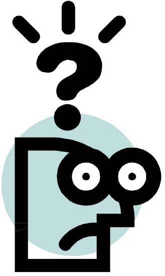
CALCULATE RANGE PARTITIONS USING *NTILE*

Range partitioning is a common way to setup table partitions. With this method, you sort the data into different buckets, corresponding to the desired partitions. The partitions don't have to be exactly evenly sized, but for performance reasons, each partition should have roughly the same number of rows.



So What's The Problem?

Sometimes, the data distribution is lumpy, complicating the partition endpoints. You might inadvertently create some partitions with millions of rows, while others could be empty. If you setup the partitions badly, you really won't get the full benefit of partitioning.

Beginning in Oracle 8i, there is a good way to accurately group data into various buckets. We use the analytic function ***ntile*** to group the data into buckets automatically. Then, we use this information to build the necessary syntax to create our partitions.

Our Test Case

In our sample case, let's assume we have a large table called *EMP_MASTER* that contains 10 million rows. We have decided to range partition[1] the EMP_MASTER table into 512 partitions. Oracle syntax requires us to specify the endpoints for each of the 512 partitions. The partition clause used in the *CREATE TABLE* statement looks like this:

```
PARTITION [name] VALUES LESS THAN (nnnnnn),
```

Instead of guessing at the above value, let's use the *ntile* analytic function to divide the data into 512 partitions. This will tell us the end point for each bucket.

The Process

First, we'll create a staging table that assigns each row in EMP_MASTER to one of 512 bins. This is trivial using *ntile*, as shown below. Note that we have specified 512 buckets:

```
Create Table Chris_Bins Nologging As
Select Emp_No, Ntile(512) Over (Order By Emp_No)
As Bin From Emp_Master;
```

Let's sample the contents of our staging table:

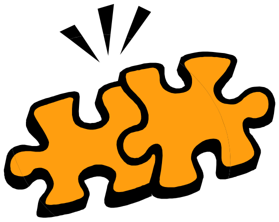
[1] In some cases, you can use hash partitioning, which avoids the need to specify endpoints. Sometimes, however, you will be forced to use range partitioning because of Oracle limitations. One such case would be partitioning an index *differently* than the underlying table.

CALCULATE RANGE PARTITIONS USING *NTILE*

```
    Select * From Chris_Bins Where Rownum < 5

    EMP_NO                             BIN
    ------------------------- ----------
    000000132350                       1
    000002291208                       1
    000002291703                       1
    000002781037                       1
```

We see that ntile has assigned the first four rows to the first bin. All of the other rows have been similarly assigned to one of the 512 bins.



Find the Endpoints

Now that we've got every row assigned to a bucket, we can easily find the endpoints of each bin. The endpoint is just the maximum value for each bin:

```
  Select Bin, Max(EMP_NO) From Chris_Bins
  Group By Bin
  Order By 1;


      BIN MAX(EMP_NO)
--------- -----------------------
        1 008421401802
        2 009800915538
        3 011132588052
        4 011545329198
        5 012333004099
        6 012333111964
        7 012495401040

        *     *
        *     *

       509 833557002032
       510 851269000689
       511 881780744786
       512 9999999999901
```

We can use this information in the "Less Than" values when creating the partitions. Recall that each partition is of the form:

```
      PARTITION [name] VALUES LESS THAN  (VALUE) ,
```

Generating the Code

We can easily create a SQL script to generate the 512 clauses. For each partition name, I've decided to use the form PARTn. Here's the SQL:

```
Select ‘PARTITION PART’ ||bin || ‘ VALUES LESS THAN (‘||max(EMP_NO)||’),’
```

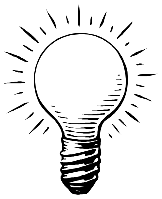
```
from Chris_Bins
Group by bin
Order by bin;
```

The output will look like this:

```
    PARTITION PART1 VALUES LESS THAN (008421401802),
    PARTITION PART2 VALUES LESS THAN (009800915538),
    PARTITION PART3 VALUES LESS THAN (011132588052),
    PARTITION PART4 VALUES LESS THAN (011545329198),
    etc
```

Simply spool the output and you'll have the range clause for each partition. You may also want to add information for which tablespace to use, storage, etc.

Don't Forget!



Finally, remember to substitute the keyword *MAXVALUE* for the 512th bin. In our test case, the syntax for this bin would look like this:

```
    PARTITION PART512 VALUES LESS THAN (MAXVALUE)
```

Chris Lawson is a performance specialist living in the San Francisco Bay area. He specializes in tuning *Retek* applications, and also publishes the online newsletter, *The Oracle Magician*. Chris can be reached at: Chris@OracleMagician.com