Chapter 4 - Partition Primary Index (PPI) Tables

“I saw an angel in the marble and carved until I set him free.”

- Michelangelo

TheConcept behind Partitioning a Table

1. Each Table in Teradata has a Primary Index unless it is a NoPI table.

2. The Primary Index is the mechanism that allows Teradata to physically distribute the rows of a table across the AMPs.

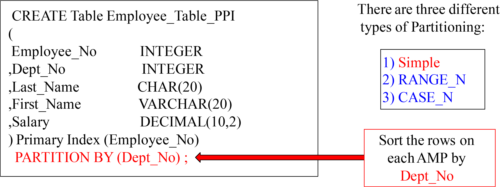
3. AMPs Sort their rows by the Row-ID, so the system can perform a lightning fast Binary Search since the rows are in Row-ID Order.

4. Partitioning merely tells the AMP to sort its tables’ rows by the Partition first, but then sort the rows by Row-ID within the partition.

5. Partitioning queries will involve all AMPs, but partitioned tables are designed to prevent FULL Table Scans.

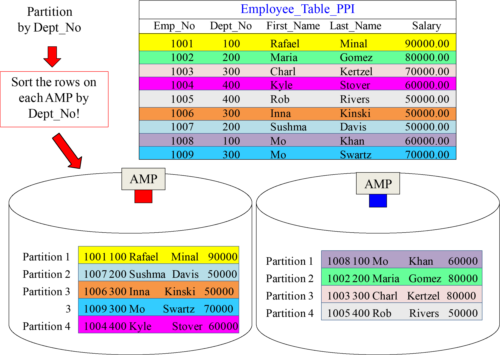
The basic concepts of Partitioning are above, so implant these in your mind.

Creatinga PPI Table with Simple Partitioning

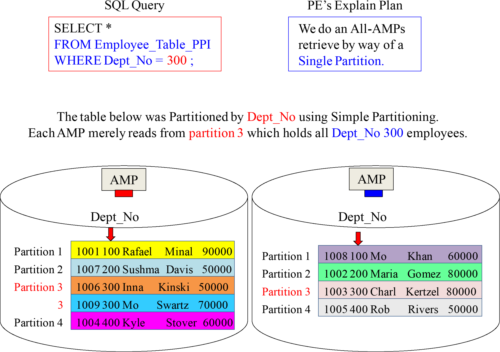


A PPI Table is a table with a Partition on it. A PPI table has the AMPs sort (order) the rows on the table by the Partition. This allows for people to avoid Full Table Scans. This is an example of Simple PPI. Each AMP will sort the rows they own by Dept\_No.

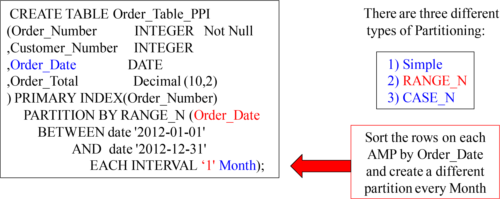
AVisual Display of Simple Partitioning



AnSQL Example that explains Simple Partitioning

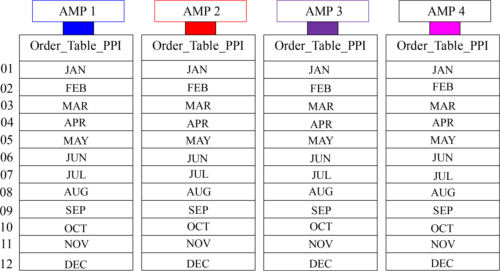


Creatinga PPI Table with RANGE\_N Partitioning per Month



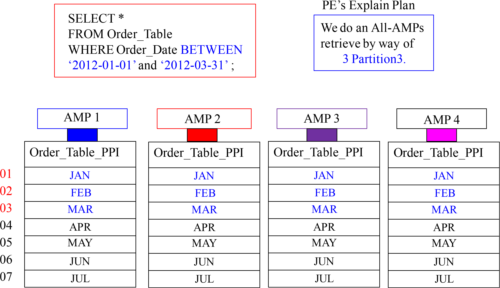
You can order everything by day. You can also order everything by week, by month, and by year. You have many options to choose from.

AVisual of One Year of Data with Range\_N per Month



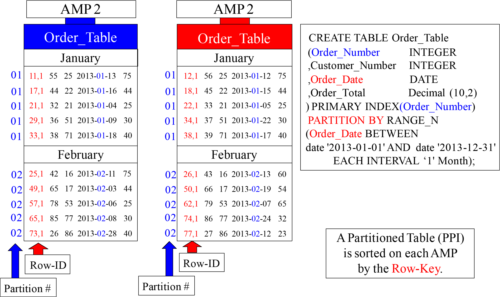
Each AMP sorts their rows by Month (of Order\_Date).

AnSQL Example explaining Range\_N Partitioning per Month



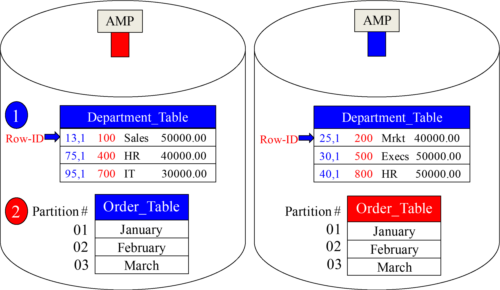
The above query wants to see orders in the first quarter, so each AMP reads 3 partitions.

APartition # and Row-ID = Row Key



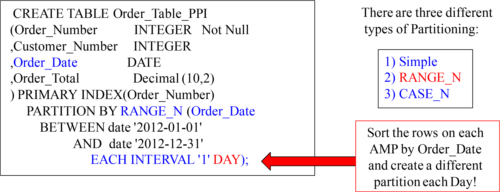
Above, is Partition # 1 (January) and Partition # 2 (February). Inside each partition, you can see the Row-ID. The Partition # combined with the Row-ID is called the Row Key.

AnAMP Stores its Rows Sorted in only Two Different Ways



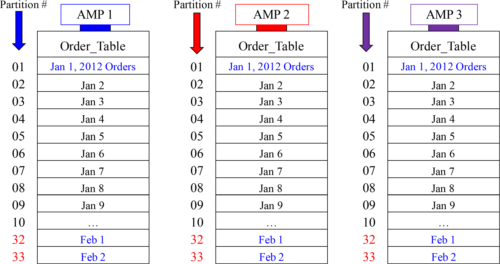
An AMP stores its rows sorted by the Row-ID or the Row Key. A normal table sorts by the Row-ID and a Partitioned Table (PPI table) sorts on the Row Key (Partition # + Row-ID). You will soon find out exactly why this is done and the true meaning of Physical Database Design! These are the two ways an AMP sorts their data on disk.

Creatinga PPI Table with RANGE\_N Partitioning per Day



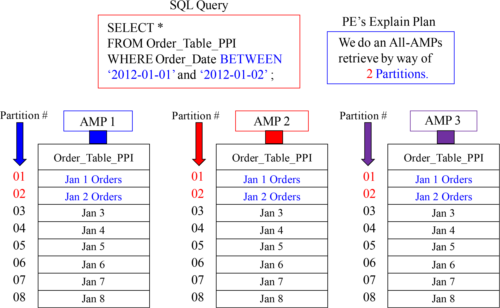
The above syntax represents a RANGE\_N partition with a partition each day.

AVisual of Range\_N Partitioning Per Day



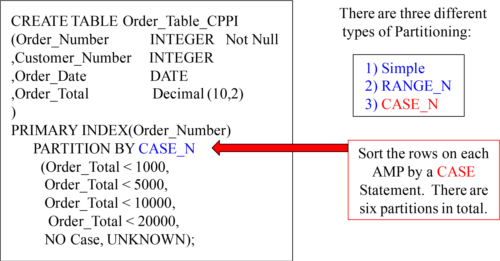
There would be 365 partitions in total for an entire year of Partitioning per Day. Each AMP holds the orders assigned to them in daily partitions on the day of the order date.

AnSQL Example that explains Range\_N Partitioning per Day



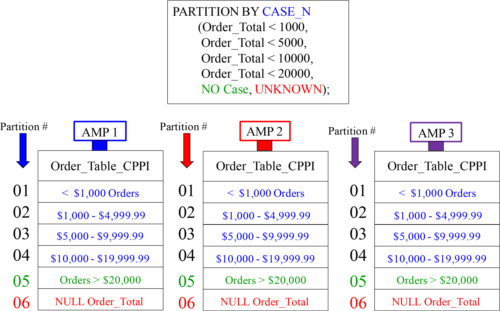
Each AMP holds the orders assigned to them in daily partitions on the day of the order.

Creatinga PPI Table with CASE\_N



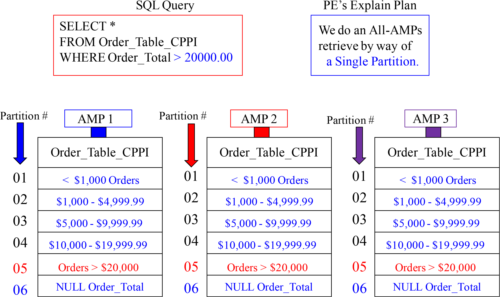
The above syntax represents the CASE\_ N partitioning. If an Order\_Total is < 1000, it will go into Partition 1. If an Order\_Total is between 1000 and 4999.99, it will go in Partition 2. The NO Case partition is for anything falling through and the UNKNOWN partition is for NULL values in the Total.

AVisual of Case\_N Partitioning



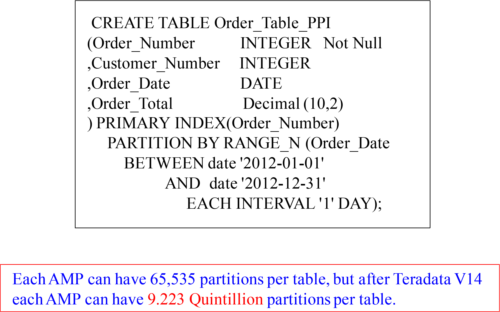
There are six partitions for this table. Orders go into partitions based on Order\_Total.

AnSQL Example that explains CASE\_N Partitioning



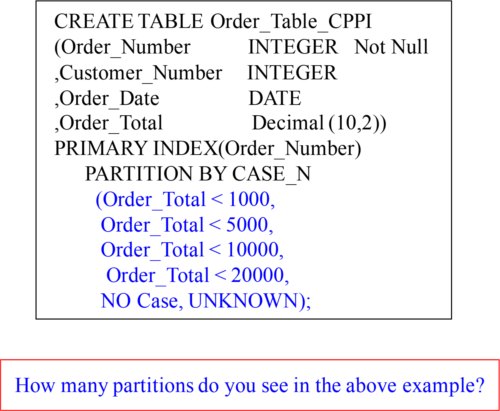
All AMPs retrieve, but each only reads partition 5 which is a mere sliver of the disk.

Numberof PPI Partitions Allowed



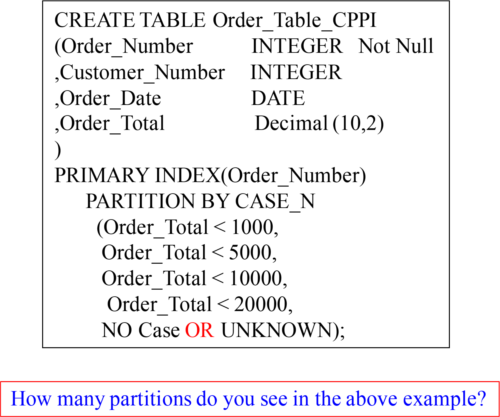
Teradata Tables can have 65,535 partitions max, but after V14, it is 9.223 Quintillion.

Howmany partitions do you see?



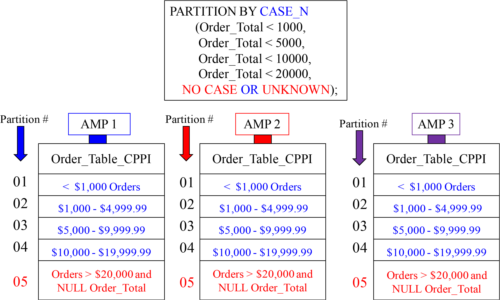
There are six partitions in the above example. Partition 1 is for Order\_Totals < 1000 and Partition 2 is for Order\_Totals < 5000, etc. The NO Case partition is for anything falling through the CASE statement like an ELSE so it is for any Order\_Total 20,000 or greater. The UNKNOWN is for Order\_Totals that are Null.

NOCASE and UNKNOWN Partitions Together



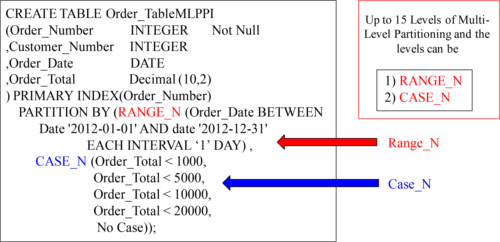
There are five partitions in the above example. Partition 1 is for Order-Totals < 1000 and Partition 2 is for Order\_Totals < 5000, etc. The NO Case partition and the UNKNOWN partition are combined. So now, any Order\_Total that is 20,000 or greater or any Order\_Total that is NULL will be in the 5th partition together.

AVisual of Case\_N Partitioning



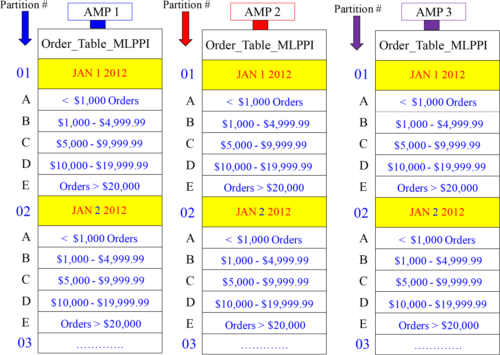
There are five partitions for this table because we combined the NO CASE and UNKNOWN partitions together. Orders are placed in partitions based on Order\_Total.

Multi-Level Partitioning Combining Range\_N and Case\_N

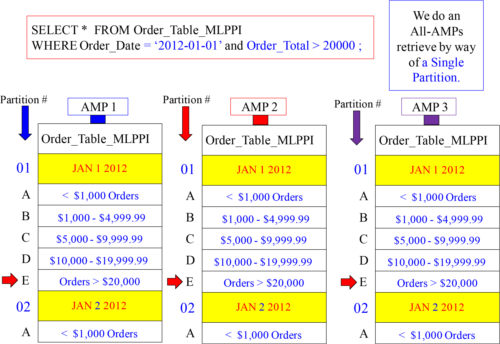


This type of partitioning is called a MULTI-LEVEL PPI. It is a Partition within a Partition. The top partition is the only one you can ALTER so keep that in mind. With Multi-Level partitioning, you can combine up to 15 Case\_N or Range\_N partitions within partitions. NO Simple Partitioning can be used in Multi-Level Partitioning.

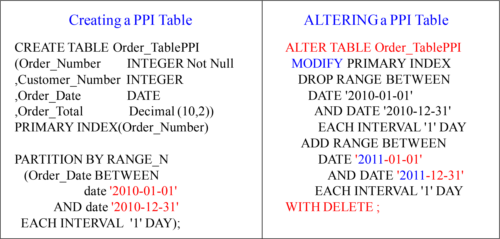
AVisual of Multi-Level Partitioning



TheSQL on a Multi-Level Partitioned Primary Index

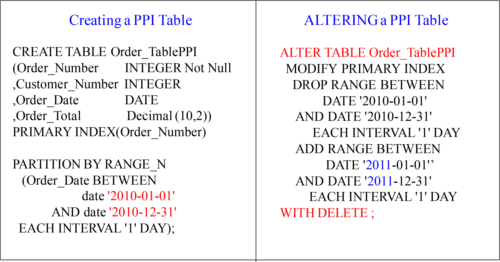


Alteringa PPI Table to Add or Drop Partitions



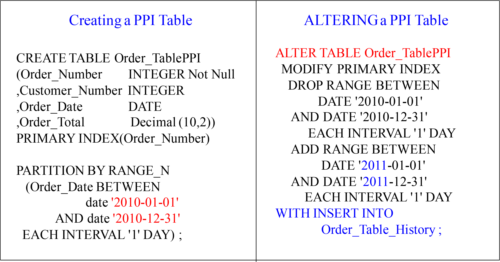
What will happen to the 2010 data after the ALTER?

Deletinga Partition



The 2010 data after the ALTER will be deleted from the table.

Deletinga Partition and saving its contents



The 2010 data after the ALTER will be deleted from the table, but the deleted data is stored in Order\_Table\_History.