Understanding Statistics

- It is possible for the DBA to generate statistics that quantify the data distribution and storage characteristics of tables, columns, indexes, and partitions.
- The cost-based optimization approach uses these statistics to calculate the selectivity of predicates and to estimate the cost of each execution plan.
- ***Selectivity*** is the fraction of rows in a table that the SQL statement's predicate chooses. The optimizer uses the selectivity of a predicate to estimate the cost of a particular access method and to determine the optimal join order.

Histograms

- Histograms is a statistic which we calculate for those columns that contain skewed data.
- **Skewed Data** : values with large variations in number of duplicates.
- The cost-based optimizer can use histograms to get accurate estimates of the distribution of column data.
- A *histogram* partitions the values in the column into bands, so that all column values in a band fall within the same range.
- Histograms provide improved selectivity estimates, resulting in optimal execution plans with nonuniform data distributions.

Types of Histograms

- There are two types of histograms:

1. Height-Based Histograms
2. Value-Based Histograms

Height-Based Histograms :

- Height-based histograms place approximately the same number of values into each range, so that the endpoints of the range are determined by how many values are in that range.
- Only the last (largest) values in each bucket appear as bucket (end point) values.

Height Based Histograms

- Consider that a table's query results in the following four sample values: 4, 18, 30, and 35.
- For a height-based histogram, each of these values occupies a portion of one bucket, in proportion to their size.
- The resulting selectivity is computed with the following formula:

```
S = Height(35)/ Height(4 + 18 + 30 + 35)
```

Value-Based Histograms

- Value-based histograms are created when the number of distinct values is less than or equal to the number of histogram buckets specified.
- In value-based histograms, all the values in the column have corresponding buckets, and the bucket number reflects the repetition count of each value. These can also be known as frequency histograms.
- Consider the same four sample values in the previous example. In a value-based histogram, a bucket is used to represent each of the four distinct values.
- In other words, one bucket represents 4, one bucket represents 18, another represents 30, and another represents 35. The resulting selectivity is computed with the following formula:
- `S = [#rows (35)/(#rows(4) + #rows(18) + #rows(30) + #rows(35))]/ #buckets`

Value-Based Histograms(contd)

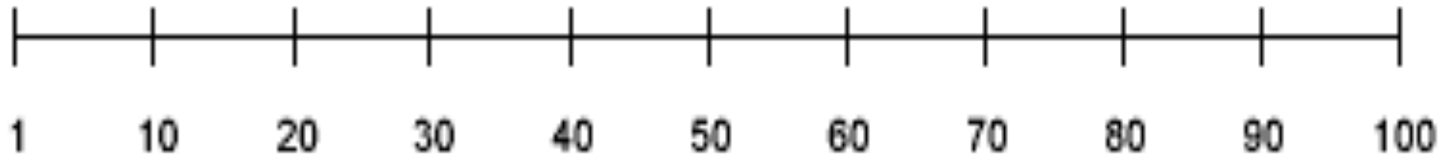
- If there are many different values anticipated for a particular column of table, it is preferable to use the value-based histogram rather than the height-based histogram.
- This is because if there is much data skew in the height, then the skew can offset the selectivity calculation and give a nonrepresentative selectivity value.

Using Histograms

- The cost-based optimizer uses height-based histograms on specified attributes to describe the distributions of nonuniform domains. In a height-based histogram, the column values are divided into bands so that each band contains approximately the same number of values.
- The useful information that the histogram provides is : where in the range of values the endpoints fall.

Using Histograms(contd)

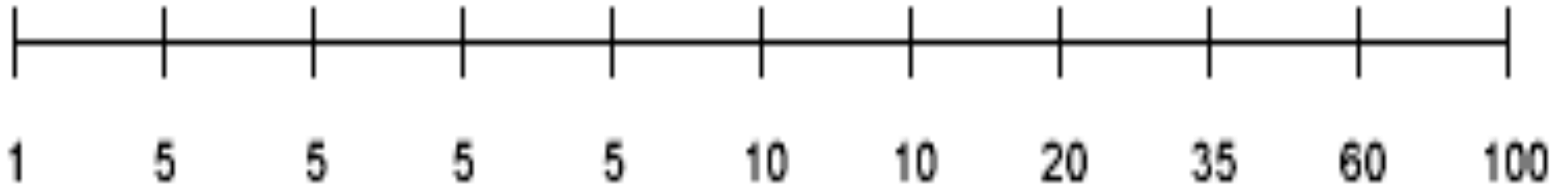
- Consider a column C with values between 1 and 100 and a histogram with 10 buckets. If the data in C is uniformly distributed, then the histogram looks like this, where the numbers are the endpoint values:



- The number of rows in each bucket is one tenth the total number of rows in the table. Four-tenths of the rows have values between 60 and 100 in this example of uniform distribution.

Using Histograms(contd)

- If the data is not uniformly distributed, then the histogram might look like this:



- In this case, most of the rows have the value 5 for the column.
- In this example, only 1/10 of the rows have values between 60 and 100.

When to Use Histograms

- Histograms can affect performance and should be used only when they substantially improve query plans.
- In general, create histograms on columns that are used frequently in `WHERE` clauses of queries and have a highly skewed data distribution.
- For uniformly distributed data, the cost-based optimizer can make fairly accurate guesses about the cost of executing a particular statement without the use of histograms.
- Histograms, like all other optimizer statistics, are static. They are useful only when they reflect the current data distribution of a given column. (The data in the column can change as long as the *distribution* remains constant.) If the data distribution of a column changes frequently, you must recompute its histogram frequently.

When to Use Histograms(contd)

- Histograms are *not* useful for columns with the following characteristics:

1. All predicates on the column use bind variables.
2. The column data is uniformly distributed.
3. The column is not used in `WHERE` clauses of queries.
4. The column is unique and is used only with equality predicates.

Creating Histograms

- You generate histograms by using the `DBMS_STATS` package. You can generate histograms for columns of a table or partition.
- For example, to create a 10-bucket histogram on the `SAL` column of the `emp` table, issue the following statement:

```
EXECUTE DBMS_STATS.GATHER_TABLE_STATS
('scott','emp', METHOD_OPT => 'FOR COLUMNS
SIZE 10 sal');
```

- The SIZE keyword declares the maximum number of buckets for the histogram.
- You would create a histogram on the SAL column if there was an unusually high number of employees with the same salary and few employees with other salaries.
- You can also collect histograms for a single partition of a table.

Viewing Histograms

- You can view histogram information with the following data dictionary views:

1. `DBA_HISTOGRAMS`
2. `DBA_PART_HISTOGRAMS`
3. `DBA_SUBPART_HISTOGRAMS`
4. `DBA_TAB_COL_STATISTICS`

Number of Rows :

View the `DBA_HISTOGRAMS` dictionary table for the number of buckets (in other words, the number of rows) for each column:

- `ENDPOINT_NUMBER`
- `ENDPOINT_VALUE`

Verifying Histogram Statistics

- To verify that histogram statistics are available, execute the following statement against the dictionary's `DBA_HISTOGRAMS` table:

```
SQL> SELECT ENDPOINT_NUMBER, ENDPOINT_VALUE
FROM DBA_HISTOGRAMS
WHERE TABLE_NAME ="SO_LINES_ALL" AND COLUMN_NAME="S2"
ORDER BY ENDPOINT_NUMBER;
```

This returns the following typical data:

```
ENDPOINT_NUMBER ENDPOINT_VALUE
--------------- ---------------
1365      4
1370      5
2124      8
2228      18
```

- Consider the difference between two `ENDPOINT_NUMBER` values, such as 1370 -1365 = 5. This indicates that five values are represented in the bucket containing the endpoint 1370.
- If endpoint numbers are very different, then this implies the use of more buckets, where one row corresponds to one bucket.