A look at Partitioning

Vikas Sawhney
Long Island Oracle Users Group
March 30, 2006

What is Partitioning?

- **Partitioning** is a method by which tables, indexes, and index-organized tables can divided into multiple smaller chunks.
- Partitioning is physical change; thus application code does not be changed to accommodate this. Logically the object is still one object; thus direct application code modification is not required to implement partitioning .
- That is not to say that the application code should be revised to take benefit like accessing partition directly where possible.
- A detailed analysis needs to be done prior to implementing partitioning on a object as it could have some serious performance impact. Make sure you test, test, and test again before making changes to your production environment.
- Test all process (insert,update,delete,select) involving the interested table to be partitioned.

Who Partitions?

- Deciding on what and how to partition is both a Developer and DBA job.
- A good of understanding needs to be known about how the data is utilized within Oracle. How data is loaded and queried.
- A great of care needs to done in selection of the type of partitioning along with the partition key.
- Poor selection of partition or partition key could lead to poor dml and ddl performance.
- Always test, test, and test again prior to implementing in production.

Why partition?

- Partitioning has various benefits such as ease of management of data, increase overall performance, better availability of the system, and enhances security.

- --–Makes management of data more controllable
  -- Bring a section of data offline and the users don't miss a beat
  -- A spew of tool to better organize partition attributes
  (ADD, COALESCE, DROP EXCHANGE MERGE, MOVE, SPLIT TRUNCATE)
- -- Failure are less likely to impact the entire object.
- -- Allow better scalability of the data
- -- Allows for a good method to archive old data out of the system
- (Sliding window Technique)
- -- Allows faster access to the data
- -- Allows mechanism to better control hot spot in data acquisition
- -- Allow more finer grained control of sensitive data

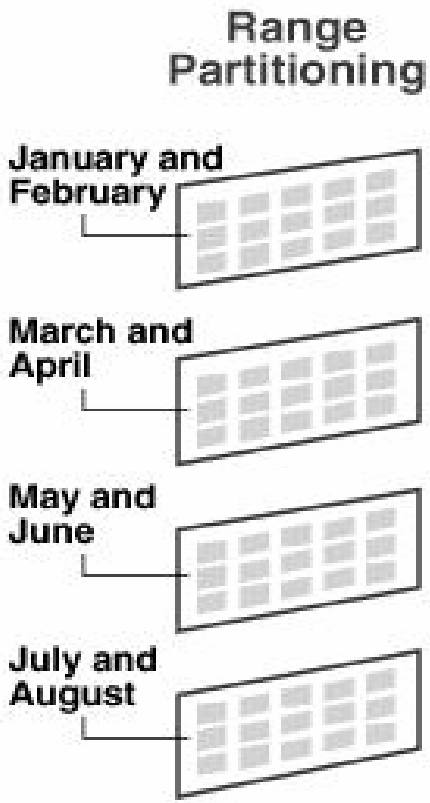
Partition Methods with Version Matrix

Partitioning Method	Version
Partitioned View	7.3.2 – Present Version
Range	8.0.4-Present Version row movement not available
Hash	8.1.5 - Present Version
Composite (Range and Hash)	8.1.5 - Present Version
List	9.0.1 - Present
Composite (Range and List)	9.0.1 - Present

Partitioned Views

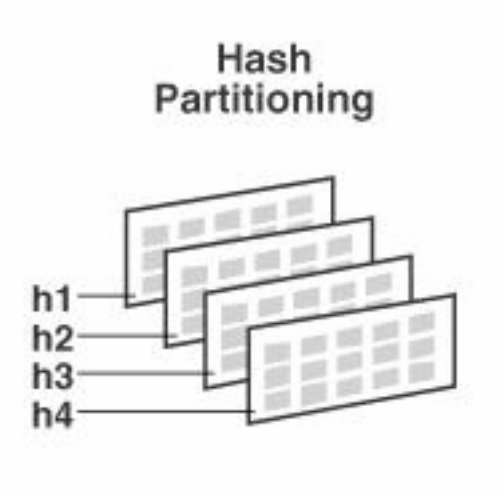
- 1. Set PARTITION_VIEW_ENABLED=TRUE
  2. Set OPTIMIZER_MODE=CHOOSE in the init.ora , rule based optimizer does not work
  3. Set COMPATIBLE=7.3.2 or higher
  4. Make sure all the tables have correctly defined CHECK constraints.
  5. Make sure all columns and all indexes of all tables in the partition view are identically defined. Columns must be of the same size and type; indexes must be on the same columns and be of the same type (regular, binary, bitmapped or reverse).
- 6. ANALYZE all tables and indexes.
- 7. Make sure the query uses only a simple predicate, such as an equality or BETWEEN against literals. The optimizer won't recognize partition views if WHERE IN, OR or functions are used.
- 8. If the partitioning column is a CHAR type, this could be Bug 366589, fixed in 7.3.3. On most platforms, the fix is also inlcuded in patch release 7.3.2.2, available from support. The workaround is to make the column a VARCHAR2 type and trim off any null padding in the data.
- 9. Oracle Corporation recommends that you use partitioned tables (available since Oracle8) rather than partition views. In general, partition views should not be used and are supported for compatibility only. Oracle Corporation intends to desupport them in a future release. (Doc ID: Note:149016.1)

Range Partitioning



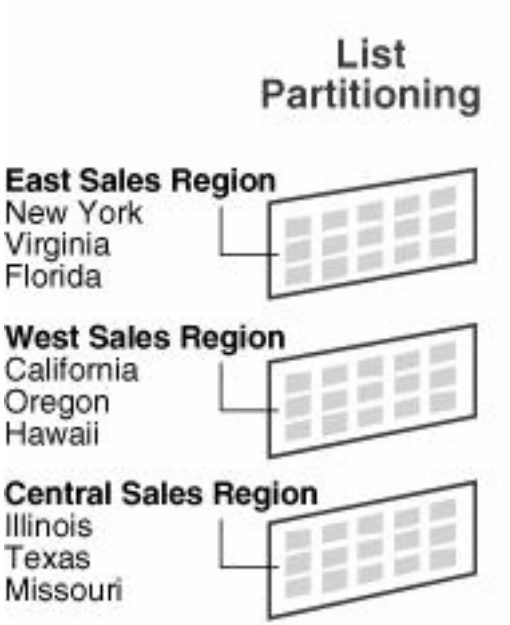
- Each partition has a VALUES LESS THAN clause, which specifies a noninclusive upper bound for the partitions. Any values of the partition key equal to or higher are added to the next higher partition.
- Each partition is defined by a Partition Boundary. The boundary basically limits the scope of the data stored within a partition.
- A MAXVALUE literal can be defined as the catch all partition where no particular partition has defined for the dataset. MAXVALUE can represent an infinite value for the partition key this includes null value.
- Most common use for Range partition is where you know a set of boundaries for values
- If MAXVALUE is defined then no new partition can be added.
- Maintenance Options available are:
  Add, Drop, Exchange, Merge, Modify Attributes, Modify Real Attributes (can allocate and deallocate extents, mark local index partitions UNUSABLE, or rebuild local indexes that have been marked UNUSABLE)

Hash Partitioning



- Data is arranged by column in the amount of partition you specific.
- User no control over the distribution of data because the data is distributed over the various partitions using the system hash function
- The best case scenario for this is that you can spread the data out on to different disks to better distribute overall disk I/O
- The data can be equally balanced in terms of rows by adding or deleting partitions from the object.
- The physical size of partition can be more finely controlled.
- Row movement needs to enabled to allow data to be moved from one partition to another.
- The concepts of splitting, dropping or merging partitions do not apply to hash partitions. Instead, hash partitions can be added and coalesced.
- A good choice of a hash partitioning key is a column that has a great many unique values and is commonly used as a predicate in queries.

List Partitioning



- A close cousin of range partitioning is list partition. Here you can more clearly define literal boundaries for partition.
  Sort for like an in list in sql:
  select table_columns from table_x
  where column1 in ('a','b','c')
- The advantage of list partitioning is that you can group and organize unordered and unrelated sets of data in a natural way.
- Should have few distinct row values as each value should be managed in the list
- DEFAULT partition key is created as the catch all partition for data set that may not have not defined.
- DEFAULT can be useful in catching value you may not have originally thought of.

Partitioning Operations

Maintenance Operation	Range	Hash	List	Composite: Range/Hash	Composite: Range/List
Adding Partitions	ADD PARTITION	ADD PARTITION	ADD PARTITION	ADD PARTITION MODIFY PARTITION...ADD SUBPARTITION	ADD PARTITION MODIFY PARTITION...ADD SUBPARTITION
Coalescing Partitions	n/a	COALESCE PARTITION	n/a	MODIFY PARTITION... COALESCE SUBPARTITION	n/a
Dropping Partitions	DROP PARTITION	n/a	DROP PARTITION	DROP PARTITION	DROP PARTITION DROP SUBPARTITION
Exchanging Partitions	EXCHANGE PARTITION	EXCHANGE PARTITION	EXCHANGE PARTITION	EXCHANGE PARTITION EXCHANGE SUBPARTITION	EXCHANGE PARTITION EXCHANGE SUBPARTITION
Merging Partitions	MERGE PARTITIONS	n/a	MERGE PARTITIONS	MERGE PARTITIONS	MERGE PARTITIONS MERGE SUBPARTITIONS

Partitioning Operations

Modifying Default Attributes	MODIFY DEFAULT ATTRIBUTES	MODIFY DEFAULT ATTRIBUTES	MODIFY DEFAULT ATTRIBUTES	MODIFY DEFAULT ATTRIBUTES MODIFY DEFAULT ATTRIBUTES FOR PARTITION	MODIFY DEFAULT ATTRIBUTES MODIFY DEFAULT ATTRIBUTES FOR PARTITION
Modifying Real Attributes of Partitions	MODIFY PARTITION	MODIFY PARTITION	MODIFY PARTITION	MODIFY PARTITION MODIFY SUBPARTITION	MODIFY PARTITION MODIFY SUBPARTITION
Modifying List Partitions: Adding Values	n/a	n/a	MODIFY PARTITION...ADD VALUES	n/a	MODIFY SUBPARTITION... ADD VALUES
Modifying List Partitions: Dropping Values	n/a	n/a	MODIFY PARTITION...DROP VALUES	n/a	MODIFY SUBPARTITION... DROP VALUES
Modifying a Subpartition Template	n/a	n/a	n/a	SET SUBPARTITION TEMPLATE	SET SUBPARTITION TEMPLATE
Moving Partitions	MOVE PARTITION	MOVE PARTITION	MOVE PARTITION	MOVE SUBPARTITION	MOVE SUBPARTITION

Partitioning Operations

Renaming Partitions	RENAME PARTITION	RENAME PARTITION	RENAME PARTITION	RENAME PARTITION RENAME SUBPARTITION	RENAME PARTITION RENAME SUBPARTITION
Splitting Partitions	SPLIT PARTITION	n/a	SPLIT PARTITION	SPLIT PARTITION	SPLIT PARTITION SPLIT SUBPARTITION
Truncating Partitions	TRUNCATE PARTITION	TRUNCATE PARTITION	TRUNCATE PARTITION	TRUNCATE PARTITION TRUNCATE SUBPARTITION	TRUNCATE PARTITION TRUNCATE SUBPARTITION