DBA

INDEX STATISTICS
INDEX_STATS IN ORACLE

Basic Definition:

Keys and Columns:

A **key** is a set of columns or expressions on which we can build an index. Although the terms are often used interchangeably, indexes and keys are different. **Indexes** are structures stored in the database that users manage using SQL statements. **Keys** are strictly a logical concept. For instance,

CREATE INDEX ord_customer_ix ON orders (customer_id);

In the preceding statement, the customer_id column is the index key. The index itself is named ord_customer_ix.

Type of indexes:

Before we discuss about Index Statistics, I would like to talk about one of the mostly used, default type of index in oracle. That is, **B-Tree Index**. Apart from this, There are many other types of indexes like **Bitmap indexes** (For columns with low distinct value -- cardinality -- like Gender Column containing Male or Female values), **Function-based indexes** (For columns that are either transformed by a function, such as *UPPER()* function, or included in an expression), **Application domain Indexes** (created by user for data in an application specific domain).

B-Tree Indexes:

A **B-tree index** is an ordered list of values divided into ranges. By associating a key with a row or range of rows, B-trees provide excellent retrieval performance for a wide range of queries, including exact match and range searches.

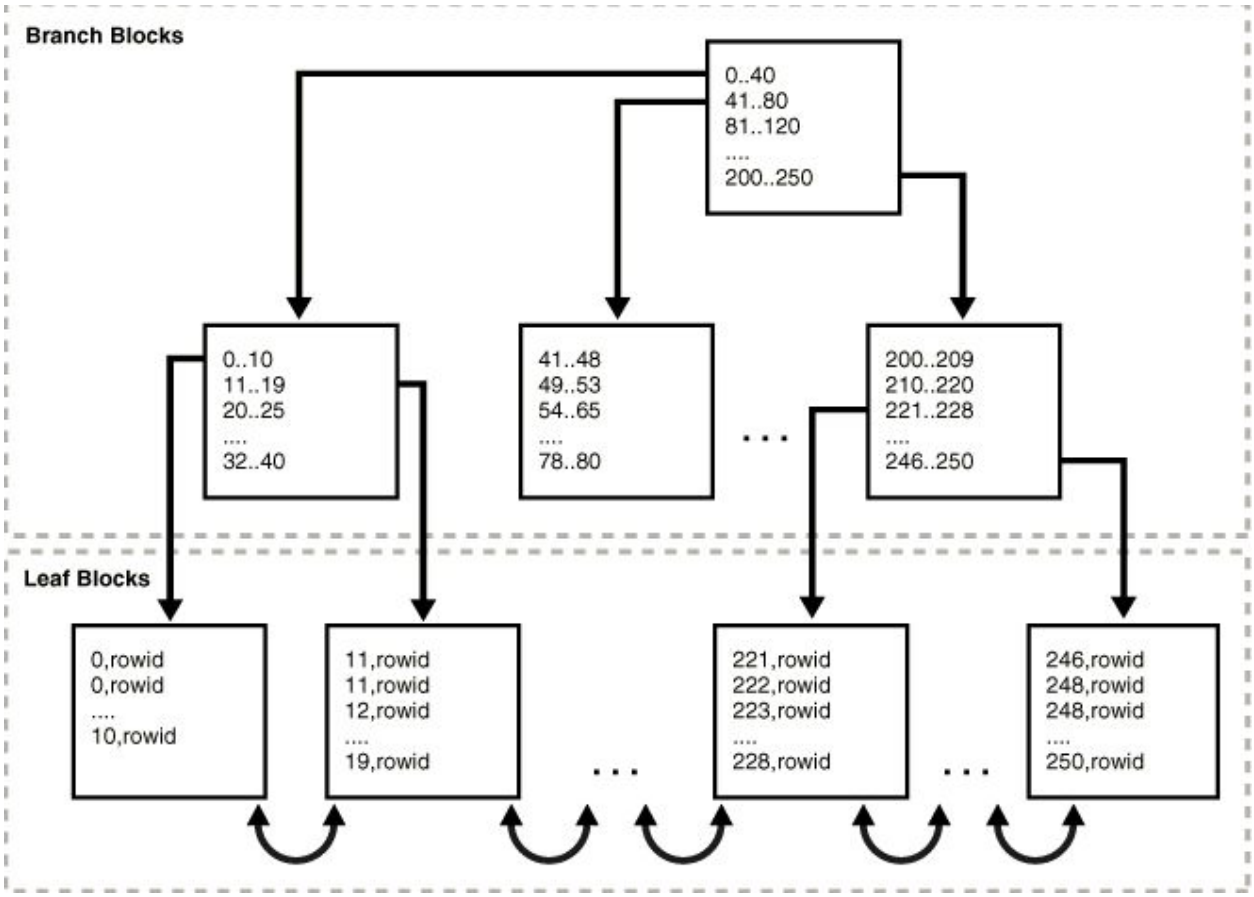

Fig: B- Tree Indexes

There are two types of blocks: **branch blocks** for searching and **leaf blocks** that store values.

The **upper-level** branch blocks of a B-tree index contain index data that points to **lower-leve**l index blocks. the root branch block has an entry 0-40, which points to the leftmost block in the next branch level. This branch block contains entries such as 0-10 and 11-19. Each of these entries points to a leaf block that contains key values that fall in the range.

A B-tree index is balanced because all **leaf blocks** automatically stay at the same depth. Thus, retrieval of any record from anywhere in the index takes approximately the same amount of time. The **height** of the index is the number of blocks required to go from the root block to a leaf block. The **branch level** is the height minus 1.

The **leaf blocks** contain every indexed data value and a corresponding rowid used to locate the actual row. Each entry is sorted by (key, rowid).

Index Statistics:

How can you tell when an index would benefit from being rebuilt? There are two Oracle views that provide index statistics, **DBA_INDEXES** and **INDEX_STATS**. The **DBA_INDEX** view contains statistical information that is placed into the view when the Oracle *ANALYZE INDEX xxx* command is issued. Unfortunately, the **DBA_INDEXES** view does not keep statistics about the internal status of the Oracle indexes, as it was designed to provide information to the *cost-based SQL optimizer*. The Oracle *ANALYZE INDEX xxx VALIDATE STRUCTURE* SQL command can be used to validate the structure for the index. This command creates a single row in a view called **INDEX_STATS**.

To view index statistics,

*Select * from index_stats;*

The Oracle index_stats view will only contain one row. This row is the statistics of the last index whose structure is validated. Therefore, we must perform

" *analyze index <index_name> validate structure*" command to populate the statistics into **index_stats** view.

Also, before we validate structure, Oracle will gather statistics on the index. We can use

DBMS_STATS.GATHER_INDEX_STATS(<owner_name>,<index_name>) or *ANALYZE INDEX <index_name>* compute statistics to gather statistics before we validate.

Note: The latter *ANALYZE INDEX<index_name> COMPUTE STATISTICS* is deprecated and ,in future, may be removed at anytime. So oracle recommends to use the former one.

Oracle indexes perform two basic operations as the index expands to hold more keys. Oracle's version of b-tree indexing uses an algorithm where each index node may contain many index keys. As new key values are added to the index, Oracle must manage the configuration of each index node. Oracle index nodes are managed with two operations; splitting and spawning.

Splitting-- This is the term used to describe what happens when an index node is filled with keys and a new index node is created at the same level as the full node. Splitting widens the b-tree horizontally.

Spawning-- This is the term used to describe the process of adding a new level to an index. As a new index is populated, it begin life as a single level index. As keys are added, a spawn take place and the first level node re-configures itself to have pointers to lower-level nodes. It is important to understand that spawning takes place at specific points within the index, and not for the entire index. For example, a three level index may have a node that experiences heavy insert activity. This node may spawn a fourth level without all of the other level three nodes spawning new levels.

The index_stats view contains information about the internal structure of the b-tree index that can be useful when determining whether or not to rebuild the index. The following columns of index_stats are especially useful:

height - This columns refers to the maximum number of levels encountered within the index. An index may have 90% of the nodes at 3 levels, but excessive splitting and spawning in one area of the index may have caused some nodes to have move then 3 levels. Whenever the value of height is more than three, you may benefit from dropping and re-creating the index. Oracle indexing will not spawn a fourth level on a clean rebuild until more than ten million kets have been added to the index.

del_lf_rows-- This columns refers to the number of leaf rows that have been deleted from the index. This occurs when heavy index update activity occurs within the index tree, and indicated that the index will benefit from being dropped and re-created.

distinct_keys-- This indicates the number of distinct key values in the index. This is called the cardinality of the index, and values less than 20 are candidates for being re-created as bitmapped indexes.

most_repeated_key-- This column counts the number of times that the most frequent key value in a non-unique index appears in the b-tree.

LF_ROWS : It is the total number of leaf rows (value in the index).

LF_BLKS : It is the total number of blocks in B-Tree.

DEL_LF_ROWS : It is the total number of deleted leaf row in the index.

BLKS_GETS_PER_ACCESS : It is the Expected number of consistent mode block reads per row, assuming that a randomly chosen row is accessed using the index.

ROW-PER-KEY : It is the average number of rows per distinct key.

BTREE_SPACE : It is the space currently allocated in B Tree.

USED_SPACE : It is the space currently used in B Tree.

PCT_SPACE : It is the percent of space allocated in the B-Tree that is being used.

Criteria for Rebuild:

Through many research, we found out that the good time to rebuild an index is when the following statistics is obtained. They are :

1. When `blks_gets_per_access > 5`
2. When deleted leaf nodes comprise more than 20 percent of the index nodes i.e `(del_lf_rows / lf_rows) * 100 > 20%`
3. When any index shows height greater than 3 i.e. `(height >3)`

References:

[1]. http://www.dba-oracle.com/t_oracle_analyze_index.htm

[2]. https://docs.oracle.com/cd/B19306_01/server.102/b14237/statviews_4216.htm