Index in Oracle SQL :

- An index in SQL is a database object used to speed up data retrieval.
- It works like a "lookup table" for the database, allowing faster access to rows in a table.
- This is especially useful when searching for data using SELECT statements or filtering results with a WHERE clause.
- However, indexes come with a trade-off, they can slow down operations like INSERT, UPDATE, or DELETE because the index itself must also be updated whenever the data changes.
- Creating Index :
  - At the time of table creation :
    - At the time of table creation, we can create indexes directly as part the of the schema.
      1) Using Primary Key
      2) Using Unique Key
      3) Using Index Keyword on specific column(s)
  - After table creation :
    - Index on Single Column

      CREATE INDEX <idx_name>
      ON <table_name> (<column_name>);
    - Index on multiple columns

      CREATE INDEX <idx_name>
      ON <table_name> (<column_name_1_>, <column_name_2_>);
    - Using Unique index on column

      CREATE UNIQUE INDEX <idx_name>
      ON <table_name> (<column_name>);
    - Full-Text Index

      CREATE FULLTEXT INDEX <idx_name>
      ON <table_name> (<column_name>);
    - Spatial Index

      CREATE SPATIAL INDEX <idx_name>
      ON <table_name> (<column_name>);
- Types of Indexes in Oracle SQL :

  B-tree Index:

  This is the most common type of index in Oracle. It is used for equality and range queries, and it organizes data in a balanced tree structure. It is suitable for columns with high cardinality (many distinct values).

Bitmap Index:

It is used for columns with low cardinality (few distinct values), such as gender or boolean flags. It creates a bitmap for each distinct value and stores the bitmaps in a compact format. It is efficient for complex queries that involve multiple conditions on columns with low cardinality.

Clustered Index:

This index stores the data in the same order as the index. It is typically used in clustered tables, where the data of multiple tables is physically grouped together.

Unique Index:

This index ensures that the indexed column(s) contain unique values. It is automatically created for primary key and unique constraints.

Composite Index:

This index is created on multiple columns. It is useful for queries that filter or sort data based on several columns.

Function-based Index:

It allows you to create an index based on a function or expression applied to one or more columns.
This is helpful when queries involve expressions, such as UPPER(column_name).

Reverse Key Index:

It stores the index values in reverse byte order, which can help avoid index block contention in certain scenarios.
This index is useful when there are a large number of sequential key values, such as timestamp data.

Domain Index:

It is created for specialized application domains like spatial, text, and object types.

Domain indexes are typically used in conjunction with Oracle Text, Oracle Spatial, and other specialized Oracle options.

Partitioned Index:

This index is created on partitioned tables, and the index itself can be partitioned.
It helps to manage large tables by dividing the data into smaller, more manageable pieces.

Global and Local Index:

Global Index: An index that spans all partitions of a partitioned table.
Local Index: An index that is local to a specific partition of a partitioned table.

Index Selection and Strategy:

When to Create an Index:

Indexes should be created on columns that are frequently queried, especially those used in WHERE, JOIN, ORDER BY, and GROUP BY clauses.

Impact on DML Operations:

Evaluate the trade-off between query speed and the overhead caused during INSERT, UPDATE, and DELETE operations. Use indexes selectively to avoid excessive performance overhead on write operations.

Choosing the Right Type of Index:

1) B-tree for high-cardinality columns.
2) Bitmap for low-cardinality columns (e.g., gender, status flags).
3) Composite for queries involving multiple columns.
4) Function-based for queries that use functions (e.g., UPPER(column_name)).
5) Reverse Key for avoiding index block contention in sequences.
6) Unique for ensuring uniqueness in data.
7) Domain for specialized indexes (e.g., spatial or text searches).

Index Maintenance:

Rebuilding Indexes:

Over time, indexes can become fragmented, especially after heavy updates or deletions. Regularly rebuilding indexes may help improve performance.

Index Monitoring:

Use Oracle's DBA_INDEXES and DBA_SEGMENTS views to monitor the usage and effectiveness of indexes. This helps identify unused or inefficient indexes.

Dropped Indexes:

If an index is not being used or is too costly to maintain, dropping it can be beneficial. This can reduce unnecessary overhead on INSERT, UPDATE, and DELETE operations.

Index Storage and Optimization:

Tablespace Considerations:

Store indexes in a separate tablespace to optimize performance, especially if the table and index will have different usage patterns (e.g., high read vs. high write).

Partitioned Indexes:

Use partitioned indexes when dealing with partitioned tables. This allows each partition to have its own index, improving manageability and query performance on specific data ranges.

Index Compression:

Compressing indexes can reduce the amount of space used, especially for large tables with low cardinality columns, but may incur CPU overhead during access.

Index Usage Analysis:

Explain Plan: Use the EXPLAIN PLAN feature in Oracle to determine how indexes are being used in your queries. This will show whether the optimizer is using indexes effectively or whether a different type of index might yield better performance.

SQL Tuning Advisor: Use Oracle's SQL Tuning Advisor to get suggestions on missing indexes or indexes that could be optimized.

Optimizer Hints: You can sometimes guide the optimizer by using hints (e.g., USE_INDEX) to enforce the use of specific indexes for better performance.

Avoid Over-Indexing:

Too many indexes can degrade the overall performance, especially during DML operations. Be selective and focus on indexes that align with the query patterns and performance goals of your application.

Specialized Indexes:

If you're working with specialized data types (e.g., spatial data), focus on the specific indexing strategies provided by Oracle, such as Spatial Indexes for geographic data or Full-text Indexes for text searches.