International Journal of Cloud Computing (IJCC)
Volume 1, Issue 1, January-December 2023, pp. 33-52, Article ID: IJCC_01_01_004
Available online at https://iaeme.com/Home/issue/IJCC?Volume=1&Issue=1
Impact Factor (2023): 2.10 (Based on Google Scholar Citation)
Journal ID: 2563-4512

MASTERING BIG DATA WITH SAP HANA: CUTTING-EDGE STRATEGIES FOR SCALABLE AND EFFICIENT DATA MANAGEMENT IN THE CLOUD TECHNIQUES

Puneet Aggarwal[*],
[*]SAP Technical lead, Deloitte LLP, Dallas, USA.

ABSTRACT

As data scale and complexity continue to grow, the effective management of large tables in SAP HANA has become essential for organizations seeking to maintain high performance and efficient operations. Large data sets can slow down requests, consume a large amount of memory and increase the complexity of tasks such as backup, recovery and data maintenance. This paper focuses on practical methods for managing large tables in SAP HANA, with special emphasis on partitioning techniques, data storage, and performance optimization. The document describes the different types of tables and partition options in SAP HANA, highlighting the challenges that large tables pose for system performance and memory usage. It then presents best practices for technical tables, methods for purging or storing data, and continuous monitoring methods to ensure that performance stays optimal over time. Based on Gartner's market research for 2023, the study highlights the importance of tailored data management strategies that can adapt to business needs. Following structured partitioning and data management methods, organizations can significantly improve SAP HANA efficiency, reduce system maintenance and support future scalability. These strategies are essential for organizations that rely on SAP HANA to manage vast and growing data sets and ensure stability, speed and reliability.

Keywords: SAP HANA, Big Data Management, Data Partitioning, Cloud Ecosystems, Scalability, Data Lifecycle Optimization, Performance Tuning, Predictive Scaling, In-Memory Database, Data Storage Optimization, Table Partitioning Strategies, Native Storage Extension (NSE), Real-Time Analytics, Multi-Cloud Resource Orchestration, Enterprise Data Management

Cite this Article: Puneet Aggarwal. Mastering Big Data with SAP HANA: Cutting-Edge Strategies for Scalable and Efficient Data Management in the Cloud Techniques. *International Journal of Cloud Computing (IJCC)*, 1(1), 2023, pp. 33-52.

https://iaeme.com/Home/issue/IJCC?Volume=1&Issue=1

I. INTRODUCTION

In today's fast-evolving data landscape, organizations must manage ever-growing datasets effectively to maintain high performance and operational efficiency. SAP HANA, a leading in-memory database platform, enables real-time data processing and analytics, making it invaluable for businesses. However, as tables increase in size, they can introduce challenges in terms of memory usage, query performance, and storage management. Addressing these challenges requires a multifaceted approach that includes archiving, purging, and strategic partitioning.

SAP HANA supports two main types of tables: column store tables and row store tables. Column store tables are optimized for analytical queries and high-performance read operations, making them well-suited for large datasets, while row store tables are generally used for transactional data and quick access to individual records. SAP HANA imposes a critical limitation: each partition can hold up to 2 billion rows. For very large tables, this limitation necessitates efficient partitioning strategies to avoid performance degradation and to maintain data integrity.

This paper explores best practices for managing large tables in SAP HANA, including strategies for archiving, purging, and leveraging SAP HANA's Native Storage Extension (NSE) to manage hot and warm data. Additionally, the study provides guidance on when and how to apply partitioning to meet the 2-billion-row limit effectively. Insights from Gartner's 2023 market research support the importance of these strategies in today's data-driven environments, where efficient data management is essential for both performance and scalability.

II. LITERATURE REVIEW

The exponential growth of big data, projected to rise from $42 billion in 2018 to $103 billion by 2027 with an 11% CAGR, underscores the urgent need for scalable data management systems. Organizations face challenges such as performance degradation, rising storage costs, and complex data lifecycle management as datasets grow.

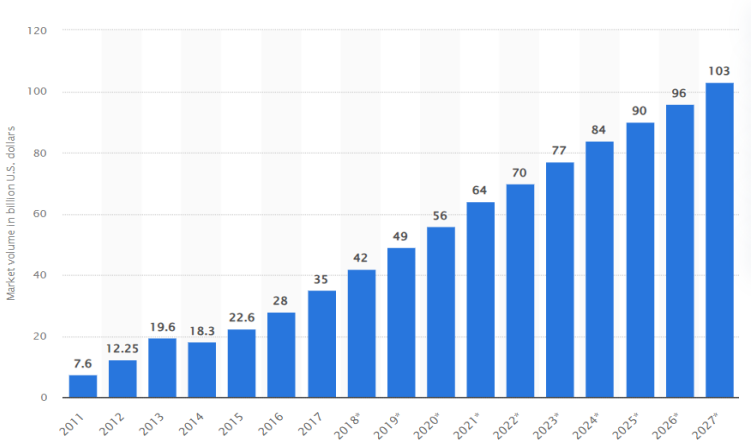

Fig 1 - **Big data market size revenue forecast worldwide from 2011 to 2027**

Advanced techniques like partitioning, predictive scaling, and hybrid storage solutions are essential for managing these challenges. SAP HANA's features, such as in-memory processing, Native Storage Extension (NSE), and efficient partitioning strategies, address scalability and performance needs while optimizing memory and storage usage.

Cloud-based environments further complicate data management, demanding dynamic resource orchestration and multi-cloud strategies. By integrating AI, blockchain, and predictive analytics, organizations can unlock real-time insights and innovative use cases, enabling efficient and sustainable data operations.

Efficient data management is no longer optional; it is the cornerstone of modern, scalable enterprise systems, driving innovation while addressing economic and environmental concerns.

III. CHALLENGES IN MANAGING LARGE TABLES IN SAP HANA

SAP HANA's in-memory architecture provides exceptional processing speed, making it ideal for real-time data access. However, as tables increase in size, several challenges arise, affecting performance, storage efficiency, and long-term scalability. Below are some of the main challenges that organizations encounter when managing large tables in SAP HANA:

A. Performance Degradation

Large tables can significantly impact query performance. As data grows, queries have to scan more data, increasing read times and memory usage. SAP HANA's column store tables, which are optimized for analytical queries, handle large datasets efficiently but are still impacted by excessive row counts.

Without strategies like partitioning and efficient data storage, the need to scan the entire table can create bottlenecks, particularly for high-demand queries.

B. Memory and Storage Constraints

As an in-memory database, SAP HANA uses RAM to store and process data. Large tables quickly consume memory, limiting the resources available for other processes. This memory demand often leads to additional hardware investments if large tables aren't managed efficiently through archiving, purging, or partitioning.

Using Native Storage Extension (NSE) to separate hot and warm data is a useful approach. Frequently accessed ("hot") data is kept in-memory, while less-used ("warm") data is stored on disk, helping to reduce memory usage without sacrificing access speed for critical data.

C. Complexity in Backup and Recovery

As tables grow, backup and recovery processes become more time-consuming, requiring more storage space for backups and longer recovery times in case of failures. Large tables can extend downtime during recovery, impacting business continuity.

Archiving historical data can reduce the size of active tables, streamlining backup and recovery tasks and improving efficiency. Purging can further help by removing outdated data, streamlining both backup and recovery processes.

D. Data Purging and Archiving Requirements

Regular purging and archiving of data that is no longer needed for daily operations are essential to keep tables manageable. However, identifying which data to archive or purge without disrupting business processes requires careful planning.

Custom reports, python scripts and procedures may be needed to automate archiving and purging tasks, given SAP HANA's limited built-in archiving functionality. Implementing regular archiving routines can significantly reduce the volume of active data, allowing SAP HANA to operate more efficiently.

E. Managing Hot and Warm Data with Native Storage Extension (NSE)

SAP HANA's NSE enables the segregation of data based on usage frequency. By storing hot (frequently accessed) data in-memory and warm (less frequently accessed) data on disk, NSE provides a cost-effective way to manage data without sacrificing performance for critical queries.

Organizations can use NSE to prioritize data based on business needs, ensuring high-priority data remains in-memory while optimizing storage for less active records.

F. Partitioning Requirement & the 2 Billion Row limit

SAP HANA imposes a 2-billion-row limit per partition in column store tables. For extremely large tables, this limitation makes partitioning essential to avoid performance issues and potential data access errors.

Partitioning large tables can help distribute data across multiple segments, making them easier to manage and improving query performance. However, choosing the right partitioning strategy—whether range, hash, or multi-level—requires careful analysis of data characteristics and usage patterns.

IV. TYPES OF TABLES AND PARTITIONING TECHNIQUES IN SAP HANA

SAP HANA Table Types

SAP HANA supports two primary types of tables—**column** tables and **row** tables—each optimized for specific performance characteristics and use cases.

Aspect	**Column Tables**	**Row Tables**
Storage Method	Data is stored in contiguous memory locations organized by columns.	Data is stored as individual records, with each row containing all fields for a single entity.
Performance	Optimized for high-performance read operations, particularly suited for analytical workloads.	Performs better in scenarios with frequent single updates, suitable for transactional tasks.
Memory Usage	Reduces memory usage due to efficient columnar compression, ideal for large datasets.	Generally higher memory usage due to diverse row data, with limited compression potential.

Best Use Cases	Ideal for analytics, reporting, and applications requiring aggregate functions on specific columns.	Suitable for transactional data and scenarios requiring quick access to full records.
Example Scenarios	Large tables needing frequent aggregations, such as financial reports or sales analysis tables.	Small tables with transactional records, like order details or customer information.

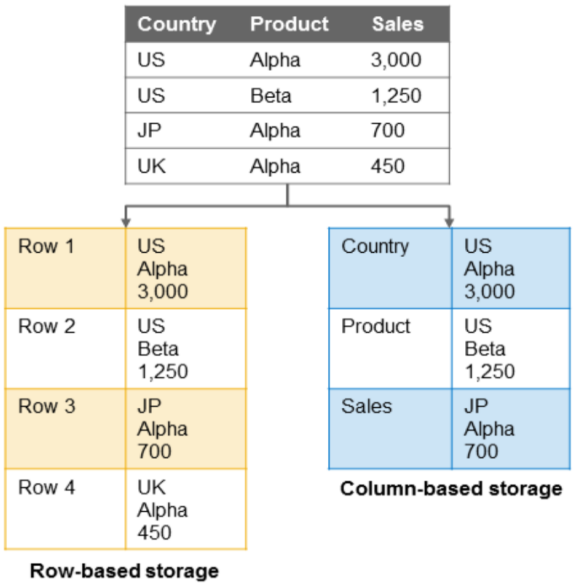

Fig 2 – **Row and Column Store tables in SAP HANA**

Partitioning Techniques in SAP HANA

Partitioning divides large tables into manageable segments, enhancing both performance and maintainability. SAP HANA offers two main partitioning techniques: **range partitioning** and **hash partitioning**, each effective for particular data structures and query requirements.

1) Range Partitioning

Range partitioning divides a table into partitions based on specified value ranges within a column, such as date ranges. For example, a sales table could be partitioned by year or month, with each partition containing data for a specific time period.

Applications: Range partitioning is effective for low cardinality columns where data naturally segments by range. It is ideal for time-based data like dates or limited categorical data, allowing efficient query performance.

Advantages: By isolating subsets of data, range partitioning allows SAP HANA to selectively load only relevant partitions, thereby optimizing query performance and reducing memory load when filtering based on these range columns.

2) Hash Partitioning

Hash partitioning uses a hash function to distribute data evenly across multiple partitions. This approach balances data among partitions, improving performance when multiple partitions need to be accessed concurrently.

Applications: Hash partitioning is effective for high cardinality columns, such as customer IDs or order numbers, where unique values are frequent. It ensures even data distribution and consistent performance across partitions.

Advantages: By distributing data evenly, hash partitioning reduces the likelihood of performance bottlenecks and is beneficial in scenarios that require consistent access speeds across partitions.

3) Choosing Between Range and Hash Partitioning

Selecting the right partitioning strategy is critical for optimizing SAP HANA performance. Generally, range partitioning is recommended for data with low cardinality, while hash partitioning is better suited for high cardinality data. Each technique helps SAP HANA manage large tables more effectively, lowering memory requirements, enhancing query performance, and offering scalability for growing datasets.

V. BEST PRACTICES FOR MANAGING TECHNICAL TABLES

SAP Note 2388483 provides guidance on managing technical tables in SAP HANA, focusing on tables that support communication, logging, tracing, administration, analysis, metadata, staging, and auditing functions. These practices are not typically applied to application tables, which require different management strategies. For a comprehensive data management approach, SAP recommends the Data Management Guide available in the Data Volume Management area, alongside other notes such as SAP Note 16083 (standard job scheduling and reorganization) and SAP Note 2190119 (S/4HANA job repository for technical tables).

Key Technical Tables to Monitor and Manage:

- CDPOS (Change Document Items)
- CDHDR (Change Document Headers)
- ACDOCA (Universal Journal Entry Line Items)
- BALDAT (Application Log Data)
- BALHDR (Application Log Header)

Due to their rapid growth from continuous logging and transactional updates, these tables require careful management practices, such as regular purging, archiving, and partitioning, to optimize system resources and prevent performance degradation.

VI. APPROACH FOR BEFORE PARTITIONING

1) Pre-Partitioning Analysis

Before implementing a partitioning strategy, it's crucial to conduct a thorough analysis of table characteristics to determine the most effective approach. Recommended pre-partitioning steps include:

Analyze the Largest Tables: Start by identifying the largest tables in the system to determine which ones may benefit most from partitioning. Tools like the SQL command HANA_Tables_LargestTables (ONLY_TECHNICAL_TABLES = 'X') (from SAP Note 1969700) or reports available in DVM (Data Volume Management) and ST14 can help identify high-growth tables. Use TAANA transaction for particular table analysis.

Check Table Types (Row or Column Store): SAP HANA only supports partitioning for column store tables. If a large table is in a row store format, consider converting it to a column store before partitioning.

Review Historical Data Growth: Analyze historical data growth trends to forecast future table sizes and ensure partitioning will adequately manage expected growth. Reviewing data from DVM or tools like ST14 can provide insights into growth rates and help set up a scalable partitioning structure.

2) Choosing the Right Partitioning Strategy

Selecting an appropriate partitioning strategy depends on data characteristics and query patterns. SAP HANA offers single-level and multi-level partitioning options, each suited to different scenarios:

Single-Level Partitioning

Hash Partitioning: Ideal for high cardinality columns, hash partitioning distributes data evenly across partitions, balancing data load and improving query performance for tables with many unique values.

Range Partitioning: Suitable for low cardinality columns, range partitioning divides data based on a set range of values, such as date ranges. It allows for efficient pruning by isolating specific data subsets for queries filtered by the range values.

Multi-Level Partitioning

Hash-Range Partitioning: Combines hash partitioning at one level (to balance load) with range partitioning at a second level (for efficient filtering), making it useful for tables where a single-level approach would lead to uneven data distribution.

Range-Hash Partitioning: This approach, which applies range partitioning first followed by hash, is less common but can be effective when a specific range-based structure is needed before further distributing data within those ranges.

Choosing the optimal partitioning strategy ensures the efficient handling of large datasets, improves query response times, and maximizes memory usage. By partitioning appropriately, SAP HANA can manage large technical tables effectively, supporting system scalability and performance.

VII. TABLE Placements RULES in HANA

Table placement is a critical aspect of managing partitions in SAP HANA, especially **in scale-out environments**. Proper placement ensures efficient data distribution, load balancing, and query performance. To achieve optimal results, groups of tables are often defined according to semantically-related tables, ensuring that related data is co-located and improving performance for queries and joins.

1) Key Considerations for Table Placement

1. **Dynamic Range Partitioning**:
   - Automatically creates new partitions when thresholds are reached.
   - Optimizes performance by balancing data distribution across worker nodes.
2. **Load Balancing Across Nodes**:
   - Evenly distribute table partitions across all worker nodes to prevent overloading a single node.
   - Ensure semantically-related tables or frequently joined tables are placed on the same node for optimal performance.
3. **Groups of Tables**:
   - Define groups of semantically-related tables to co-locate data for efficient joins and logical data organization.
4. **Avoid Overloading the Master Node**:
   - Minimize table placement on the master node to reserve its capacity for transactional tasks and overall coordination.
5. **Application-Specific Rules**:
   - Align table placement with the specific application needs (e.g., BW/4HANA or S/4HANA environments).

2) SQL Commands for Table Placement

- **View Current Table Placement Settings**

Check the current configuration for table placement, including dynamic range partitioning and max rows per partition:

```
SELECT SCHEMA_NAME, TABLE_NAME, GROUP_TYPE,
MIN_ROWS_FOR_PARTITIONING, INITIAL_PARTITIONS,
REPARTITIONING_THRESHOLDS, LOCATION,
DYNAMIC_RANGE_PARTITIONING
FROM "SYS"."TABLE_PLACEMENT";
```

- **Set Table Placement Rules**

Define placement rules, such as enabling dynamic range partitioning or setting maximum rows per partition:

```
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'system')
SET ('table_placement', 'dynamic_range_partitioning') = 'true' WITH RECONFIGURE;

ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'system')
SET ('table_placement', 'max_rows_per_partition') = '1500000000' WITH RECONFIGURE;
```

- **Move a Partition to Another Node**

Reassign a specific partition to a different node to balance the load:

```
ALTER TABLE "<schema_name>.<table_name>" MOVE PARTITION <partition_number> TO '<server_name>:<port_number>' PHYSICAL;
```

- **Enable Automatic Table Distribution (Scale-Out Scenarios)**

Optimize existing table distribution automatically in a scale-out landscape:

```
ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'SYSTEM')
SET ('table_redist', 'balance_by_execution_count') = 'true' WITH RECONFIGURE;
```

- **Dynamic Range Partitioning for Insert Tables**

Enable dynamic range partitioning for insert-heavy tables:

```
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'SYSTEM')
SET ('table_placement', 'dynamic_range_partitioning') = 'true' WITH RECONFIGURE;
```

- **Check Table Distribution or Placement Configuration**

Verify the applied table placement rules:

```
SELECT SCHEMA_NAME, GROUP_TYPE, MIN_ROWS_FOR_PARTITIONING, INITIAL_PARTITIONS, REPARTITIONING_THRESHOLDS, LOCATION, DYNAMIC_RANGE_PARTITIONING
FROM "SYS"."TABLE_PLACEMENT";
```

- **Analyze Table Placement Across Nodes**

Identify the distribution of table partitions across nodes for load balancing analysis:

```
SELECT PARTITION_ID, PARTITION_SPEC, HOST, PORT
FROM "SYS"."TABLE_PLACEMENT";
```

- **Reorganize Table Placement Plan**

Generate and execute a reorganization plan to redistribute tables:

```
-- Generate a reorganization plan
CALL REORG_GENERATE(6, 'GROUPNAME=><your_table_group_name>');

-- Check the generated plan
SELECT * FROM REORG_PLAN;

-- Execute the plan
CALL REORG_EXECUTE(?);

-- Monitor the execution status
SELECT * FROM REORG_STEPS WHERE REORG_ID = <value_returned_by_reorg_execute>;

https://help.sap.com/docs/SAP_HANA_PLATFORM/6b94445c94ae495c83a19646e7c3fd56/667bfd8f3adc47f8af4c753447e40693.html?version=2.0.05&locale=en-US
```

3) Best Practices for Table Placement

- **Define Table Groups:**

Group semantically-related tables together for co-location, improving performance for joins and logical operations.

- **Dynamic Range Partitioning**:

Use for tables with unpredictable growth patterns to automatically create new partitions when thresholds are exceeded.

- **Minimize Master Node Workload:**

Place partitions on worker nodes rather than the master node, reserving the master node for coordination tasks.

- **Regular Monitoring and Adjustment:**

Use TABLE_PLACEMENT and M_EFFECTIVE_TABLE_PLACEMENT system views to monitor placement and rebalance when necessary.

- **Application-Specific Configurations:**

Follow table placement rules for specific systems like BW/4HANA (SAP Note 2334091) or Business Suite on HANA.

By aligning table placement strategies with the system architecture and application needs, including grouping semantically-related tables, organizations can optimize SAP HANA's performance and scalability in both scale-out and scale-up environments.

VIII. APPROACH FOR PURGING AND ARCHIVING

A. Data Purging Approach

When managing large SAP HANA systems, especially in transactional (OLTP) environments like S/4HANA and analytical (OLAP) setups like BW/4HANA, purging unnecessary data is essential to maintain system performance and manage storage costs. This process involves identifying high-growth tables, assessing data retention needs, and implementing structured purging methods. Here's an overview of approaches and tools for effective data purging in SAP HANA environments.

1) Identifying High-Growth Tables

Effective purging starts with pinpointing tables that accumulate data rapidly. These tables often drive the largest data volumes and can impact performance over time if not managed. To identify these tables:

SQL Scripts for Table Analysis: SAP HANA provides several monitoring SQL scripts and views to track table growth and resource usage. For example:

- Use *HANA_Tables_LargestTables* to find tables with the largest memory footprints.
- Run SELECT * FROM m_cs_tables ORDER BY memory_size_in_total DESC to identify column store tables with the highest memory consumption.

DVM (Data Volume Management) Reports: SAP Data Volume Management reports available in SAP Solution Manager can highlight high-growth areas, providing insights into tables with rapid data accumulation and historical growth trends.

These insights help teams prioritize tables that require regular purging, often focusing on technical tables with audit logs, transactional data, or other high-frequency entries, like CDPOS/CDHDR

(change document items/headers) in S/4HANA and RS tables* (e.g., RSBKREQUEST) in BW/4HANA.

For Functional tables, make sure to get necessary business approvals and decide frequency and cadence for the same.

Once tables are identified, follow further techniques

2) SQL-Based Purging Techniques

SAP HANA supports SQL-based purging methods, allowing custom scripts to delete or archive data directly from large tables based on specific criteria, such as date or status. This method is flexible and can be tailored to suit particular data retention policies.

- **Simple SQL Delete Statements**: You can purge data older than a specific date with SQL statements, e.g.,

```
DELETE FROM <table_name> WHERE <date_column> < ADD_DAYS(NOW(), -180);
```

This approach helps remove data beyond a defined retention period, such as records older than six months.

- **Scheduled Jobs for Automated Purging**: SAP HANA allows for scheduling SQL scripts using **Job Scheduler** or **custom ABAP programs**. Regular jobs can automate SQL scripts to purge data periodically, ensuring tables stay within manageable sizes without manual intervention.Also you can save the SQL in DBACOCKPIT and schedule it from DB13 Planning calendar

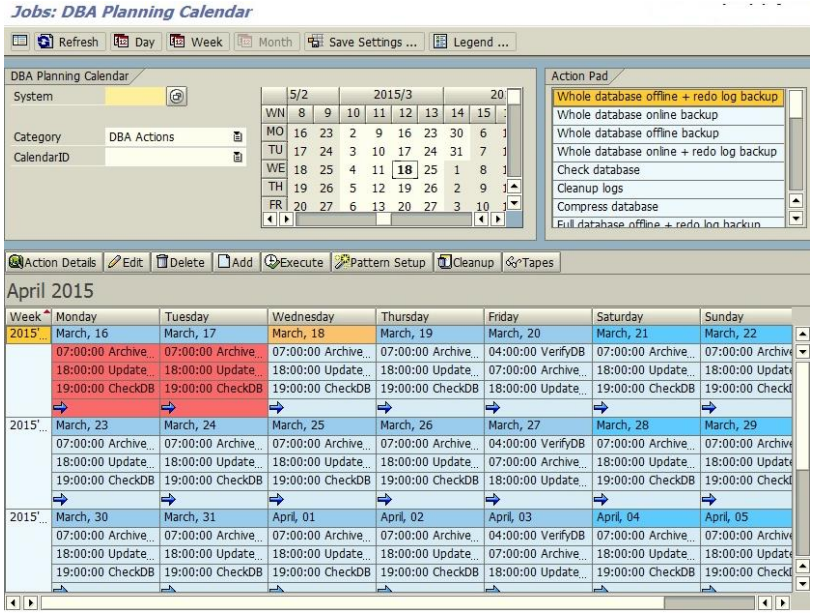

Fig 3. DB13 SAP schedules

Fig: Example of schedules in DB13

- **Scheduled Jobs for Automated Purging**: SAP HANA allows for scheduling SQL scripts using **Job Scheduler** or **custom ABAP programs**. Regular jobs can automate SQL scripts to purge data periodically, ensuring tables stay within manageable sizes without manual intervention.
- **Using Standard or Customized SAP Programs for Purging**
- For high-growth tables that SAP recognizes as key to system health, SAP offers **standardized cleanup programs**. These programs are designed to target specific tables and data types for regular purging:
- **SARA for Archiving**: The SARA transaction in SAP is a powerful tool for archiving data from high-growth tables such as **CDPOS, CDHDR, BALDAT, and BALHDR**. This tool allows data to be archived based on configured retention policies, making it accessible for audits while freeing up main storage.
- **SLG2 for Application Logs**: For managing large log tables like BALDAT (Application Log Data), SLG2 provides a streamlined way to delete or archive old logs based on retention requirements.
- **SAP Housekeeping Jobs**: SAP provides specific housekeeping jobs for frequently used tables, such as RSBKREQUEST in BW/4HANA, to delete obsolete entries automatically, preserving only essential data.

Data Archiving Approach

- Archiving historical data is crucial for keeping SAP HANA systems efficient and responsive. By moving older, less-used data to archive storage, you can reduce table sizes, improve system performance, and cut down on storage costs—all without disrupting daily business operations.
- **Time-Based Archiving**: Archive data older than a defined retention period, such as transactions or logs exceeding a year. Tools like SAP's **SARA (Archive Administration)** transaction allow businesses to set up and manage these archiving processes efficiently.
- **Event-Based Archiving**: Archive records tied to specific statuses or completed events, such as finalized orders, closed projects, or inactive customers. SAP's **Standard Archiving Objects** help streamline this process by grouping related data for archiving.
- **Selective Archiving**: Archive non-critical data like logs or metadata while retaining frequently accessed information in the active database. SAP's **SARI (Archive Information System)** ensures archived data can still be searched and retrieved when needed.

Key tools

- **SARA**: Configures, schedules, and executes archiving using predefined Standard Archiving Objects like FI_DOCUMNT, MM_EKKO, and SD_VBAK. (Standard ADK)
- **SARI**: Enables search and retrieval of archived data for audits and reporting. (Standard ADK)
- **Opentext Content Server**: Provides secure, cost-effective storage for archived data integrated with SAP. (license required)

- **SAP Information Lifecycle Management:** Manages data retention policies and automates lifecycle archiving with tiered storage options. (license required)
- **Data tiering Optimization:** Moves older, less-used BW data to warm or cold storage to optimize active memory use.(Standard BW4/HANA)

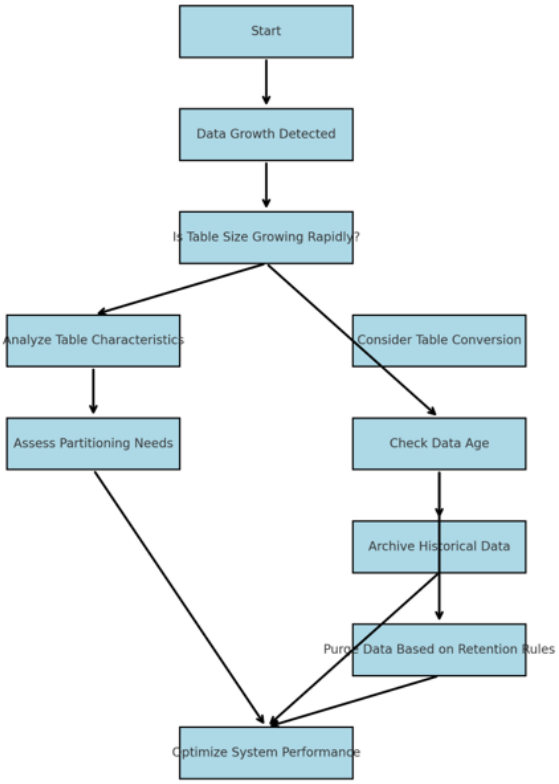

Fig 4 – Decision workflow for Data management

IX. PARTITIONING EXECUTION METHODOLOGY

1) Pre-Execution Analysis

- **Identify Large Tables**: Start with the top 50 largest tables by memory and CPU consumption using SQL scripts like *HANA_Tables_LargestTables and HANA_Resources_CPUAndMemory_History.*

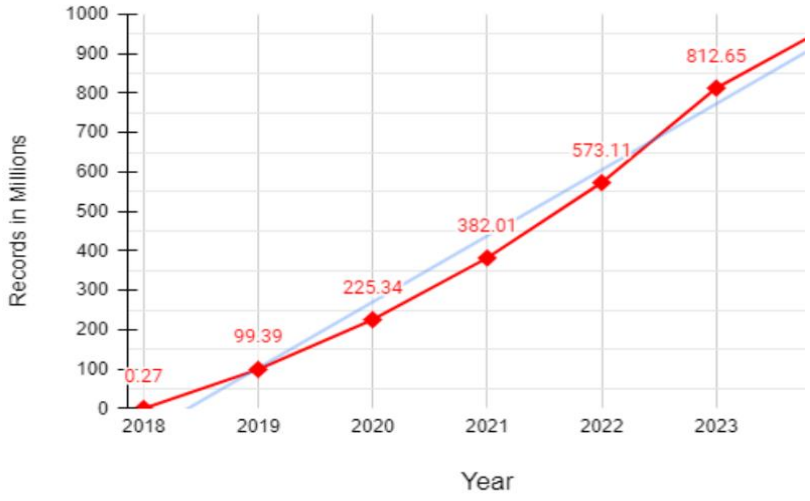

Fig 5 Record count Vs YoY growth chart

- **Check Table Type**: Determine if the table is stored in the row store or column store. Convert to column store if necessary, as partitioning applies only to column store tables.
- **Analyze Growth Trends**: Review table growth history using tools like DVM or ST14. Use Year-over-Year (YoY) growth charts to evaluate trends.
- **Assess Purging/Archiving Options**: Identify opportunities for data cleanup or archiving before partitioning. If purging is not possible, proceed to partitioning.
- **Analyze Growth Sources**: Use SQL scripts to identify growing programs or data

SELECT progname, COUNT(*) FROM dbtablog GROUP BY progname ORDER BY 2 DESC;

Further refine growth analysis with:

SELECT username, tabname, COUNT(*) FROM dbtablog WHERE progname='<Programname>' AND logdate <= ADD_DAYS(NOW(), -60) GROUP BY username, tabname ORDER BY 3 DESC;

2) Execution Planning

Table Structure Analysis

Identify primary keys, indices, and query patterns. Use the following query to analyze frequently used SQL statements:

```
SELECT a.*, SUBSTR_AFTER(a.statement, 'WHERE ') AS filters FROM ( SELECT statement_hash, TO_VARCHAR(statement_string) AS statement, MAX(start_time) AS latest_runtime, COUNT(*) AS count, SUM(cpu_time) FROM m_expensive_statements WHERE TO_VARCHAR(statement_string) LIKE '%<REPLACE_TABLE_NAME>%' GROUP BY statement_hash, TO_VARCHAR(statement_string) ) a ORDER BY 4 DESC;
```

Optimize Partitioning:

Optimizing partitioning in SAP HANA involves fine-tuning specific parameters related to partitioning, persistence, and execution. Adjusting these settings based on testing in lower environments, as well as considering factors like table size and record count, can enhance performance during partitioning operations.

Partitioning Parameters:

- **split_threads**: Determines the number of parallel workers for split/merge operations. The default value is 0, which implies 80% of maximum concurrency. Adjusting this parameter can expedite partitioning tasks.
- **bulk_load_threads**: Controls the number of threads used during bulk load operations. Increasing this value can improve data loading performance but may also increase resource consumption.

Refer: https://me.sap.com/notes/2044468/E

Persistence Parameters:

- **savepoint_interval_s**: Specifies the interval (in seconds) between automatic savepoints. The default is 300 seconds (5 minutes). Adjusting this interval can balance between data safety and system performance.
- **logreplay_savepoint_interval_s**: Defines the interval for log replay savepoints. Modifying this setting can influence recovery times and system performance.
- **log_buffer_size_kb**: Sets the size of the log buffer in kilobytes. A larger buffer can reduce the frequency of disk writes but may consume more memory.

Execution Parameter:

- **default_statement_concurrency_limit**: Limits the maximum parallelism for a single database request per SAP HANA node. Adjusting this parameter can help manage system load during intensive operations.

By carefully tuning these parameters, you can optimize partitioning operations in SAP HANA, leading to improved performance and resource utilization.

Online vs. Offline

- **Online Partitioning**: Suitable for environments where latency can be tolerated with less DML operations.

- **Offline Partitioning**: Preferred when DML operations cannot be blocked.

3) Partitioning Execution

- **Disable Auto Merge**:

ALTER TABLE <table_name> DISABLE AUTOMERGE;

- **Execute Partitioning Commands**:

For **Hash Partitioning**:

ALTER TABLE <table_name> PARTITION BY HASH (<column>) PARTITIONS <number>;

For **Range Partitioning**:

```
ALTER TABLE <table_name> PARTITION BY RANGE (<column>) (
  PARTITION '2021' <= VALUES < '2022',
  PARTITION OTHERS
);
```

For **Multi-Level Partitioning**:

ALTER TABLE <table_name> PARTITION BY HASH (<col1>) PARTITIONS <num>, RANGE (<col2>);

Monitor Execution: Use DBACOCKPIT or SQL monitoring tools like HANA Studio or hdbsql to track system performance during partitioning.

4) Post-Execution Steps

- **Re-enable Auto Merge**:

ALTER TABLE <table_name> ENABLE AUTOMERGE;

- **Validate Partitioning**: Check partitioning status using views like Monitoring views PARTITIONED_TABLES, TABLE_PARTITIONS

SQL:"HANA_Tables_ColumnStore_PartitionedTables" (SAP Note 1969700) to list partitioned tables

SQL:"HANA_Tables_ColumnStore_Partitions" (SAP Note 1969700) to list individual partitions of one or multiple tables

SQL:"HANA_Tables_ColumnStore_TableHostMapping" (SAP Note 1969700) to show the partition distribution of tables across nodes in a scale-out scenario

- **Reset Parameters**: Restore modified configuration settings to their original values:

5) Monitoring and Maintenance

- Schedule periodic checks on table growth and partition distribution.
- Optimize queries using partition pruning to load only relevant data.

This methodology ensures efficient execution, minimal disruption, and sustained performance improvements in SAP HANA systems.

X. BENEFITS OF PARTITIONING LARGE TABLES

Partitioning large tables in SAP HANA offers several significant benefits that enhance database performance and manageability:

1) **Managing Large Tables:**

- **Row Limit Compliance:** SAP HANA column-store tables have a maximum limit of 2 billion records per table or partition. Partitioning tables nearing this limit ensures compliance and prevents potential issues.
- **SID Tables in BW Environments:** Typically, SID tables (e.g., those following the naming convention /B%/S%) should not be partitioned because they inherently cannot exceed the 2 billion record limit due to BW SID constraints. Additionally, partitioning these tables can introduce overhead due to unique index checks.

2) **Optimizing Memory Usage:**

- **Delta Merge Operations:** Large tables can lead to increased memory consumption during delta merge operations. Partitioning these tables into smaller segments reduces memory requirements and enhances performance during such operations.
- **Avoiding Page Faults:** To prevent overhead from page faults and excessive CPU consumption, it's advisable to limit records per table or partition to less than 1.07 billion. This practice helps in managing OLAP bitmaps efficiently.

3) **Enhancing Query Performance:**

- **Parallel Processing:** In scale-out scenarios, distributing table partitions across multiple hosts allows complex queries to be processed in parallel, reducing execution time.
- **Partition Pruning:** Partitioning enables the database to access only relevant partitions during queries, improving performance by avoiding full table scans.

4) **Managing Data Temperature:**

- **Hot and Warm Data Segmentation:** Partitioning allows separation of frequently accessed (hot) data from less accessed (warm) data. This segmentation facilitates efficient data management and retrieval.
- **Native Storage Extension (NSE):** Features like NSE enable certain partitions to remain on disk while keeping hot data in memory, optimizing resource utilization.

5) **Improving Table Optimization Processes:**

- **Targeted Operations:** Partitioning allows table optimizations, such as delta merges and compression, to be performed at the partition level. This granularity reduces data volume and runtime for these operations.
- **Reduced Locking:** Smaller partitions mean quicker optimizations, leading to less locking of concurrent activities and earlier reduction of delta storage overhead.

6) **NUMA Optimization:**

- **Balanced Workload:** In Non-Uniform Memory Access (NUMA) architectures, partitioning tables across multiple NUMA nodes distributes the workload evenly, preventing overload on specific nodes and enhancing performance.

7) **Enhances Log Replay Performance:**

- **Parallel Log Replay:** Partitioning tables that undergo frequent changes allows log replay operations to be parallelized, improving performance during system replication, restarts, or recovery scenarios.

8) Reducing Delta Storage Contention:

- **Minimized Contention:** Partitioning reduces contention on delta storage, leading to improved performance during data modifications and reducing issues like delta append rollovers.

Implementing appropriate partitioning strategies in SAP HANA is crucial for maintaining system performance, managing large datasets, and optimizing resource utilization.

XI. CONCLUSION

As organizations grapple with the relentless growth of data, partitioning in SAP HANA emerges as a vital solution for maintaining system efficiency and scalability. Partitioning strategies not only adhere to technical limits like the 2-billion-row threshold but also enable robust data management that supports performance-critical operations in on-premises, cloud, or hybrid environments.

The ability to optimize memory usage, accelerate query execution through partition pruning and parallelism, and effectively manage hot and warm data ensures that SAP HANA systems can seamlessly adapt to the increasing complexity and volume of enterprise data. These capabilities are particularly critical in a data-driven world where operational efficiency and real-time insights are paramount.

Moreover, partitioning plays a key role in addressing challenges such as memory overhead during delta merges, NUMA optimization, and reducing contention in delta storage, thereby supporting the broader goal of sustainable and future-proof data management. As data growth becomes unavoidable across industries, leveraging advanced partitioning techniques empowers organizations to not just handle big data but to extract its full potential, driving innovation and maintaining a competitive edge.

REFERENCES

[1] Statista, "Big Data Market Size Revenue Forecast Worldwide from 2011 to 2027." Available at: https://www.statista.com/statistics/254266/global-big-data-market-forecast/

[2] SAP Training, "HA 200: SAP HANA Installing and Administering," SAP Training Materials. Available at: https://training.sap.com/course/ha200-sap-hana-installing-and-administering-classroom-019-g-en/

[3] SAP Help Portal, "SAP HANA Platform Documentation," SAP Help. Available at: https://help.sap.com/docs/SAP_HANA_PLATFORM

[4] SAP Help Portal, "Table Partitioning," SAP HANA Docs. Available at: https://help.sap.com/docs/hana-cloud-database/sap-hana-cloud-sap-hana-database-administration-guide/table-partitioning

[5] SAP Knowledge Base, "SAP Note 2044468 - FAQ: SAP HANA Partitioning." Available at: https://userapps.support.sap.com/sap/support/knowledge/en/2044468

[6] SAP Community Blogs, "Data Management Concepts and Techniques in SAP HANA Cloud." Available at: https://community.sap.com/t5/technology-blogs-by-sap/data-management-concepts-and-techniques-in-sap-hana-cloud/ba-p/13523597

[7] Emergen Research, "Big Data Market: Industry Trends and Forecast." Available at: https://www.emergenresearch.com/industry-report/big-data-market

[8] Fortune Business Insights, "Big Data Technology Market Analysis." Available at: https://www.fortunebusinessinsights.com/industry-reports/big-data-technology-market-100144

[9] https://help.sap.com/docs/SAP_HANA_PLATFORM/6b94445c94ae495c83a19646e7c3fd56/0bee129c18e7437c9e4aa38b9fb7a757.html?version=2.0.05&locale=en-US&q=TABLE_PLACEMENT+view
[10] SAP Note 2044468: "FAQ: SAP HANA Partitioning"
[11] SAP Note 2334091: "Partitioning Recommendations for BW/4HANA"
[12] SAP Note 1908075: "SAP BW powered by SAP HANA: Partitioning"
[13] SAP Note 1899817: "Table Partitioning for Business Suite on SAP HANA"
[14] SAP Note 2081591: "FAQ: SAP HANA Table Distribution"
[15] SAP Note 2418299: "Partitioning Best Practices for SAP HANA"
[16] SAP Note 2289491: "Table Partitioning in Finance (S/4HANA)"
[17] SAP Note 1969700: "HANA_Tables_TopGrowingTables_Records_History" Script
[18] SAP Note 2019973: "Repartitioning Plan and Execution in SAP HANA"
[19] SAP Note 2100010: "Delta Merge Performance Guidelines"
[20] SAP Note 1958216: "Automatic Table Distribution and Dynamic Range Partitioning"
[21] 2890332: "Performance Tuning for Partitioning in SAP HANA"
[22] 2874176: "Optimizing Parameters for Partitioning"
[23] SAP Note 2569097: "Table Group Type and Consistency Check for BW/4HANA"
[24] SAP Course: HA200 - Installing and Administering SAP HANA.

Citation: Puneet Aggarwal. Mastering Big Data with SAP HANA: Cutting-Edge Strategies for Scalable and Efficient Data Management in the Cloud Techniques. International Journal of Cloud Computing (IJCC), 1(1), 2023, pp. 33-52.

Article Link:
https://iaeme.com/MasterAdmin/Journal_uploads/IJCC/VOLUME_1_ISSUE_1/IJCC_01_01_004.pdf

Abstract Link: https://iaeme.com/Home/article_id/IJCC_01_01_004

✉ editor@iaeme.com