Introduction To Indexes ====>
==========

* Oracle index is one of the effective tools for boosting the query performance.

* However, in order to use it effectively, we must understand it correctly.

* Just as we use the index in the back of a book to quickly find information, similarlyOracle uses indexes to speed up data retrieval.

* If the appropriate index does not exist on a table, then Oracle needs to examine every row.

* This is called a full table scan.

An Important Point ==>
==========

* If an index decreases query time, why not just index every column in a table?

* When we retrieve a large number of rows in a table, it might be more efficient to read the entire table rather than look up the values from the index.

* It also takes a significant amount of time and storage space to build and maintain an index.

* For each DML statement that changes a value in an indexed column, the index needs to be maintained.

Types Of Indexes ==>
==========

* Broadly , Oracle gives us 2 types of indexes :
1. Single Index => Created on the basis of just one column
2. Composite Index => Created on the basis of 2 or more columns

Creating Single Index ==>
==========

* Syntax:
- CREATE INDEX index_name ON table_name (column_name);

* Example:
- CREATE INDEX Vendor_Id ON Vendor_Master (vid);

Creating Composite Index ==>
==========

* Sometimes, it is useful to build indexes based on multiple columns; this type of index is called a composite index, or concatenated index.

* For example, we can create a composite index on two columns with a low selectivity (that is, not many distinct values).

* The combination of these low-selectivity values makes the composite index more selective.

* Syntax:
- CREATE INDEX index_name ON table_name (column_name1,column_name2,. . .);

* Example:
- CREATE INDEX Vendor_Id2 ON Vendor_Master (vid, pid);

Point To Remember ==>
==========

* Columns that are used together frequently in a WHERE clause and combined with the AND logical operator are often good candidates for a composite index.

* The order of the individual columns in the index can affect query performance.

* Choose the column you use most frequently in the WHERE clause first.

* If both columns are accessed with equal frequency, then choose the column with the highest selectivity.

* Suppose , the PID column has very few distinct values then it is considered a low-selectivity column.

* Access against an index with a low-selectivity column as the leading edge requires more index block reads and is therefore less desirable.

NULLs And Indexes ===>
==========

* NULL values are not stored in index, unless it is a composite index where at least the first column of the index contains a value.

* The following query does not make use of the single-column index on the VID column:
- SELECT * FROM Vendor_Master WHERE Vid is null;

Unique Indexes ===>
==========

* Unique indexes are a special type of index created using the command CREATE UNIQUE INDEX.

* A Unique Index automatically applies a UNIQUE CONSTRAINT also but can only be created if the column used for indexing is having unique values.

* When we apply a PRIMARY KEY or UNIQUE CONSTRAINT on a table then Oracle automatically creates a UNIQUE index on those columns.

* Such indexes are also called IMPLICIT INDEX.

Syntax:
- CREATE UNIQUE INDEX index_name ON table_name (column_name);

Example:

- CREATE UNIQUE INDEX Vendor_Id3 ON Vendor_Master (vid);

Function Based Index ===>
==========

* When the query mentioned on the previous slide will be executed , Oracle will not refer the index table because the query is based on LOWER(ENAME) while index is only on ENAME.

* To handle this situation Oracle advices us to create FUNCTION BASED INDEXES as shown below:
- CREATE INDEX Emp_Ind ON Emp(lower(Ename));

* This allows for case-insensitive searches on the ENAME column.

Removing Indexes ===>
==========

* To drop an index, use the DROP INDEX command.
- DROP INDEX <index_name>;

* However the above command allows deletion of the indexes we have created explicitly and not those which Oracle creates implicitly for us.

Obtaining Details About Indexes ===>
==========

* Whenever we apply indexes on a table , then Oracle internally maintains it’s details in it’s DATA DICTIONARIES.

* For indexes , Oracle has 2 DATA DICTIONARIES:
- USER_INDEXES
- USER_IND_COLUMNS

USER_INDEXES ==>
==========

* It contains the following useful columns:
- INDEX_NAME: Stores the name of the index
- UNIQUENESS: Indicates whether an index is unique or not
- TABLE_NAME: Name of the table on which index is created

USER_IND_COLUMNS: ==>
==========

* It contains the following useful columns:
- INDEX_NAME: Stores the name of the index
- COLUMN_NAME: Stores a name of column on which the index has been applied
- COLUMN_POSITION: Contains the position of the column in the index
- TABLE_NAME: Name of the table on which index is applied