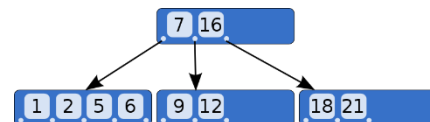**Types of Indexes and Partitions**

1. B-tree indexes

These indexes are the standard index type. They are excellent for primary key and highly-selective indexes. Used as concatenated indexes, B-tree indexes can retrieve data sorted by the indexed columns.
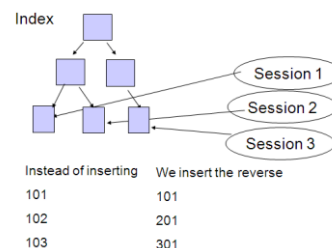
2. Index-organized tables

An index-organized table differs from a heap-organized because the data is itself the index.
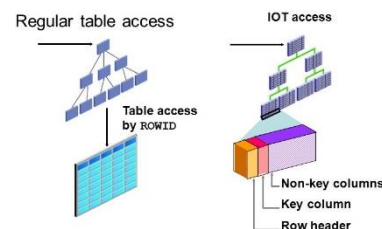
3. Reverse key indexes

In this type of index, the bytes of the index key are reversed, for example, 103 is stored as 301. The reversal of bytes spreads out inserts into the index over many blocks.

4. Function-based indexes

This type of index includes columns that are either transformed by a function, such as the UPPER function, or included in an expression. B-tree or bitmap indexes can be function-based.

5. Fast full indexes

 A fast full index scan is a full index scan in which the database accesses the data in the index itself without accessing the table, and the database reads the index blocks in no particular order.

1. Range Partitioning

Range partitioning maps data to partitions based on ranges of partition key values that you establish for each partition. It is the most common type of partitioning and is often used with dates. For example, you might want to partition sales data into monthly partitions.

2. Hash Partitioning

Hash partitioning maps data to partitions based on a hashing algorithm that Oracle applies to a partitioning key that you identify. The hashing algorithm evenly distributes rows among partitions, giving partitions approximately the same size. Hash partitioning is the ideal method for distributing data evenly across devices. Hash partitioning is a good and easy-to-use alternative to range partitioning when data is not historical and there is no obvious column or column list where logical range partition pruning can be advantageous.

3. List Partitioning

List partitioning enables you to explicitly control how rows map to partitions. You do this by specifying a list of discrete values for the partitioning column in the description for each partition. This is different from range partitioning, where a range of values is associated with a partition and with hash partitioning, where you have no control of the row-to-partition mapping. The advantage of list partitioning is that you can group and organize unordered and unrelated sets of data in a natural way.

4. Composite Partitioning

Composite partitioning combines range and hash or list partitioning. Oracle Database first distributes data into partitions according to boundaries established by the partition ranges. Then, for range-hash partitioning, Oracle uses a hashing algorithm to further divide the data into sub partitions within each range partition. For range-list partitioning, Oracle divides the data into sub partitions within each range partition based on the explicit list you chose.