data	Requires in-depth knowledge of actual content of data	No in-depth knowledge of actual content of data
Columns must be specified as partitioning columns	Columns must be specified as partitioning columns	Does not require to specify partitioning columns
Partitioning columns must be part of key field of table	Partitioning columns must be part of key field of table	Table must not have primary keys
Ex: CREATE COLUMN TABLE ZTABLE_TEST (a INT, b INT, c INT, d INT PRIMARY KEY (a,c)) PARTITION BY HASH (a) PARTITIONS 4	Ex: CREATE COLUMN TABLE ZTABLE_TEST (a INT, b INT, c INT, d INT PRIMARY KEY (a,c)) PARTITION BY RANGE (c) (PARTITION 10 <= VALUES < 50, PARTITION 50 <= VALUES < 200, PARTITION OTHERS)	Ex: CREATE COLUMN TABLE ZTABLE_TEST (a INT, b INT, c INT d INT) PARTITION BY ROUNDROBIN PARTITIONS 5

- Partitioning column can have data types as STRING, TINYINT, SMALLINT, INT, BIGINT, DECIMAL, SHORTTEXT, VARCHAR, NVARCHAR, DATE, TIMESTAMP, SECONDDATE, FIXED, RAW
- There can be scenarios where non-primary key columns need to be added as partitioning columns. In such cases, multi-level partitioning can help which allows to select primary key column as first level and non-primary column as second level. Hash-hash, hash-range, range-range, round robin-range are such possible multi-level partitioning techniques.

Key Points for Partitioning:

- Each partition should have at least 100 million records.

- Memory use for each partition should not be more than 50GB at any given point of time else it will cause delta merge issues.
- Any table (without partition) which has more than 2.14 billion records will lead to system crash.
- Repartition should be planned if any partition reaches 1.6 billion records.
- Repartition should be done as a multiple of existing partition. Ex: If current number of partitions are 16, then new partitions can be 16*2, 16*3 etc.
- To finalize the partition strategy, always analyze the 'Where' clause of all SQL statements on table to be partitioned and try to look for common patter. Below query can be used to get the result:
- select statement_string, execution_count from m_sql_plan_cache where statement_string like '%<Table Name>%' order by execution_count desc.
- SAP HANA supports 2 level of partition only (first level & second level)
- One table can have maximum of 1000 (SPS <= 09) or 16000 (SPS >= 10) partitions
- HANA parameters should be adjusted based on number of partitions. Ex: If number of partitions are 60 then system will utilize 1 CPU thread for each partition, hence statement concurrency limit should be adjusted to support partitions else system will be slow due to resource unavailability

Tuning HANA Parameters:

HANA parameters play a key role in performance tuning of the system. There are couple of parameters which are linked to table partitioning (delta merge, compressions etc.) so we will discuss a few important parameters here. During application usage by SAP users, there can be performance issues which leads to the system being unresponsive or out of memory dump issues. SAP HANA environments might have resource bottlenecks due to limitations of CPU/threads (Number of physical/logical CPU), memory (Physical memory), or network (bandwidth & latency). SAP HANA operations like Delta Merge, Compression can interfere with the heavy volume application activities and degrade the performance. Hence, in addition to Application Team performing the analysis, HANA Database administrator should monitor & validate below Parameters settings:

1. default_statement_concurrency_limit:

    - SAP HANA tasks like executing of SQL statements or background tasks is performed by threads.
    - This parameter decides the number of CPU threads that can be used for a database request & causes high CPU utilization if not maintained properly.
    - Recommended value is MIN (10 %, 8) - 50 % of available CPU threads.
    - CPU Consumption can be checked via SAP HANA Studio -> Administration -> Overview -> CPU Usage
    - Changes to this parameter is reflected at once; no restart required.

2. statement_memory_limit

    - It decides the maximum memory consumption for a single SQL statement & causes out of memory (OOM) issues if limit is breached.
    - Changes to this parameter is reflected at once; no restart required.
    - Recommended value is MIN (10 % of allocation limit, 30 GB) - MIN (30 % of allocation limit, 500 GB)

3. row_order_optimizer_threads:

    - This parameter controls the number of parallel threads during optimize compression runs.
    - Available as of SAP HANA Rev. 1.00.112.02
    - Recommended value is MAX (4, CPU threads / 10) as lower limit for SAP HANA >= 2.00.030

4. max_concurrency

- It decides total number of CPU threads that can be used by parallelized requests (values below 50% only required in rare cases)
- Recommended value is 33-100 % of available CPU threads.

5. max_concurrency_hint

- Maximum number of CPU threads that can be used by a single parallelized operation, optimal setting is individual.
- This is identical to num_cores parameter.
- Recommended value is MIN (10 %, 8) - 50 % of available CPU threads.

Overall, it is of high importance that parameters for SAP HANA and other involved system components are set according to the best practices.

3. Archiving of Large Tables:

Data Archival simply moves the data from the high-performance, costlier HANA database to a disk-based storage environment which is more cost effective. The archived data can still be read until purged. Data is archived using archiving objects, which describe the data structure and context. The Archive Development Kit (ADK) is the technical framework which provides all the tools for the SAP data archiving solution. It forms an interface between applications, database, and the place where archived data is stored.

Benefits:

- Deleting the old/historical data as per the compliance and policy leads to reduction in footprint within the database & improves the system performance.
- Archival helps to move the data from High-cost environment to the low-cost secondary database (example: SAP IQ)
- Archived data can be accessed using some of the SAP standard reports which are provisioned to pull the data from the Archived database (example: SAP IQ)

Steps for Archiving:

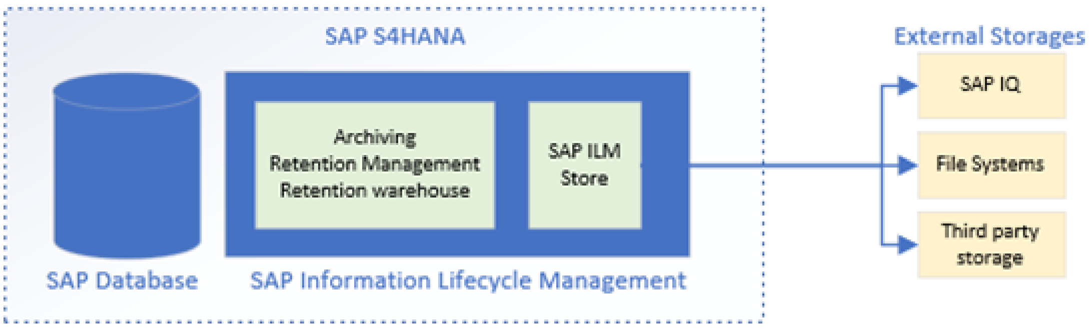
Before we start with Archiving, the first step is to identify large tables in terms of memory size via SAP Transaction DB02. Once huge tables are identified, SAP Transaction DB15 is used to identify relevant Archive Objects for those tables. There are multiple steps to achieve Archival:

1. Preprocessing (Optional)
At this step, system prepares the data for archival and makes a deletion indicator. No data is removed at this step.
2. Write
System selects the chosen records, creates new archive file & writes the records into the archive file. No data is removed yet from database.
3. Delete
At this step, system reads the data from the archived files and removes the data from the database.
4. Post processing:
It also runs on the Database and does not require any Archive files. If the data is not removed via Delete step, then it can be deleted via post processing program.

SAP ILM:

- SAP Information Lifecycle Management (SAP ILM) enhances the SAP standard delivery with the ability to manage the lifecycle of live and archived data based on rules.

- SAP ILM helps to manage information through its lifecycle while balancing cost, risk, and compliance. It mitigates business risk with compliant data archiving and retention by streamlining IT infrastructure and protecting the privacy rights of data with the SAP ILM.
- In SAP ILM, time-based data lifecycle management is performed with the help of rules that you define in the context of policies.
  - **Residence Period**: Data can only be archived after residence period is completed. Hence, data stays in HANA database until this period has expired.
  - **Retention Period:** Data cannot be destroyed/removed until this period has expired.



SAP IQ:

- SAP IQ is a column-based, petabyte scale, relational database software system; produced by Sybase Inc., now an SAP company.
- SAP IQ (Sybase IQ) is a high-performance decision support server designed specifically for data warehousing. This cross-platform product runs on several popular Unix, Linux, and Windows platforms.

Key Points for Archiving:

- We cannot change archived data.
- SAP IQ is secondary database which acts as a cold storage and provides read access to archived data on need-basis; Data stored in IQ is compressed.
- SAP does not recommend using same IQ instance for multiple SAP environments like SAP S4HANA & SAP BW4HANA (owing to backups & performance concerns)
- Recovery of archived data back into respective SAP database tables is possible using reload programs but it should be used in case of emergency only as it may cause inconsistency if there is change in configuration.
- SAP has delivered out-of-the box functionality to read the archived data for critical transactions. In case there is no choice, then custom enhancement will be required to enable it.

4. Conclusion

Data Archival and Partitioning of large tables can significantly reduce the issues caused by high number of records in a database tables.

References

[1] Roland Kramer article- "SAP ILM with SAP IQ Database" published on SAP Community
[2] SAP ILM Overview on sap.com/products
[3] SAP Notes related to SAP HANA CPU, Parameters & Delta Merge operations from SAP Service Portal Knowledge base - Note 2100040, Note 2186744, Note 2057046.
[4] Table Partitioning, SAP HANA Docs, SAP Help at: help.sap.com/docs/hana-cloud-database