Indexes And Partitions In Data Warehouse

What is Indexing?

Indexing is a data structure technique which allows you to quickly retrieve the data. An Index is a small table having only two columns. The first column comprises a copy of the primary or candidate key of a table. Its second column contains a set of pointers for holding the address of the disk block where that specific key value is stored. An index – Takes a search key as input and efficiently returns a collection of matching records.

Types of Indexing

Indexing can defined based on its indexing attributes into:

- Primary Indexing.
- Secondary Indexing.

Indexing can defined based on whether records in the file are physically ordered according to the fields of the index into:

- Clustered Index.
- Non-clustered Index.

Indexing can defined based on whether an index is split in several smaller indexes, with an index to these indexes into:

- Single-level.
- Multiple-level: often implemented by using **B-trees of B^+-trees.**

However, different indexing techniques have been developed which are being used for fast data retrieval in Data warehouse environments, but a low level of concurrent DML transactions.
But our focus is on **Bit-Map** Indexing technique, **Cluster Based** Indexing technique. Brief description of these two techniques is given below.

1- Bitmap indexing

Bitmap indexes are widely used in data warehousing environments, bitmap indexing provides:

- Reduced response time for large classes of ad hoc queries.
- Reduced storage requirements compared to other indexing techniques.
- Dramatic performance gains even on hardware with a relatively small number of CPUs or a small amount of memory.
- Efficient maintenance during parallel DML and loads.

An index provides pointers to the rows in a table that contain a given key value. A regular index stores a list of rowids for each key corresponding to the rows with that key value. In a bitmap index, a bitmap for each key value replaces a list of rowids.

Each bit in the bitmap corresponds to a possible rowid, and if the bit is set, it means that the row with the corresponding rowid contains the key value. A mapping function converts the bit position to an actual rowid, so that the bitmap index provides the same functionality as a regular index. Bitmap indexes store

the bitmaps in a compressed way. If the number of distinct key values is small, bitmap indexes compress better and the space saving benefit compared to a B-tree index becomes even better.

2- Cluster Based Indexing

Data records are ordered according to a field that does not have a distinct value for each record. That field is called a clustering field and cluster index built on that field. Cluster index key is used to order records in the table. On the basis of cluster index two types of index can be formed dense and sparse to get optimized performance. Cluster Based Indexing Technique is good for range based queries but requires sorted data. If data is not sorted then the cost of sorting is also added up. Also Insertions often require reordering of data, so it is a costly operation in terms of time and resources in Data warehousing.

What is Partitioning in the Data Warehouse?

Partitioning helps to scale a data warehouse by dividing database objects into smaller pieces, enabling access to smaller, more manageable objects. Having direct access to smaller objects addresses the scalability requirements of data warehouses.

The data of partitioned tables and indexes is divided into units that may be spread across more than one filegroup in a database or stored in a single filegroup. When multiple files exist in a filegroup, data is spread across files using the proportional fill algorithm. The data is partitioned horizontally, so that groups of rows are mapped into individual partitions. All partitions of a single index or table must reside in the same database. The table or index is treated as a single logical entity when queries or updates are performed on the data.

Partitioning Methods

There are different methods of partitioning. *Range Partitioning, Hash Partitioning, List Partitioning, Composite Partitioning.* Each partitioning method has different advantages and design considerations. Thus, each method is more appropriate for a particular situation.

However, we will focus on *Index Partitioning* and a brief description is given below.

Index Partitioning

You can choose whether or not to inherit the partitioning strategy of the underlying tables. You can create both local and global indexes on a table partitioned by range, hash, or composite methods. Local indexes inherit the partitioning attributes of their related tables. For example, if you create a local index on a composite table, Oracle automatically partitions the local index using the composite method. The rules for partitioning indexes are similar to those for tables:

- An index can be partitioned unless the index is a cluster index and the index is defined on a clustered table.
- You can mix partitioned and nonpartitioned indexes with partitioned and nonpartitioned tables, a partitioned table can have partitioned or nonpartitioned indexes.a nonpartitioned table can have partitioned or nonpartitioned B-tree indexes.
- Bitmap indexes on nonpartitioned tables cannot be partitioned.
- A bitmap index on a partitioned table must be a local index.