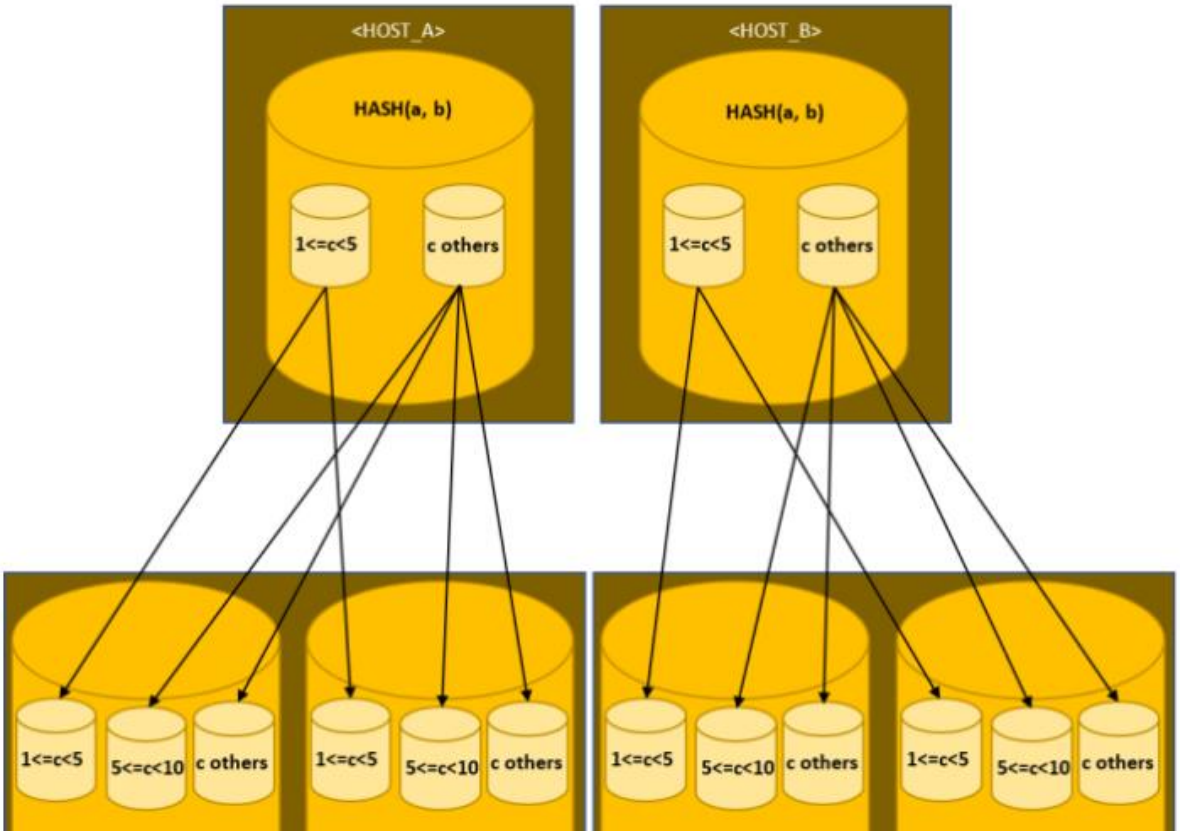
SAP HANA Partitioning

What does partitioning in SAP HANA environments mean? Partitioning means that tables are split into sub-tables, the so-called partitions, based on defined partitioning criteria



SAP HANA Database Partitioning by Irshad Rather

SAP HANA Database Partitioning by Irshad Rather

Resolution

1. Where do I find detailed information about partitioning?

General information related to partitioning is contained in the SAP HANA Administration Guide.

2. Which indications exist for issues related to SAP HANA partitioning?

The following SAP HANA alerts indicate problems in the memory area:

Alert	Name	Description
17	Record count of non-partitioned column-store tables	Determines the number of records in non-partitioned column-store tables. Current table size is not critical. Partitioning need only be considered if tables are expected to grow rapidly (a non-partitioned table cannot contain more than 2,147,483,648 (2 billion) rows).
20	Table growth of non-partitioned column-store tables	Determines the growth rate of non-partitioned column-store tables.
27	Record count of column-store table partitions	Determines the number of records in the partitions of column-store tables. A table partition cannot contain more than 2,147,483,648 (2 billion) rows.

SQL: "HANA_Configuration_MiniChecks" (SAP Notes 1969700, 1999993) returns a potentially critical issue (C = 'X') for one of the following individual checks:

Check ID	Details
M0510	Tables with > 100 partitions
M0512	Hash partitioning on multiple columns
M0513	Tables with many empty partitions
M0515	Partitioned tables with inverted hash indexes
M0519	Tables with large overflow partition
M0520	Tables / partitions > 1.5 billion rows
M0521	Table histories > 1.5 billion rows

M0525	Tables / partitions with large memory size
M0526	Tables / partitions with large memory share
M2020	Partitioned SID tables
M2025	Partitioned special BW tables < 1.5 bill. rows

SQL: "HANA_TraceFiles_MiniChecks" (SAP Note 2380176) reports one of the following check IDs:

Check ID	Details
T1100	Inadequate BW partitioning
T2020	Maximum number of rows per partition reached

SQL: "HANA_Threads_Callstacks_MiniChecks" (SAP Notes 1969700, 2313619) reports one of the following check IDs:

Check ID	Details
C1200	Record move between partitions

3. How can I check and evaluate the current partitioning in a system?

Partitioning information can be retrieved in the following ways:

- Monitoring view M_CS_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

4. What does partitioning in SAP HANA environments mean?

Partitioning means that tables are split into sub-tables, the so-called partitions, based on defined partitioning criteria.

5. Is partitioning available for both row store and column store?

Partitioning is only available for tables located in the column store. The row store doesn't support partitioning.

6. Is partitioning transparent for the application?

Partitioning is transparent for the application in a way that applications work properly with all partitioning strategies.

Nevertheless partitioning can have an impact on performance, so it can make a difference for the end user and the system load - both in a positive and negative way. In order to minimize the risk of performance regressions it is important to implement a good partitioning strategy.

Moreover, applications may implement special handling for partitions, for example they may add range partitions for upcoming time periods.

7. For what reasons is partitioning required or useful?

SAP HANA Database Partitioning by Irshad Rather

In general partitioning is most useful for large tables. In the following situations you can take advantage of partitioning:

Scenario	Details
Tables with many records	Each table and partition must not contain more than 2 billion records in the column store. Tables with a risk to reach the 2 billion record limit mid-term should be partitioned. Be aware that SID tables in BW environments (typically following the naming convention /B%/S% should normally not be partitioned because: They can never exceed the 2 billion records limit due to BW SID limitations (see SAP Note 1331403). They typically have 2 unique indexes and so changes would require expensive remote partition checks for uniqueness. Also other BW attribute tables (typically following the naming convention /B%/H%, /B%/I%, /B%/J%, /B%/K%, /B%/Q%, /B%/X% and /B%/Y%) should only be considered in special cases, e.g. if the number of records exceeds 1.5 billion and there is a risk to hit the technical limit of 2 billion rows. Mini checks M0520 ("Tables / partitions > 1.5 billion rows"), M0521 ("Table histories > 1.5 billion rows") and M0522 ("Tables / partitions > 1.5 billion UDIV rows") report tables with a particularly high amount of rows (SAP Note 1999993).
Tables with large memory footprint	Large table / partition sizes in column store are mainly critical with respect to table optimizations like delta merges and optimize compressions (SAP Notes 2057046, 2112604): Memory requirements are doubled at the time of the table optimization. There is an increased risk of locking issues during table optimization. The CPU consumption can be significant, particularly during optimize compression runs. The I/O write load for savepoints is significant and can lead to trouble like a long critical phase (SAP Note 2100009) Therefore you should avoid using particularly large tables and partition and consider a more granular partitioning instead. A reasonable size threshold is typically 50 GB, so it can be useful to use a more granular partitioning in case this limit is exceeded. Mini check M0525 ("Tables / partitions with large memory size") reports tables and partitions with a significant memory size (SAP Note 1999993).
Complex queries	In scale-out scenarios you can distribute the load of complex requests across different nodes if you locate the table partitions on different hosts. Be aware that this kind of approach can have a negative impact on "simple" queries that also require to access multiple hosts. Complex queries can also take advantage of parallelism. By processing data in all partitions concurrently the runtime can be reduced significantly.
Tables with hot and warm data	If tables contain frequently accessed (hot) areas and other areas with less frequent accessed (warm) these ranges can be separated by partitioning. The possibility to satisfy a query with accessing only a subset of partitions is also called partition pruning. If features like Native Storage Extension (NSE, SAP Note 2799997), data aging (SAP Note 2416490), time selection or paged attributes are used, it is additionally possible to make sure that only a subset of partitions is loaded into memory while partitions containing no hot data remain on disk.
Optimization of table optimizations	Table optimizations like delta merge (SAP Note 2057046) or optimize compression runs (SAP Note 2112604) can be improved by proper partitioning. Positive effects can be: Creating more partitions reduced the table optimization data volume and runtime because table optimizations happen on partition level and not globally. Quicker optimizations can have various benefits (e.g. less locking of concurrent activities and earlier reduction of delta storage overhead). If tables can be partitioned so that changes happen only to a subset of partitions, table optimizations are performed more targeted and unchanged partitions don't need to be touched.
NUMA optimization	Expensive queries related to specific large tables can result in an overload on certain NUMA nodes and effects like the following are possible: Performance regression due to increased number of queued JobWorkers that need to run on the overloaded NUMA node

	Performance regression and increased CPU consumption due to execution of expensive queries on remote NUMA nodes By creating more partitions the table is typically split across several NUMA nodes and so the risk of local overloads is reduced. Starting with SAP HANA 2.0 SPS 03 you can explicitly assign a partition to a NUMA node using the NUMA NODE clause. See SAP Note 2470289 for more information related to non-nuniform memory access (NUMA) in SAP HANA environments.
Log replay performance	Log replay can happen in different contexts, e.g.: System replication (SAP Note 1999880) with logreplay / logreplay_readaccess operation mode SAP HANA restart (SAP Note 2222217) SAP HANA recovery System replication takeover Log replay of one table (partition) is performed sequentially without the possibility to parallelize. So if you have tables that are very frequently changed and so a lot of redo information is generated, these tables can be the deciding factor for log replay performance and it can also be responsible for a system replication log replay backlog. Setting up more and smaller partitions for a heavily modified table can improve the performance and reduce the backlog of log replay operations.
Delta storage contention	Contention on delta storage (SAP Note 1999998, Sleeping / Sleep Semaphore / BTree GuardContainer) can be reduced by setting up more partitions. The same applies to delta append rollovers (SAP Note 1999998, DeltaDataObjectAppendRollover).
Explicit partition handling	In some cases it can be useful that the application controls the creation and existence of partitions based on specific criteria.

8. What kind of problems can be introduced with partitioning?

The following potential problems should be considered before implementing partitioning. The actual impact depends on the implementation, so with a good preparation you can minimize overhead and risks.

Problem	SAP Note	Details
Memory overhead	1999997	The collection and merging of data from different partitions can increase the memory requirements of SQL statements.
Transactional deadlocks	1999998	Bulk modifications on partitioned tables don't guarantee the specified order of the records and so there is an increased risk of transactional deadlocks ("SQL error 133: transaction rolled back by detected deadlock").
Increased CPU consumption Performance regressions	2000002 2100040	For various reason the CPU consumption can increase. See SAP Note 2000002 ("What are typical approaches to tune expensive SQL statements?" -> "Execution time higher than expected, negative impact by existing partitioning") for more information.
Increased scale-out network traffic	2222200	Partitioning typically increases the inter-node network traffic in scale-out environments.
Risk of inconsistencies in case of inverted hash indexes	2436619	Partitioning in context of inverted hash indexes (SAP Note 2109355) can result in inconsistencies with certain SAP HANA Revision levels.
Increased dictionaries and translation tables	1998599	Joining partitioned tables can result in a high amount of translation tables because in the worst case every partition of the first table has to be joined with every partition of the second table and each join is based on a individual translation table. So more partitions can result in more translation tables. If partitioning distributes identical column values of columns with a rather high

		amount of distinct values to different partitions, these values need to be included into several (partition-level dictionaries) and so the overall dictionary size increases. As a consequence also translation table sizes increase when the column is joined.

9. Which partitioning types exist?

The following partitioning types exist in SAP HANA environments:

Partitioning type	Details	Advantages	Disadvantages / Restrictions
HASH	Records are assigned to partitions based on a hash algorithm on the partition key columns	Easy setup No maintenance required Rather even row distribution if partition keys are selective without frequent values	All partitions need to be scanned unless all partition key columns are specified with "=" or "IN" conditions in the WHERE clause. Need to be based on primary key columns (if primary key exists, for first-level partitioning, homogeneous partitioning) No logical separation of data (e.g. hot vs. cold) possible
ROUNDROBIN	Records are assigned in a round-robin manner to the available partitions, partition key columns are not required	Easy setup No maintenance required Even row distribution	All partitions need to be scanned, no partition pruning possible No logical separation of data (e.g. hot vs. cold) possible No primary key allowed (because overhead of constraint evaluation would be significant)
RANGE	Records are assigned to partitions using defined (non-overlapping) partition ranges	Allows application related separation of data into different partitions (e.g. hot vs. cold) Consequently the best way to take advantage of partition pruning and delta merge optimization	Setup of optimal ranges requires application knowledge Regular maintenance required (e.g. definition of new partitions for upcoming year in case of time-based ranges, deletion of old partitions after archiving) Potentially uneven row distribution Need to be based on primary key columns (if primary key exists, first-level partitioning, homogeneous partitioning)

10. Is LIST partitioning available?

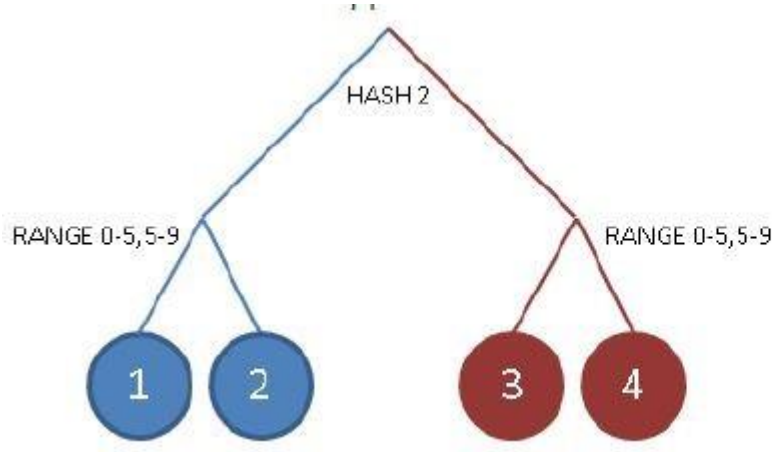
LIST partitioning allows to assign single or multiple values to a single partition. As part of RANGE partitioning SAP HANA allows to assign a single value to a single partition, but it is not possible to assign multiple values to a single partition. A dedicated LIST partitioning option is not available.

11. What is single-level and multi-level partitioning?

If a table is partitioned exclusively by one of the above partitioning approaches (HASH, ROUNDROBIN, RANGE), it is called single-level partitioning.

If each of the partitions itself is partitioned again by other criteria, we call it multi-level partitioning. In the following picture you can see a multi-level partitioning with a HASH partitioning on level 1 and a RANGE partitioning on level 2:

SAP HANA Database Partitioning by Irshad Rather




```
PARTITION BY
  HASH (<column>) PARTITIONS 2,
  RANGE (<column>)
    (PARTITION 0 <= VALUES < 5,
    (PARTITION 5 <= VALUES < 9)
```

The total number of partitions is the product of the number of partitions on every level.

The partition key columns of the second level can be chosen independent of the primary key of the table in case of HASH / HASH and RANGE / HASH partitioning.

The partitions on the second level form so-called partition groups (in the example above partitions 1 and 2 are one partition group, partitions 3 and 4 are another partition group). All members of a partition group are always located on the same node and can only be moved as a unit.

12. Are locks involved when a table is partitioned?

With SAP HANA 1.0 the standard table (re-)partitioning sets an exclusive object lock (see SAP Note 1999998), so all modifying DML operations are blocked while the partitioning is performed. SELECTs are still possible without any restriction.

Starting with SAP HANA 2.0 SPS 01 table (re-)partitioning is an online operation that doesn't permanently set the exclusive object lock. A lock is only required at the end of the partitioning operation for delta storage synchronization. To minimize locking times and other issues you should consider the following aspects:

- Avoid large delta storages before starting the repartitioning (e.g. by manually executing a delta merge, SAP Note 2057046).
- Execute the repartitioning at a time when there is limited modification load on the underlying table. Otherwise the delta storage can grow significantly during the repartitioning and the final locking phase can increase.
- SAP Note 2873607 describes potential problems with SAP HANA <= 2.00.048.01 when table replication is used (SAP Note 2340450) or repartitioning is started concurrently for different tables.

13. Which best practices exist for partitioning tables?

The following general best practices exist for partitioning tables:

Rule	Details
As few partitioned tables as possible	Only partition tables if you see a clear benefit without significant regressions.
As few partitions as possible	An unnecessary high amount of partitions results in overhead because some queries may have to access all partitions to find the data: A high amount of network channels are opened and so the system is at risk to reach the maxchannels limitation (SAP Note 2222200) and run into network related terminations. Certain operations like the determination of column statistics ("getColumnStat", SAP Note 2114710) have to be

	performed individually for each partition. So consider the following general rules before defining a certain number of partitions: If you partition tables due to the 2 billion limit it is usually acceptable if individual partitions contain up to 1.5 billion records (less if you expect a significant future growth). If you partition by date, you should avoid using granular ranges (e.g. days or weeks) resulting in a high amount of partitions. If you use range partition on columns with data that is not evenly distributed (e.g. number range column with multiple different number ranges), you should check the actual value distribution and define the range limits accordingly.
As few empty partitions as possible	The existence of many empty partitions can confuse the SAP HANA optimizer. See check ID M0513 of the SAP HANA Mini Checks (SAP Note 1999993) and make sure that you don't have a dominating amount of empty partitions. For example, don't perform time based range partitioning for ten years in the future and instead extend the partitions on a regular basis for the subsequent years.
As few partition key columns as possible	It is often useful to keep the number of partition key columns at a minimum extent because: Partition pruning of HASH partitions can only be used if all underlying partitioning columns are specified with "=" or "IN" in the WHERE clause. Partition columns like CLIENT that are specified both with "=" and used in a join aren't considered for partition-wise grouping operations (due to grouping simplification), so expensive global grouping / distinct operations can happen. Determining partition pruning can be quite time consuming if many partition keys are involved. See SAP Note 2000002 ("What are typical approaches to tune expensive SQL statements?" -> "Execution time higher than expected, negative impact by existing partitioning") for more details. Updating a column that is used for partitioning imposes the possibility of a partition move and significant overhead is possible in this context, see e.g. SAP Note 2100040 ("How can CPU intensive operations in SAP HANA be identified and optimized?" -> "TrexStore::UdivListManager::getEqualSSNUdivs"). A reduced number of partition columns reduces the risk of this scenario. In case of HASH partitioning it is often useful to use only the most selective primary key column as partition key column.
SAP Suite on HANA: All partitions on same host	In scale-out Suite on HANA environments it is typically of advantage to keep all partitions of a table on the same host. As of SPS 08 this can be achieved with an appropriate table placement configuration. As a fallback option you can use a dummy first level partitioning (e.g. on MANDT) and perform the actual partitioning on the second level. In this case all partitions will be located on the same host.
Repartitioning: Choose new number of partitions as multiple / divider of current number of partitions	If a table is already partitioned it is most efficient to choose a new number of partitions that is a factor 2 multiple or divider of the current number of partitions (e.g. 4 --> 8 or 6 --> 3 partitions), because only in this case the repartitioning can happen in parallel on different partition groups and hosts (“parallel split / merge”).
No additional unique constraints	Avoid partitioning tables with additional unique constraints (e.g. unique secondary index), because the uniqueness checks impose a significant overhead.

SAP Note 2000002 -> "What are typical approaches to tune expensive SQL statements?" -> "Execution time higher than expected, negative impact by existing partitioning" describes symptoms that can be introduced by inadequate partitioning.

14. How can partitioning changes be implemented?

SAP HANA Database Partitioning by Irshad Rather

In BW environments partitioning is typically automatized and no manual repartitioning is required. See SAP Note 2143736 for more information.

In other environments you can use table distribution as described in SAP Note 2081591.

In SLT scenarios you can define the partitioning scheme for new tables via transaction LTRS -> Table Settings -> Partition Command (SAP Note 2528241).

Alternatively it is possible to use individual SQL statements. The table below contains some important examples. For further details check the SAP HANA SQL Reference.

Activity	Example command
Create a table with HASH partitioning	CREATE COLUMN TABLE ... PARTITION BY HASH (<column1>, ..., <columnN>) PARTITIONS <num_partitions>
Create a table with ROUNDROBIN partitioning	CREATE COLUMN TABLE ... PARTITION BY ROUNDROBIN PARTITIONS <num_partitions>
Create a table with RANGE partitioning	CREATE COLUMN TABLE ... PARTITION BY RANGE (<column1>) (PARTITION 1 <= VALUES < 100, PARTITION 100 <= VALUES < 200, PARTITION VALUE = 400, PARTITION OTHERS)
Create a table with multi-level HASH / RANGE partitioning	CREATE COLUMN TABLE ... PARTITION BY HASH (<column1>, <column2>) PARTITIONS <num_partitions>, RANGE (<column3>) (PARTITION 1 <= VALUES < 5, PARTITION 5 <= VALUES < 20)
Add a new RANGE partition	ALTER TABLE ... ADD PARTITION 200 <= VALUE < 300
Drop an existing RANGE partition	ALTER TABLE ... DROP PARTITION 100 <= VALUE < 200
Move a partition to a different host	ALTER TABLE ... MOVE PARTITION <partition> TO '<host>:<port>'
Adjust partitioning of an already existing table	ALTER TABLE ... PARTITION BY ...
Transfer a partitioned table in a non-partitioned table	ALTER TABLE ... MERGE PARTITIONS

Tables partitioned by applications for aging (time selection partitioning) cannot be re-partitioned or converted to a non-partitioned table ("If a table is partitioned with Time Selection, it is not allowed to repartition it to anything") prior to SAP HANA 2 SPS3. This is a limitation caused by the non-enforced constraint checks on historical partitions. New re-partitioning/converting options are offered from SAP HANA 2 SPS3 onwards (SAP Note 2416490).

15. How can the consistency of partitions be checked?

The consistency of partitions, which e.g. includes the correct assignment of records to partitions, can be checked with the following CHECK_TABLE_CONSISTENCY options:

- CHECK_PARTITIONING
- CHECK_PARTITIONING_DATA

See SAP Note 1977584 for more details.

16. Is it possible to truncate a partition?

SAP HANA Database Partitioning by Irshad Rather

Up to SAP HANA 2.0 SPS 05, it is not possible to use a TRUNCATE operation on partition level. Starting with SAP HANA 2.0 SPS 06 it is possible to by specifying a list of partitions to be truncated.

Example: (truncation of partition IDs 2 and 3 of table ZMF_PART)

```
TRUNCATE TABLE ZMF_PART PARTITION (2, 3)
```

See "How can database requests be restricted to a subset of partitions of a table?" for more information.

17. How should tables be partitioned in BW environments?

BW takes care of the partitioning of its tables on its own, manual intervention is usually not required. You mainly have to take care that the table placement configuration is maintained properly (SAP Notes 1908075 and 2334091). Table distribution (SAP Note 2143736) will then implement the configuration.

The number of first level partitions depends on number of records in the largest table of the table group respectively the TABLE_PLACEMENT configuration. Default scenario:

Records	Partitions
< 40 million	1
40 - 120 million	3
120 - 240 million	6
> 240 million	12

The following table provides an overview how tables are typically partitioned in BW environments. <bw_prefix> is a place-holder for BW related table prefixes like "/BIC", "/BI0" or other prefixes defined in table RSNSPACE.

Object type	Object detail	Table name	First level partitioning	Second level partitioning	SAP Note	Details
advanced DSO	inbound queue	/<bw_prefix>/A<dso_name>1	HASH <semantic_key>	RANGE [DYNAMIC] REQTSN	2081135 2374652	As of SAP HANA Rev. 83 dynamic range partitioning can be used for the inbound queues of advanced DSOs. See "Is there a way to automatize the creation of new range partitions?" for more information.
advanced DSO	active data	/<bw_prefix>/A<dso_name>2	HASH <semantic_key>	optional: RANGE <user_defined_fields>	2081135 2374652	
advanced DSO	change log	/<bw_prefix>/A<dso_name>3	HASH <semantic_key>	RANGE [DYNAMIC] REQTSN	2081135 2374652	As of SAP HANA Rev. 83 dynamic range partitioning

						can be used for the change logs of advanced DSOs. See "Is there a way to automatize the creation of new range partitions?" for more information.
advanced DSO	validity table	/<bw_prefix>/A<dso_name>4	HASH REQTSN, <optional_validity_characteristics>		2081135 2374652	
advanced DSO	reference points	/<bw_prefix>/A<dso_name>5	HASH <semantic_key>	optional: RANGE <user_defined_fields>	2081135 2374652	
standard DSO	active data	/<bw_prefix>/A<dso_name>00	HASH <semantic_key>	optional: RANGE [FISCPER | CALMONTH]	1908075 2334091	Second level range partitioning is optional and either done on FISCPER or on CALMONTH.
standard DSO	activation queue	/<bw_prefix>/A<dso_name>40	HASH <semantic_key>		1908075 2334091	
standard DSO	change log	/<bw_prefix>/B*	HASH REQUEST, DATAPAKID, RECORD	optional: RANGE PARTNO	1767880 2081135	PSA tables and standard DSO change log tables share the naming convention and structure.
write-optimized DSO	'Allow duplicate data records' = TRUE	/<bw_prefix>/A<dso_name>00	HASH REQUEST, DATAPAKID, PARTNO	optional: RANGE PARTNO	1767880	With NetWeaver <= 7.30 (7) different partitioning approaches were used.
write-optimized DSO	'Allow duplicate data records' = FALSE	/<bw_prefix>/A<dso_name>00	HASH <semantic_key>		1767880	With NetWeaver <= 7.30 (7) different partitioning approaches were used.
PSA	PSA	/<bw_prefix>/B*	HASH REQUEST, DATAPAKID, RECORD	optional: RANGE PARTNO	1767880 2081135	PSA tables and standard DSO change log tables share the naming convention and

						structure.
dimension tables		/<bw_prefix>/D*				In case of dimension tables with a high number of records you can consider the transition to a line item dimension so that the SIDs are directly joined rather than being part of a dimension table.
classic E fact tables	compressed	/<bw_prefix>/E<cube_name>	ROUNDROBIN	RANGE KEY_<infocube>P, <dimids>	19080752334091	In each first-level partition three second-level RANGE partitions are defined on the KEY_<infocube>P which refers to different types of fact table data: Compressed requests Reference points Historic movements
classic F fact tables	uncompressed	/<bw_prefix>/F<cube_name>	ROUNDROBIN		19080752334091	
flat / in-memory optimized / HANA optimized F fact tables	combined	/<bw_prefix>/F<cube_name>	ROUNDROBIN	RANGE KEY_<infocube>P, <dimids>	19080752334091	In each first-level partition four second-level RANGE partitions are defined on the KEY_<infocube>P which refers to different types of fact table data: Compressed requests Uncompressed requests Reference points Historic movements
InfoObjects	SID table	/<bw_prefix>/S*			1331403	Generally avoid partitioning SID tables (/BI0/S*,

						/BIC/S*) because the combination of partitioning, two unique indexes and the particular change load can result in problems due to uniqueness checks (thread method "CheckRemoteUniqueConstraint", see SAP Note 2114710). See SAP Notes 1331403 and 2019973 for managing large SID tables from a BW perspective.
InfoObjects	other tables	/<bw_prefix>/H*/<bw_prefix>/I* /<bw_prefix>/J* /<bw_prefix>/K* /<bw_prefix>/P* /<bw_prefix>/Q* /<bw_prefix>/T* /<bw_prefix>/X* /<bw_prefix>/Y*			2019973	These tables aren't partitioned per default and should only be partitioned in rare cases when there is a risk that the 2 billion record limit is reached. In that case consider remodeling the scenario or InfoObjects with high cardinality.
InfoObjects with high cardinality	time-independent attributes	/<bw_prefix>/P*	HASH <InfoObject>, OBJVERS		2019973	For InfoObjects with high cardinality, P-, Q- and T-table are partitioned by default.
InfoObjects with high cardinality	time dependent attributes	/<bw_prefix>/Q*	HASH <InfoObject>, OBJVERS, DATETO		2019973	For InfoObjects with high cardinality, P-, Q- and T-table are partitioned by default.
InfoObjects	master data	/<bw_prefix>/T	HASH		2019973	For InfoObjects

with high cardinality	texts	*	<InfoObject>, LANGU, DATETO			with high cardinality, P-, Q- and T-table are partitioned by default. DATETO is part of the partition specification if time-dependent texts are used.

In addition to the above database partitioning BW also provides the option for semantic partitioning. This is based on different tables with the same structure, but it doesn't involve partitioning on SAP HANA level.

18. Is the partitioning information kept during homogeneous SAP HANA system copies?

If the homogeneous system copy is performed using backup and restore the partitioning information is kept.

A system copy based on R3load doesn't copy the partitioning information per default. If you want to keep it you have to use the SMIGR_CREATE_DDL report.

19. Is the partitioning information kept during transports?

Partitioning information is not available in ABAP DDIC. Therefore transports don't consider it and partitioning is not transported. This means that you have to activate partitioning in all involved systems of your ABAP system landscape individually.

20. Is the partitioning information kept during ABAP table conversions and SAP upgrades?

Partitioning information is kept when a table conversion in SAP ABAP is performed and when a SAP upgrade is performed.

21. My table has a partition specification, but it shows only PART_ID 0 in M_CS_TABLES. Is this correct?

Tables partitioned with HASH or ROUNDROBIN and only one partition are non-partitioned tables which have a partition specification.

22. Where can I see which partitions are loaded into memory?

This information is available in column LOADED of monitoring view M_CS_TABLES:

LOADED	Description
NO	No column of partition is loaded into memory
PARTIALLY	Some columns of partition are loaded into memory
FULL	Partition is completely loaded into memory

SQL: "HANA_Tables_Partitions" (SAP Note 1969700) can be used to display the LOADED state and other partition details.

23. Is it possible to load a single partition only?

No. The load and preload operations consider all partitions of a table (exception: see next question).

24. Are cold partitions for aging and time-selection tables pre-loaded?

As of SAP HANA 1.0 SPS 09, load and pre-load do not consider cold partitions.

25. How many partitions are allowed for one table?

There is a maximum of 1000 (SPS <= 09) or 16000 (SPS >= 10) partitions for one table. For multi-level partitioning, multiply the number of first-level partitions with the number of second-level partitions to get the total number of partitions. This maximum number for one table is independent of the table location in a scale-out landscape.

26. Which data types are allowed for partitioning columns?

The supported data types are listed in the Partitioning Limits section of the SAP HANA Administration Guide.

Partitioning is not supported for columns defined with GENERATED ALWAYS and for concat attribute columns (SAP Note 1986747).

27. Are there specific partitioning recommendations for certain SAP applications and tables?

For the standard approach (HASH partitioning on a selective column, typically part of primary key), the following partitioning recommendations can be used. Typically this partitioning layout is a good compromise between efforts and results. The number of partitions depends on the reason for partitioning (e.g. reducing data footprint per partition, reducing records per partition).

Be aware that for tables making use of Data Aging (SAP Note 2416490) or NSE (SAP Note 2799997) there are certain limitations related to partitioning and so the simple standard approach isn't possible.

Table	Type	Columns
/1CADMC/<id>	HASH	IUUC_SEQUENCE
ACDOCA	HASH	BELNR
ACCTIT	HASH	AWREF
ADRC, ADRU	HASH	ADDRNUMBER
AFFV	HASH	AUFPL
AUSP	HASH	OBJEK
BALDAT	HASH	LOG_HANDLE
BDCP2	HASH	CPIDENT
BDSCONT10, DMS_CONT1_CD1, SBCMCONT1	HASH	PHIO_ID
BKPF, BSEG, BSIS	HASH	BELNR
CDHDR, CDPOS	HASH	OBJECTID, CHANGENR or TABKEY (use column with best value distribution and use same column for both tables if possible, in some cases OBJECTID for CDHDR and CHANGENR for

		CDPOS can be the best solution)
CE4<operating_concern>, CE4<operating_concern>_ACCT e.g.: CE41000_ACCT, CE4A001_ACCT, CE4HI01, CE4HI01_ACCT	HASH	PAOBJNR
CIFBALSEL	HASH	LOGNUMBER
CKMLKEPH	HASH	KALNR
COEP	HASH	BELNR
COFV	HASH	CRID
COKA, COSB	HASH	OBJNR
DBERCHZ<id>	HASH	BELNR
DBERDL, DBERDLB	HASH	PRINTDOC
DBTABLOG	HASH	LOGID
DBVM	HASH	MATNR
DFKKCODCLUST	HASH	COKEY
DFKKKO, DFKKOP, DFKKOPK, DFKKOPW, DFKKZR	HASH	OPBEL
DFKKOPBW	HASH	RECNO
DIMAPARSCPOS	HASH	POSNR
/DOC/D_TC_TEXT_CNT	HASH	NODE_ID
DPAYP	HASH	PAYNO
EDID4, EDIDS	HASH	DOCNUM
EIPO	HASH	EXNUM
EKBE	HASH	BELNR
EKPO	HASH	EBELN
EMMA_JOBRUNIDMSG	HASH	SRTFD
EQKT	HASH	EQUNR
FAGLFLEXA	HASH	DOCNR
FAGL_SPLINFO, FAGL_SPLINFO_VAL	HASH	BELNR
FKKMAZE	HASH	OPBEL
FPLT	HASH	FPLNR
FQM_FLOW	HASH	FLOW_ID
GLFUNCA, GLPCA	HASH	GL_SIRID
ICLACTIVITY, ICLITEM, ICLITEMEV, ICLPARTOCC, ICLPAYI, ICSACTIVITY, ICSITEMEV, ICSSUBCL	HASH	CLAIM
IDOCREL	HASH	ROLE_A
INDX	HASH	SRTFD
JCDS, JEST	HASH	OBJNR
KEPH	HASH	KALNR

KONV	HASH	KNUMV
KSSK	HASH	OBJEK
LIPS	HASH	VBELN
LOYD_MA_GENATTR, LOYD_MA_SPECATTR, LOYD_MSH_MEMS	HASH	GUID
MARDH	HASH	MATNR
MATDOC	HASH	MBLNR
MBEW, MBEWH, MVER, MYMFT	HASH	MATNR
MLAUF, MLAUFCR, MLAUFCRLD, MLAUFKEPH, MLAUFKEPHLD, MLBE, MLBECR, MLBECRLD	HASH	MLVNR
MLDOC, MLDOCCCS	HASH	DOCREF
MSEG	HASH	MBLNR
NAST	HASH	OBJKY
OBJK	HASH	OBKNR
PCL2, PCL4	HASH	SRTFD
POC_DB_VALUE	HASH	OS_GUID
/POSDW/PLOG1S	HASH	GUID
PPOIX	HASH	PERNR
PRCD_ELEMENTS	HASH	KNUMV
REGUP	HASH	BELNR
RSBATCHDATA	HASH	TIMESTAMP
RSBMLOGPART_DTP, RSBMONMESS_DTP, RSBMREQ_DTP	HASH	REQUID
RSDDSTATDTP	HASH	INSTANCE
RSEG	HASH	BELNR
RSHIENODETMP, RSMONMESS	HASH	RNR
RSODSACTUPDTYPE	HASH	REQUEST
RSRWBSTORE	HASH	WORKBOOKID
S027, S827	HASH	MATNR
SBCMCONT1	HASH	PHIO_ID
/SCDL/DB_REFDOC	HASH	REFDOCID
/SCDL/DB_STATUS	HASH	DOCID
SLPE_RT_PLOG	HASH	LOG_ID
SOC3	HASH	SRTFD
SRAL_EXP_DATA	HASH	LOG_ID
SRRELROLES	HASH	OBJKEY
/SSF/BTAB	HASH	OBJKEY
/STTP/SNR_LIST	HASH	SERNO
SWFRCNTXML	HASH	GUID

SWFREVTLOG	HASH	LOGGUID
SWPNODELOG	HASH	WF_ID
SWWCNTP0, SWWLOGHIST	HASH	WI_ID
T0A01	HASH	ARC_DOC_ID
VBFA	HASH	SoH: HASH (VBELV), S/4HANA: HASH (RUUID)
VBOX	HASH	VAKEY
VAPMA, VBAP, VBPA, VBUP	HASH	VBELN
VBSK	HASH	SAMMG
VEKP, VEPO	HASH	VENUM
VVSCITEM, VVSCPOS	HASH	POSNR
WLK1	HASH	ARTNR

In addition the following SAP Notes contain suggestions for partitioning specific SAP application tables. In general it is recommended to proceed based on the standard approach described above unless it is required to implement a more complex partitioning for specific reasons.

SAP Note	Application	Tables
1719282	SAP Point of Sale (POS) SAP Customer Activity Repository (CAR)	/POSDW/TLOGF, /POSDW/TLOGF_EXT
2190377 (single node) 2700982 (scale-out)	SAP Unified Demand Forecast (UDF) SAP Demand Data Foundation(DDF)	/DMF/LANE, /DMF/LANE_PRC, /DMF/LANE_TD, /DMF/OFR_FIN_DTL, /DMF/OFR_PL, /DMF/PRDLOCEXTXR, /DMF/PRODLOC, /DMF/PRODLOC_PRC, /DMF/PRODLOC_TD, /DMF/TS_INV, /DMF/TS_PS, /DMF/TS_UN, /DMF/UFC_TSD, /DMF/UMD_TSD, /DMF/UMD_PAR_COV, /DMF/UMD_TS, /DMF/UMD_PAR, /DMF/UMD_PRI, /DMF/UMD_MET, /DMF/UTASK_CONT
2259038	S/4HANA material management	MATDOC
2289491	SoH / S/4HANA finance tables	ACCTCR, ACCTHD, ACCTIT, ACDOCA, BSEG, BSAS, BSIS
2418299	SAP ABAP	CDPOS, EDID4, JEST
2524869	Bank Analyzer / Smart AFI	/BA1/BR_REG_AD, /BA1/BR_REG_BT, /BA1/BR_REG_MD, /BA1/FC_PCDSTATF, /BA1/FC_REGSTAT, /BA1/F1_CON_FLAT, /BA1/F1_*_FLAT, /BA1/F2_BT_FLAT, /1BA/HM_*
2722355	S/4HANA Finance Products Subledger	/BA1/F1_BPR_FLAT, /BA1/F1_CFH_FLAT, /BA1/F1_CON_FLAT, /BA1/F1_CRL_FLAT, /BA1/F1_CTN_FLAT, /BA1/F1_FCC_FLAT, /BA1/F1_FCD_FLAT, /BA1/F1_KFG_FLAT, /BA1/F1_LIM_FLAT, /BA1/F1_OPD_FLAT, /BA1/F1_OPE_FLAT, /BA1/F1_OPF_FLAT, /BA1/F1_OPH_FLAT, /BA1/F1_RCH_FLAT, /BA1/F1_RRC_FLAT, /BA1/F2_BT_FLAT, /BA1/HFGPD, /BA1/HFPPD, /BA1/HFSPD, /BA1/HKAAS, /BA1/HKACG, /BA1/HKAMS, /BA1/HKANA, /BA1/HKAPA, /BA1/HKAPD, /BA1/HKCDA, /BA1/HKCFO, /BA1/HKCFR, /BA1/HKCOR, /BA1/HKEAS, /BA1/HKEPR, /BA1/HKEPS, /BA1/HKETV, /BA1/HKIMT, /BA1/HKLFP, /BA1/HKLSE, /BA1/HKMES, /BA1/HKPAI, /BA1/HKPAP, /BA1/HKPAT, /BA1/HKPGD, /BA1/HKRIC, /BA1/HKRMC, /BA1/HKRPE, /BA1/HKRPS, /BA1/HKTVL, /BA1/HKTVR, /BA1/HKULT, /BA1/HKVEC,

		/BA1/HKVOL, /BA1/HSAAS, /BA1/HSACG, /BA1/HSAMS, /BA1/HSANA, /BA1/HSAPA, /BA1/HSAPD, /BA1/HSCDA, /BA1/HSEAS, /BA1/HSEPR, /BA1/HSEPS, /BA1/HSETV, /BA1/HSLFP, /BA1/HSMES, /BA1/HSPAI, /BA1/HSPAP, /BA1/HSPAT, /BA1/HSRIC, /BA1/HSRMC, /BA1/HSRPE, /BA1/HSRPS, /BA1/HSSPD, /BA1/HSULT, /BA1/HSVEC, /BA1/HSVOL, /BA1/DT_EOPR_WL, /BA1/HKCBD, /BA1/HKCBD_EBA, /BA1/BR_REG_BT, /BA1/BR_REG_MD, /BA1/BR_REG_MDC, /BA1/BR_REG_AD, /BA1/FC_PCDSTAT, /BA1/FC_REGCAP, /BA1/FC_MATCH_SR, /BA1/BR_REG_BPC, /BA1/BR_CAP_BECF
2845463	Material Ledger	CKMLCR
2849678	Transactional Banking	/FSFAC/RES_XL, /FSFAC/RLOG_XL, /FSFAC/ULOG_XL, /FSFAC/UTIL_XL, BCA92, BCA92_RESTART, BCA96, BCA98, BCASO_PAORN, BCA_ACCTBAL, BCA_ACCTBAL_FP, BCA_BANO_DUE, BCA_BCAS_DUE, BCA_BCT_CN_OBJV, BCA_BL_SCHED, BCA_BL_SCHED_H, BCA_CAP_KDATE, BCA_CARD_HEADER, BCA_CNFW_EVENT, BCA_CNSP_ACCT, BCA_CN_EVENT, BCA_CN_LINK, BCA_CN_PER_ACBAL, BCA_CN_SL_DATA, BCA_CN_TDLK, BCA_CONTRACT, BCA_COUNTER, BCA_GL_BALCN, BCA_GL_BPITEM, BCA_GL_BSPR_PROT, BCA_GL_PAYMITEM, BCA_GL_SNITEM, BCA_GL_TRNTP, BCA_INV_DETAILS, BCA_NOW_ABS, BCA_PAYMITEM, BCA_PAYMITEM_NT, BCA_PO_IT, BCA_PO_NT, BCA_PRENOTE, BCA_SB_2BR, BCA_SB_2BR_CHNG, BCA_SNITEM, BCA_TRANSFIG, BCA_TRNOVER, BKK92_POSTINGS, BKK92_SIM, BKK92_SUMS, BKK92_SUMS_SIM, BKK96_SIM, BKKSOHD, BKKSOIT, BKKSOITNT, BKKSOIT_VAR_AMNT
3139140	Material Ledger - Actual Costing	MLDOC, MLDOCCCS
2014562	SLT logging tables	/1CADMC/<id>

28. Is there a way to automatize the creation of new range partitions?

Normally it is required to define necessary range partitioning manually. An automatic partitioning is possible using the dynamic range partitioning feature. With the following command you can define a partition as dynamic:

```
CREATE COLUMN TABLE <table> (<columns>) PARTITION BY RANGE (<part_columns>) (PARTITION OTHERS
DYNAMIC THRESHOLD <threshold_rows>)
```

Dynamic range partitioning for an already range partitioned column can be activated / deactivated in the following way:

```
ALTER TABLE <table> PARTITION OTHERS DYNAMIC THRESHOLD <threshold_rows>
ALTER TABLE <table> PARTITION OTHERS NO DYNAMIC
```

With this feature you can use the following parameters to automatize the split of the dynamic partition based on the number of records:

Parameter	Default	Unit	Details
indexserver.ini -> [partitioning] -> dynamic_range_default_thr eshold	10000000	rows	SAP HANA automatically splits the dynamic partition into two partitions once the row threshold has been reached. The system-wide default can be overwritten

			by table placement settings (SAP Note 2081591).
indexserver.ini -> [partitioning] -> dynamic_range_check_time_ interval_sec	900	s	This parameter defines how often the threshold check is performed.

With current patch levels dynamic range partitioning is per default used in the following scenarios:

- BW advanced DSO tables (SAP Note 2081135)
- Characteristics with Enhanced Master Data Update
- Data Transfer Immediate Storage (DTIS) tables
- RSPM* tables (e.g. RSPMDATEPID or RSPMLOG) in BW

Be aware that dynamic range partitioning isn't supported for BW business tables (/.../*) in contexts different from the ones described above.

The dynamic range check can be quite expensive in case many tables with dynamic range partitioning exist because an implicit TABLE_GROUPS lookup can be quite expensive in context of a high number of temporary BW tables. See SAP Note 2000002 -> TABLE_GROUPS (statement hash 25a6171ba41bdf171e818c986177f37e) for more information. The garbage collection (SAP Note 2169283) is globally blocked while the dynamic range check runs.

Be aware that the transactional lock wait timeout for the internal dynamic repartitioning can differ from the default. See SAP Note 1999998 ("Are there specific operations using a non-default transactional lock wait timeout?") for more details.

29. How can I check the progress of an ongoing partitioning activity?

Partitioning activities can be monitored in the following ways:

Tables	SQL statement (SAP Note 1969700)	Details
M_SERVICE_THREADS M_SERVICE_THREAD_SAMPLES HOST_SERVICE_THREAD_SAMPLES	SQL: "HANA_Threads_CurrentThreads" SQL: "HANA_Threads_ThreadSamples_FilterAndAggregation" SQL: "HANA_Threads_ThreadSamples_AggregationPerTimeSlice"	Partitioning related thread information can provide insight which tables are currently partitioned and what kind of detailed activity is executed. See SAP Note 2114710 for more information.
M_JOB_PROGRESS	SQL: "HANA_Jobs_JobProgress"	The view M_JOB_PROGRESS contains information for the current partitioning related activities (JOB_NAME = 'Re-partitioning'). For details how to interpret the details see "How can the job progress details be interpreted for repartitioning tasks?" below.

M_EXECUTED_STATEMENTS	SQL: "HANA_SQL_ExecutedStatements"	The executed statements trace (SAP Note 2366291) provides information about executed DDL operations. You can e.g. run the command with SQL_PATTERN = '%<table_name>%' in order to find all DDL operations that were executed in the repartitioned table <table_name>.

30. What does _SYS_SPLIT in SAP HANA monitoring views and traces mean?

At different places in monitoring views and trace files (SAP Note 2119087) you can find strings starting with _SYS_SPLIT like:

```
_SYS_SPLIT_<table_name>~<part_id>
```

This is the internal representation for a table partition. The trailing identifier indicates the partition ID.

31. What standard partitioning approach is used during migrations?

If a table is partitioned during the migration to SAP HANA (e.g. in order to bypass the 2 billion record limit), HASH partitioning based on the complete primary key is used per default. Per default, tables with more than 1 billion records are considered for partitioning by DMO.

Exception: Tables originating from former physical cluster tables (e.g. BSEG originating from RFBLG) are partitioned by the primary key of that physical cluster table excluding the PAGENO field.

SAP Note 2396601 describes how you can adjust the default partitioning during migrations.

SAP Notes 2779173 and 2784715 describe how to configure partitioning in context of SWPM.

32. What should be considered in terms of terminating long running repartitioning operations?

Manual termination of repartitioning is possible. A rollback happens and no inconsistency is introduced.

Very long running repartitioning operations may also be terminated by the MVCC anti ager / Kernel Sentinel Job (SAP Note 2169283) with errors like "Data receive failed [Connection reset]" once the idle cursor time exceeds the timeout defined with the following parameter (default: 12 hours):

```
indexserver.ini -> [transaction] -> idle_cursor_lifetime
```

Consider a temporary increase of this parameter if you expect that a repartitioning operation takes longer than the configured limit. See SAP Note 2890332 for more information.

33. How can partitioning operations be traced?

You can activate the trace of partitioning operations with the following database trace parameter:

```
indexserver.ini -> [trace] -> partitioning = debug
```

See SAP Note 2119087 for more information related to the database trace.

34. Can range partitions overlap?

No, range partitions need to be disjoint, an overlap isn't possible. Be aware that you have to be careful interpreting boundary information.

Example:

```
CREATE COLUMN TABLE AAA (X INTEGER) PARTITION BY RANGE (X)
( PARTITION 1 <= VALUES < 100,
PARTITION 100 <= VALUES < 200,
PARTITION OTHERS );

-- alternative notation

CREATE COLUMN TABLE AAA (X INTEGER) PARTITION BY 'RANGE X 1-100,100-200,*'
```

The partition ranges shown in M_CS_PARTITIONS seem to overlap in value 100:

```
-------------------------------------------------
|TABLE_NAME|PART_ID|RANGE |SUBRANGE|
-------------------------------------------------
|AAA | 1 |1-100 | |
|AAA | 2 |100-200| |
|AAA | 3 | | |
-------------------------------------------------
```

This is a wrong conclusion. The upper limit of the range is exclusive and not inclusive, so value 100 is always inserted in partition 2, never in partition 1.

35. Which errors can happen in relation to partitioning?

The following errors can happen in the partitioning context:

Error message	Details
Could not allocate value '<value>' for column '<column>' to a part of table <schema>:<table>en. The table has no rest part. Create a new partition for the value. The table has no OTHERS part. Create a new partition for the value.	This error is issued if a record is inserted, a value isn't covered by the partition definition and no OTHERS partition is defined. Example: A table is defined with range partitions for each calendar year between 2010 and 2016 and now a new record is inserted with calendar year 2017. Adjust the partitioning or the application logic so that all inserted values are actually covered by the partitioning scheme. The corresponding Oracle error message is: ORA-14400: inserted partition key does not map to any partition
7: feature not supported: cannot use round-robin partitioning with primary key 7: feature not supported: cannot add primary key to round-robin partitioned table 5001: partition specification not valid; The table has a primary key. RoundRobin is only allowed for tables without primary keys.	Tables with ROUNDROBIN partitioning must not have a primary key because otherwise each INSERT and UPDATE would require a key check in all partitions.
7: feature not supported: NUMA node	Specifying a NUMA node (e.g. via "ALTER PARTITION ... NUMA

preference not supported on current partition spec of table: <table>	NODE") only works for range partitioning. Doing it in other partition contexts, this error is thrown.
7: feature not supported: cannot set partition for temporary tables 7: feature not supported: partition is not supported for row tables	Partitioning is neither supported for temporary tables (SAP Notes 2800007) nor row tables (SAP Note 2222277).
7: feature not supported: could not complete partitioning: partition spec must be HASH, HASH-HASH, HASH-RANGE, RANGE, or RANGE-RANGE	In case of sub-partitioning only specific combinations of partition types are allowed. Repartitioning (ALTER TABLE ... PARTITION) a table with configured data aging (SAP Note 2416490) is also not possible and fails with the same error.
7: feature not supported: The table is partitioned with a time selection partitioning. For this kind of partitioning it is not allowed to change loading type.	It is not possible to adjust the load unit of a table using time selection partitioning (SAP Note 3142798).
column store error: process parameters error: [2051] partition specification not valid; The column <column> is not a key column.	First level hash and range partitioning can only be done based on columns belonging to the primary key of the table. If you try to use a different column, you receive this error.
7: feature not supported: not allowed on table having page loadable column 2048: column store error: fail to alter partition: [2593] Error during split/merge operation; If a table is partitioned with Time Selection, it is not allowed to repartition to non Time Selection table except for merging Time Selection range partitions.	This error is issued if an already TIME SELECTION partitioned table is re-partitioned via an ALTER TABLE statement without providing all the TIME SELECTION partitions again (SAP Note 2416490) Example: A Table TAB_DATAAGING is 1-level HASH 2 partitioned on column B and 2-level TIME SELECTION (RANGE) partitioned with hot partition (00000000) and one cold partition (20100101-20110101). If for changing the 1-level HASH partition KEY from B to A following statement is used, the re-partitioning is failing with this error: ALTER table TAB_DATAAGING partition by hash (A) partitions 2; It is crucial to always provide all sub partitions (RANGES) within the ALTER TABLE statement like: ALTER TABLE TAB_DATAAGING WITH PARAMETERS('PARTITION_SPEC' = 'HASH 2 A; RANGE[TIME SELECTION: PAGED ATTRIBUTES, NO UNIQUE CHECK] _DATAAGING 00000000,20100101-20110101');
435: invalid expression: partition column not in primary key columns: <columns>	As described in "Which partition types exist?" certain first-level partitionings need to be based on primary key columns. This restriction doesn't exist for heterogeneous partitioning, see "What is heterogeneous partitioning?".
1094: invalid combination of settings specified: cannot create SIMPLE VALID FOR DATA DEPENDENCY on unpartitioned column store table	This error happens when you try to create SIMPLE VALID FOR DATA DEPENDENCY data statistics in context of dynamic partition pruning and the table is not partitioned. This statistics type is only useful and permitted in context of partitioned tables.
2048: column store error: fail to alter partition: [2593] Error during split/merge operation; Moving data between servers is not allowed. During Repartitioning.,object=<object>	This error is issued if an already partitioned table needs to be repartitioned in a scale-out environment. The workaround is to move all table partitions to one host and start the repartitioning.

36. What are typical partitioning runtimes?

SAP HANA Database Partitioning by Irshad Rather

Partitioning performance depends on many factors like available resources and SAP HANA configuration (see "How can the performance of repartitioning activities be influenced?" for details). During real-life repartitioning of large tables a throughput of 10 to 100 GB / h was observed.

37. Are there specific scenarios where partition pruning isn't possible?

In general SAP HANA tries to take advantage of partition pruning as much as possible. One exception is the use of functions on partition columns. For example, when a table is range partitioned based on month and in the WHERE clause YEAR(<column>) = 2018 is selected, all partitions are scanned and not only the 12 partitions belonging to the year 2018.

38. What are the main repartitioning phases?

In SAP HANA 1.0 repartition consists of the following steps:

1. Acquire exclusive lock of table
2. Calculate repartition strategy
3. Perform delta merge
4. Create target partitions for the partitions that have changed
5. Distribute the data from the source to the target partitions
6. Finalization tasks, like cleanup of obsolete source partitions and recalculation of internal columns
7. Release exclusive lock of table

In SAP HANA >= 2.0 reduced locking was introduced and repartition consists of the following steps:

1. Acquire intentional exclusive lock of table
2. Calculate repartition strategy
3. Create target partitions for the partitions that have changed
4. Distribute the data of the main part from the source to the target partitions
5. Acquire exclusive lock of table
6. Distribute the data of the delta part from the source to the target partitions
7. Finalization tasks, like cleanup of obsolete source partitions and recalculation of internal columns
8. Release exclusive lock of table

39. How can the performance of repartitioning activities be influenced?

The following factors influence runtime and performance of repartitioning activities:

Area	SAP Notes	Details
System resources	1999997 2100040	Available system resources in terms of CPU and memory can impact the repartitioning runtime. It is important that enough memory is available to repartition without having to perform unloads or swapping.
Disk I/O performance	1999930	Bottlenecks in the I/O stack can significantly increase the repartitioning time.
SAP HANA configuration	2222250 2600030 2874176	In general it is important to make sure that SAP HANA is configured based on SAP best practices as described in SAP Note 2600030. In addition you can use the following setting to adjust parallelism for split and bulk load during table repartitioning: indexserver.ini -> [partitioning] -> split_threads (default: 16) indexserver.ini -> [partitioning] -> bulk_load_threads (default: 4) Higher values (up to 128 split threads and 20 bulk load threads) result in higher parallelism if sufficient resources are available and the partitioning criteria fulfill the multiple /

		divider rule (see "Partitioning criteria" below). If SAP HANA downgrades the parallelism due to limited memory, the following entry is written to the database trace (SAP Note 2380176): Requested to split index with X threads. As the memory is scarce, the number of threads is reduced to Y. SAP Note 2874176 provides further parameter recommendations for redistribution / repartitioning activities.
Delta storage	2057046	Size and workload on the delta storage can have a performance impact, so the following considerations are useful: Make sure that the initial delta storage is small and consider an explicit delta merge in case of larger delta storages. Minimize the amount of change operations that happen in parallel to the ongoing repartitioning so that the delta storage size is kept small. SAP Note 2871405 suggests a change to the auto merge decision function in context of online repartitioning operations in order to increase the number of merges, thus reducing the memory requirements.
Partitioning criteria		If you repartition tables that are already partitioned, the following rules can reduce the effort and runtime: HASH, ROUNDROBIN: Choose a new number of partitions that is a multiple or divider of the current number of partitions, because in this case SAP HANA is able to parallelize on source partitioning level. This rule also applies for multi-level partitioning. Both levels can be treated individually (e.g. repartitioning from 2/2 partitions to 2/4, 8/2 or 16/16 partitions possible). RANGE with OTHERS partition: When adding a new range to this table only new partitions for the new range and a new OTHERS partition are created. The data distribution time highly depends on the size of the OTHERS partition since all the data of the old OTHERS partition has to be redistributed either in the newly created range partition or the new OTHERS partition. previously non-partitioned tables: For tables that are currently not partitioned at maximum one thread per column can be used to distribute the data to the new partitions
Availability of table data in memory	2127458 2220627	All table data has to be read during repartitioning, so in the following scenarios significant overhead can be introduced when data has to be read from disk: Columns that weren't loaded to memory before Access to disk LOBs that are not cached in the SAP HANA page allocator The following command can be used to load all column tables into memory: LOAD "<table_name>" ALL From a LOB perspective it is more efficient to use packed LOBs. If a table doesn't contain packed LOBs, yet, the following can be used for migrating (SAP Note 2220627): ALTER TABLE "<schema_name>"."<table_name>" LOB REORGANIZE [ONLINE] Attention: With SAP HANA <= 2.00.046 orphan persistence files can remain in case packed LOBs were already used before and a manual cleanup is required (SAP Note 2910004).
Index considerations	2160391	The existence of large multi-column indexes can slow down the partitioning throughput in some cases. You can detect expensive index processing via thread analysis (SAP Note 2114710). For example, the following thread method indicates expensive parallelized processing for the primary key ($trexexternalkey$): split/merge attribute <schema>:<table> $trexexternalkey$ It is possible that a recreation of the index after the repartitioning is quicker than the actual repartitioning of the index. In this case you can proceed as follows and drop the index before starting the repartitioning: Make sure that the application is stopped (in case of primary keys and unique indexes to avoid risk of duplicate keys) Drop the index / constraint (DROP INDEX / ALTER TABLE ... DROP CONSTRAINT) Perform the actual repartitioning Recreate the index / constraint (CREATE INDEX / ALTER TABLE ... ADD CONSTRAINT)

40. How does an example for repartitioning look like?

To explain the details of repartitioning we use the following table as an example which is distributed over two indexserver running on port <indexserver_port> on hosts <HOST_A> and <HOST_B>:

```
CREATE COLUMN TABLE <SCHEMA_NAME>.<TABLE_NAME> (a INT, b INT, c INT, PRIMARY KEY (a,b))
PARTITION BY
HASH (a, b) PARTITIONS 2,
RANGE (c)
(PARTITION 1 <= VALUES < 5,
PARTITION OTHERS)
```

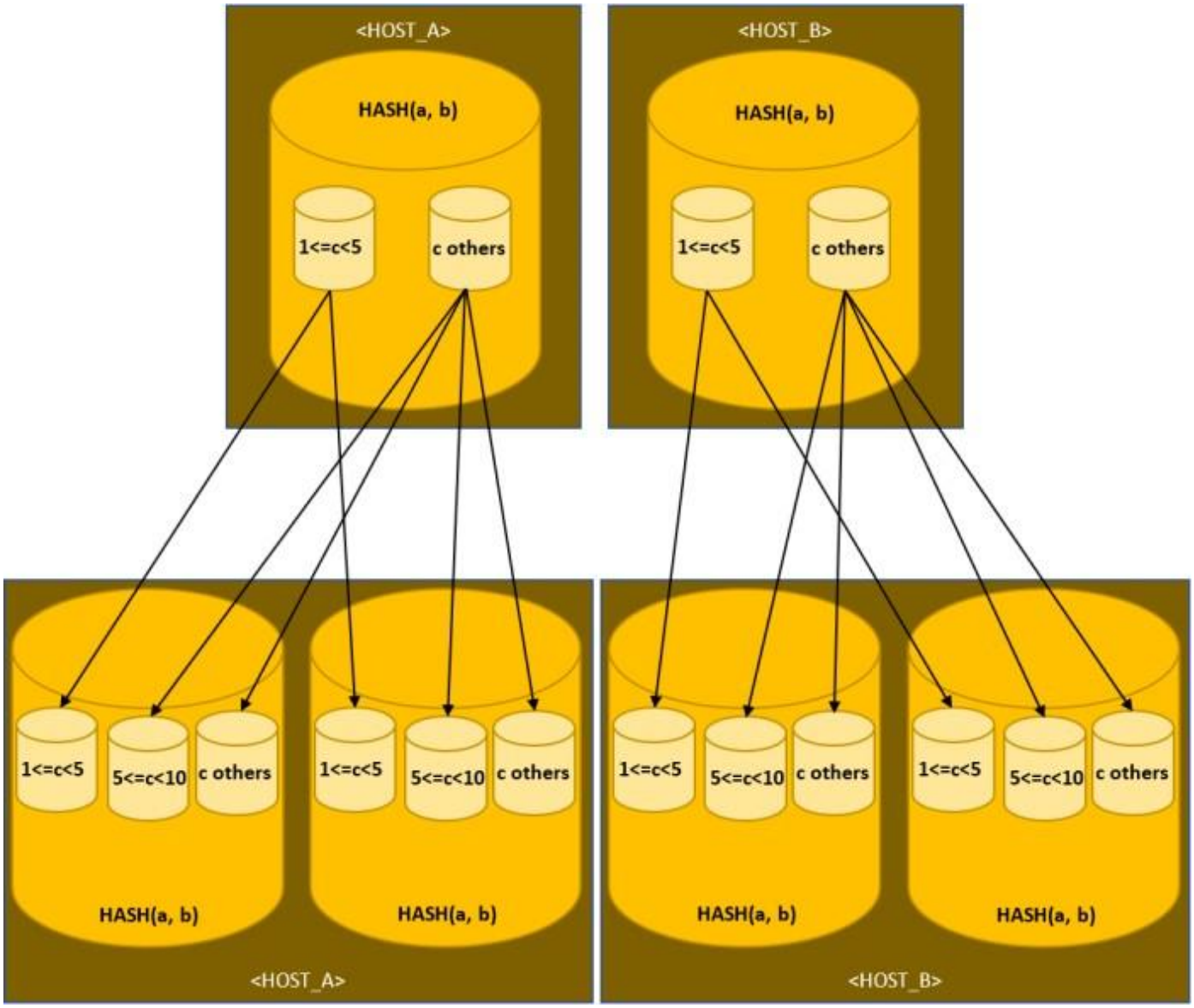
The partition specification is changed in such a way that on first level now 4 instead of 2 partitions exist. On second level an additional range partition for the range 5 <= c < 10 is added:

```
ALTER TABLE <SCHEMA_NAME>.<TABLE_NAME>
PARTITION BY
HASH (a, b) PARTITIONS 4,
RANGE (c)
(PARTITION 1 <= VALUES < 5,
PARTITION 5 <= VALUES < 10,
PARTITION OTHERS);
```

The number of hash partitions is increasing from 2 to 4. 4 is a multiple of 2. As described in "How can the performance of repartitioning activities be influenced?", this means that the repartition task can parallelize per source hash partition.

On the second partition level only one new range is added while the definition of the previously existing range is not changed. For this reason, further parallelization based on every source range partition can be performed.

Here you can find a visualization of the initial partitioning scheme, its changes and the data flow during the reorganization:



41. How can the job progress details be interpreted for repartitioning tasks?

The monitoring view M_JOB_PROGRESS contains detailed information about repartitioning progress that depends on the SAP HANA version.

Important details for SAP HANA 1.0 are:

- CURRENT_PROGRESS: Number of repartitioned columns
- MAX_PROGRESS: Total number of columns that need to be repartitioned (including internal columns)

With SAP HANA >= 2.0 the M_JOB_PROGRESS monitoring view contains more detailed information. It first contains information about the overall status of the MASTER process. This master process can create worker groups which are called GRP1, GRP2, … Basically every possible parallelization per partition gets assigned one group. Each group then can have up to one worker thread per partition column. These worker threads are called WORK0, WORK1, …

Important details for SAP HANA >= 2.0 are:

- CURRENT_PROGRESS: This value shows how many steps are already processed. Details about the current step can be found in the PROGRESS_DETAIL column.
- MAX_PROGRESS: Shows the overall number of steps that need to be performed. Depending on the

process type a different number of total steps exists:
  - The master process has in total 7 steps: Mapping, GetDataRanges, PreProcess, RepartitionMain, RepartitionDeltaAndMVCC, PostProcess, DropSources
  - Each worker group consists of 13 steps: CreateTargets, DeterminepartsMain, DeterminepartsDelta, InitPersistenceMain, InitPersistenceDelta, CheckConsistencyMain, CheckConsistencyDelta, RepartitionMainLocal, RepartitionDeltaLocal, RepartitionMVCCMain, RepartitionMVCCDelta, UpdateRuntime, Finalize
  - Each thread worker has 3 steps: SplitMergeAttributes, Save, Idle

- PROGRESS_DETAIL: [MASTER|GRPX|WORKX] [COLUMN_NAME] STEP_NAME
  Shows more details about the individual process:
  - For the master process it shows “MASTER” and the step of the overall repartition task.
    Example: MASTER RepartitionMain
  - For a worker group it shows the name of the group and the current step of the group. Additionally, it shows in brackets the partition IDs of the source table on which this group is working on.
    Example: GRP1 RepartitionMainLocal (1)
  - For a worker thread it shows the name of the worker and the current step of the worker thread. Moreover, it shows the name of the column for which it is currently repartitioning the data. Furthermore, the number of columns for which the redistribution finished when the step started and the overall number of columns in this partition is shown in brackets.
    Example: WORK1 A (3/6) SplitMergeAttribute
    Attention: The number of all columns also includes all internal columns. For some of these columns, like $rowid$ or $trex_udiv$, a special handling at the end of the repartition run exists. For this reason, it is likely that you will not see worker threads for the complete number of (internal) columns.

42. What are tables with names starting with "_SYS_OMR_"?

Tables with the naming convention _SYS_OMR_<source_table>#<id> are used as interim tables during online repartitioning operations that are triggered by commands like:

```
ALTER TABLE "<source_table>" PARTITION BY ... ONLINE
```

Actually the online repartitioning is based on table replication (SAP Note 2340450) and "OMR" in this context is the abbreviation for "online-mode replica".

43. What kind of resources are required during repartitioning?

When doing repartitioning you have to consider that significant requirements for specific system resources can exist:

Resource	Details
CPU	Several SAP HANA threads can work on the same repartitioning operation in parallel, so a significant amount of CPU resources can be consumed. The parallelism can be controlled via the split_threads parameter as described in SAP Note 2222250.
Memory	During repartitioning both the old and the new table is held in memory and the new table is initially set up without compression, so the memory requirements for the table can be factors higher until both repartitioning and delta merge and optimize compression is finished.
Disk	The increased (intermediate) table data is also persisted to disk, so also the used space in the data area increases. Once all operations are finished, the additional disk space is released but kept allocated. It can be reclaimed via RECLAIM DATAVOLUME (SAP Note 2400005).

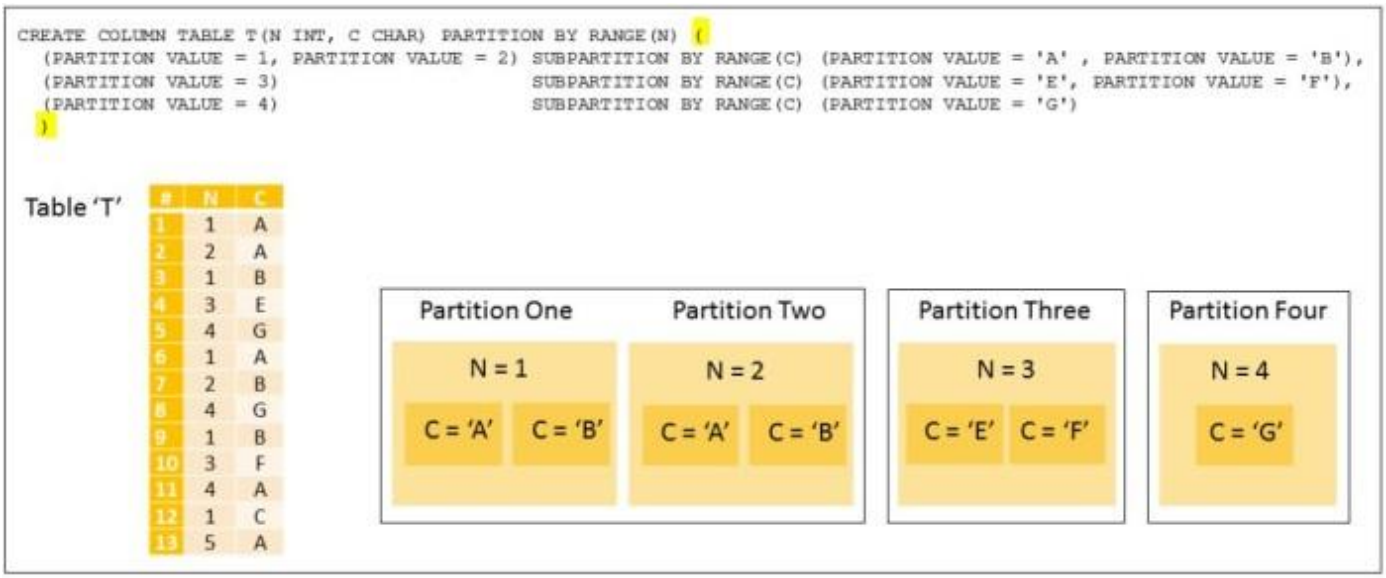
Log	All data that is inserted in the new table is written to the redo logs, so a significant amount of redo logs can be generated during repartitioning. This may have impact on the overall system performance in case of bottlenecks in the I/O area (SAP Note 1999930) or system replication (SAP Note 1999880).

Be aware that after repartitioning typically a delta merge (SAP Note 2057046) and an optimize compression (SAP Note 2112604) is performed which similarly to repartitioning itself can consume a lot of system resources.

44. What is heterogeneous partitioning?

The normal second-level range partitioning schema applies the same second level specification to all first level partitions. For some scenarios a flexible heterogeneous second-level range partitioning schema can be useful with different second-level range specifications for each first level partition. Heterogeneous partitioning is available starting with SAP HANA 2.0 SPS 03. See the Heterogeneous Partitioning section of the SAP HANA online help for more details.

Example: (Heterogeneous 2-level range-range partitions)



The deciding factor in the heterogeneous partitioning syntax are the brackets highlighted yellow. Alternatively you can identify heterogeneously partitioned tables by the term "CompactPartitionSpec2" being part of the partition specification (e.g. visible in column PARTITION_SPEC of viewTABLES).

For usual partitioning the first level partitioning columns need to be part of the primary key (see "Which partition types exist?"). This restriction doesn't apply for heterogeneous partitioning. So in specific cases heterogeneous partitioning syntax can be used just for the purpose of avoiding the primary key column limitation.

Example:

```
CREATE TABLE AAA (A INT PRIMARY KEY, B INT) PARTITION BY RANGE (B) (PARTITION OTHERS)
-- 435: invalid expression: partition column not in primary key columns: B

CREATE TABLE AAA (A INT PRIMARY KEY, B INT) PARTITION BY RANGE (B) ((PARTITION OTHERS))
-- works fine
```

45. Which types of partition pruning exist?

Partition pruning makes sure that only partitions are accessed that are relevant for the query. The following

pruning types are distinguished:

Pruning type	Details
Static partition pruning	Static partition pruning is based on the partition definition. The query optimizer analyzes the WHERE clause of queries to determine whether or not the filters match the given partitioning specification of a table. If a match is found, it may be possible to target only the specific partitions that hold the data and thus reduce the number of partitions being queried.
Dynamic partition pruning	Dynamic partition pruning is content-based and can be applied to historical partitions in context of data aging (SAP Note 2416490) or Native Storage Extension (SAP Note 2799997). This type of pruning takes place at run time but the decision about which partitions to load is based on pre-calculated column statistics of the data in selected columns. Based on the existence of statistics valid for partition pruning (which must be up to date at the time the query is processed), query evaluation can determine if it is necessary to read the partitioned column of data and load it into memory. Dynamic partition pruning can also be of advantage in context of partitioned column store tables (independent of NSE or data aging), e.g. in order to reduce accesses to remote partitions in scale-out scenarios. For homogeneously partitioned tables dynamic pruning is available for all engines. For the OLAP engine it has to be activated explicitly with the following parameter setting: indexserver.ini -> [query_mediator] -> use_dynamic_pruning = true Heterogeneous partitioning is only supported by HEX (SAP Note 2570371). Only the following datatypes are supported for dynamic partition pruning: VARCHAR (strings with numeric content) INTEGER DECIMAL DATE In addition data statistics (SAP Note 2800028) of type SIMPLE VALID FOR DATA DEPENDENCY need to be defined on relevant columns: CREATE STATISTICS ON "<table>" ("<column>") TYPE SIMPLE VALID FOR DATA DEPENDENCY You can use SQL: "HANA_SQL_Statistics_DataStatistics" (SAP Note 1969700) in order to check for existing statistics. The statistics become invalid as soon as a data modification happens. They will be recreated during a delta merge (SAP Note 2057046) of the table or manually with the following command: REFRESH STATISTICS ON "<table>"

See Static and Dynamic Partition Pruning for more information.

46. How can database requests be restricted to a subset of partitions of a table?

It is possible to limit database requests to only specific partition IDs by specifying one or more partition IDs via the following PARTITION clause:

```
PARTITION(<part_id1>, ..., <part_idN>)
```

The partition IDs can e.g. be determined via output column PART of *SQL: "HANA_Tables_ColumnStore_Partitions"* available via SAP Note 1969700 or via column PART_ID of various SAP HANA monitoring views.

Example: (selection of records from MY_PART_TABLE partitions 2, 5 and 8)

```
SELECT * FROM MY_PART_TABLE PARTITION (2, 5, 8)
```

Keywords

Record count
Table growth
Billion
HASH

ROUNDROBIN
RANGE
CHECK_PARTITIONING
CHECK_PARTITIONING_DATA

NUM_HDB_HASH_PAR

Other Components

Component	Description
HAN-DB	SAP HANA Database

This document refers to

SAP Note/KBA	Title
3142798	Re-partitioning of time selection (RANGE) table is required to change partitioning definition
2890332	Table re-partitioning failed with error "Data receive failed"
2845463	CKMLCR Table reached 2B limit in HANA
2800028	FAQ: SAP HANA Optimizer Statistics
2800007	FAQ: SAP HANA Temporary Tables
2799997	FAQ: SAP HANA Native Storage Extension (NSE)
2600030	Parameter Recommendations in SAP HANA Environments
2570371	FAQ: SAP HANA Execution Engine (HEX)
2528241	Partitioning of tables replicated by SLT to SAP HANA database
2470289	FAQ: SAP HANA Non-Uniform Memory Access (NUMA)
2416490	FAQ: SAP HANA Data Aging in SAP S/4HANA
2400005	FAQ: SAP HANA Persistence

SAP HANA Database Partitioning by Irshad Rather

2396601	How to change the partitioning information for a table during the DB migration to HANA
2380176	FAQ: SAP HANA Database Trace
2366291	FAQ: SAP HANA Executed Statements Trace
2340450	FAQ: SAP HANA Table Replication
2313619	How-To: Generating and Evaluating SAP HANA Call Stacks
2222277	FAQ: SAP HANA Column Store and Row Store
2222250	FAQ: SAP HANA Workload Management
2222217	How-To: Troubleshooting SAP HANA Startup Times
2222200	FAQ: SAP HANA Network
2222110	FAQ: SAP HANA Load History
2220627	FAQ: SAP HANA LOBs
2169283	FAQ: SAP HANA Garbage Collection
2160391	FAQ: SAP HANA Indexes
2143736	FAQ: SAP HANA Table Distribution for BW
2127458	FAQ: SAP HANA Loads and Unloads
2119087	How-To: Configuring SAP HANA Traces
2114710	FAQ: SAP HANA Threads and Thread Samples
2112604	FAQ: SAP HANA Compression
2109355	How-To: Configuring SAP HANA Inverted Hash Indexes
2100040	FAQ: SAP HANA CPU
2100009	FAQ: SAP HANA Savepoints
2081591	FAQ: SAP HANA Table Distribution
2057046	FAQ: SAP HANA Delta Merges
2014562	FAQ: SAP HANA LT Replication Server (SLT)
2000002	FAQ: SAP HANA SQL Optimization
1999998	
1999997	FAQ: SAP HANA Memory
1999993	How-To: Interpreting SAP HANA Mini Check Results
1999930	FAQ: SAP HANA I/O Analysis

SAP HANA Database Partitioning by Irshad Rather

1999880	FAQ: SAP HANA System Replication
1998599	How-To: Analyzing high SAP HANA Memory Consumption due to Translation Tables
1986747	How-To: Analyzing internal Columns in SAP HANA Column Store
1910188	How to handle HANA Alert 27: ‘Record count of column-store table partitions’
1910140	How to Handle Alert 20 ‘Table growth of non-partitioned column-store tables’
1909763	How to handle HANA Alert 17: ‘Record count of non-partitioned column-store tables’
3139140	Partitioning for material ledger tables MLDOC and MLDOCCCS
2910004	Increase Disk Usage After Performing DDL Operation on Table With Packed LOB Column
2874176	Parameter Recommendations for Online Table Operations
2873607	Potential Performance Impact by Concurrent Online DDLs
2871405	Elevated Memory Consumption During Online Table Repartitioning
2849678	Performance SAP Transactional Banking for SAP S/4HANA - partitioning
2784715	SWPM: Using R3load-based migration for SAP System on HANA when tables contain more than 2 billion rows
2779173	
2722355	Partitioning of database tables in S4FPSL
2700982	Table Partitioning for Unified Demand Forecast and Demand Data Foundation (for Multiple-Host Scenarios)
2524869	Partitioning of database tables for Smart AFI
2436619	Inconsistency After Repartitioning a Table Using Inverted Hash Indexes
2418299	SAP HANA: Partitioning Best Practices / Examples for SAP Tables
2334091	SAP BW/4HANA: Table Placement and Landscape Redistribution
2259038	S/4HANA: Partitioning of table MATDOC
1969700	SQL Statement Collection for SAP HANA
1908075	SAP BW on SAP HANA: Table Placement and Landscape Redistribution
	Characteristics with Enhanced Master Data Update
	Static and Dynamic Partition Pruning
	Partitioning Limits
	Heterogeneous Partitioning
	Data Transfer Immediate Storage (DTIS)

SAP HANA Database Partitioning by Irshad Rather
