

# **INDEXES**

## Let us Increase The Speed And Performance of The Database Using Indexes

## **Working With Indexes:**

#### Index:

- It Is A Schema Object Which Contains An Entry For Each Value That Appears In The Indexed Column(s) Of The Table Or Cluster.
- Index Provides Direct, Fast Access To Rows.

## **Types Of Indexes:**

## **Normal Indexes:**

- They Are Default Indexes.
- They Are Created With B-tree Principle.

## **Bitmap Indexes:**

They Store ROWID'S Associated With A Key Value As A Bitmap.

#### **Partitioned Indexes:**

 They Contain Partitions Containing An Entry For Each Value That Appears In The Indexed Columns of The Table.

## **Function Based Indexes:**

- They Are Based On Expressions.
- Enable Query To Evaluate Value Returned By An Expression.

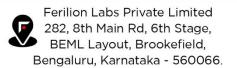
#### **Domain Indexes:**

They Are Indexes Which Are Instances of An Application Specific Index of Type Indextype.

#### **Pre Requsites:**

- The TABLE OR CLUSTER To Be Indexed Must Be In The Own Schema.
- INDEX OBJECT Privilege Should Be Available on The Table To Be Indexed.
- CREATE ANY INDEX System Privilege Must Be Available.
- Unlimited Tablespace System Privilege OR Space Quota On Table Spaces Must Be Available









## Dream, Create, Achieve.

- For DOMAIN Indexes, EXECUTIVE OBJECT Privilege on The IndexType Should Be Available.
- For FUNCTION Based Indexes, The Function Used For Indexing Must Be Marked As Deterministic.

## **Restrictions:**

- If INDEX is Locally Partioned Then The TABLE Must Be Partitioned.
- If The TABLE is INDEX ORGANIZED, Then A Secondary INDEX is Created.
- If The TABLE is TEMPORARY TABLE, Then INDEX is Also Temporary With The Same Scope, As of The Table.

#### Syntax:

```
SQL> CREATE {[UNIQUE/[BITMAP]} INDEX IndexName ON
TableName (ColumnName, [ColumnName...]) TABLESPACE TableSpaceName;
```

## Simple Index OR Normal Index:

- A Index Simple Index OR Normal is Created Upon A Table By Considering Only One
- These Indexes Are Generally Created Using The Algorithm of B-Tree Index.
- We Can Create Index On Column Containing NULLS OR Repeated Data if UNIQUE Key Word is Not Used.
- These Are Default Indexes in Oracle.

```
SQL> CREATE INDEX EmpEmpnoIDX ON Emp(Empno);
SQL> CREATE INDEX DeptDeptnoIDX ON Dept(Deptno);
```

#### <u>Creating Composite Indexes:</u>

- COMPOSITE Index is An INDEX on Multiple Columns of A Table.
- COMPOSITE Index Can Be Created Upon A Table To The Maximun Collection of 32 Columns.

```
Stud Fname Lname Idx ON Student (Fname, Lname);
SQL> CREATE INDEX
SQL> CREATE INDEX Emp Name Place IDX ON Emp(Ename, StreetName);
```

## **Creating Unique Indexes:**

Specify UNIQUE To Indicate That The Value of the Column OR Columns Upon Which The INDEX is Based Must Be UNIQUE.

## **Restrictions:**

- We Cannot Specify Both UNIQUE and BITMAP Indexes at a Time.
- UNIQUE Key Word Cannot Be Specified For A DOMAIN INDEX.

SQL> CREATE UNIQUE INDEX EmpEmailIDxUNQ ON Emp(Email); SQL> CREATE UNIQUE INDEX StuPhnoIDxUNQ ON Student(PhoneNo);













#### **BITMAP Indexes:**

- Specify BITMAP To Indicate That INDEX Has To Be Created With A BITMAP For Each DISTINCT KEY in The Table.
- BITMAP Indexes Store The ROWID's Associated With A KEY Value As A BITMAP.
- Each BIT in The BITMAP Corresponds To A Possible ROWID.
- These Indexes Are Used To Tune Queries That Use Non Selective Columns in Their Limiting Conditions.
- BITMAP INDEXES Should Be Used Only When The Data is Infrequently Updated.
- BITMAP INDEXES Add To The Cost of All Data Manipulation Transactions Against The Tables They INDEX.
- The Oracle Optimizer Can Dynamically Convert BITMAP INDEX Entries To ROWID's During The Query Processing.

#### **Restrictions:**

- BITMAP Cannot Be Specified When Creating A Global Partitioned Index.
- Bitmap Secondary Index Cannot Be Created on An INDEX ORGANIZED TABLE Unless The Index Organized Table Has A Mapping Table Associated With It.
- BITMAP Cannot Be Specified For A DOMAIN INDEX.
- BITMAP INDEXES Should Not Be Used For Tables Involved in OLTP.
- BITMAP Indexes Increase The Load Factor on The INTERNAL Mechanism of Oracle To Maintain Them.
- Restricted With Usage To Tables Involved in BATCH Transactions.

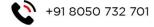
EmpBitMapDeptno ON Emp(Deptno); SQL> CREATE BITMAP INDEX

#### **Bitmap Index Structure:**

- Each Row in The Table Being Indexed Adds Only One Bit to The Size of The Bitmap Pattern Column For The Bitmap Index.
- Each Distinct Value Adds Another Row to The Bitmap Index.

## **Creating Function Based Indexes:**

- These Are Indexes Based On Expressions in SELECT Statements.
- The INDEX Expressions Are Built From Table Columns, Containing SQL Functions OR User Defined Functions.
- FUNCTION BASED INDEXES Defined With The UPPER(Column Name) OR LOWER(ColumnName) Allow Case Insensitive Searches.
- We Should Have CREATE INDEX And QUERY REWRITE Privileges.
- We Should Have EXECUTE Object Privilege on The Functions Used in The Function Based Indexes.
- Function Based Indexes Are Designed To Improve Query Performance When The Function is Used in WHERE Clause.













- To Ensure That Oracle Uses The INDEX Rather Than Performing A Full Table Scan, We Should Be Sure That The Value of The Function is NOT NULL in The Subsequent Queries.
- Oracle Treats INDEXES With Columns Marked DESC As Function Based Indexes.
- The Function Based Indexes Are Used Only When The Query Statement is Executed Through The Specified Function.

SQL> CREATE INDEX EmpFuncAnnSalIDX ON Emp(Sal \* 12);

#### **INDEX Creation is of Two Types:**

- Automatic.
- Manual.

## **Specifications of An INDEX:**

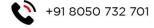
- Index is A Schema Object.
- Index is Used By The Oracle Server To Speed Up Retrieval of Rows By Using A Pointer.
- INDEX Reduces The Disk I/O By Using Rapid Path Access Method To Locate The Data Quickly.
- Index's Are Independent of The Table, Both Logically And Physically.
- Index is Used And Maintained Automatically By The Oracle Server.
- Index's Can Be Created OR Dropped At Any Time And Have No Effect On The Base Tables OR Other Indexes.
- When A Table is Dropped, The Corresponding Indexes Are Also Dropped Automatically.
- On One Table More Than One Index Can Be Created, But This Does Not Mean That, More The Indexes, Lead To More Faster Performance.
- Each DML Operation That is Committed on A Table With Index, Means That The Index Must Be Updated.

#### **Dropping An Index:**

**Syntax:** SQL> DROP INDEX INDEX NAME;

#### When To Create An Index:

- The Column is Used Frequently in The WHERE Clause OR in A Join Condition.
- The Column Contains A Wide Range Of Values.
- The Column Contains A Large Number of NULL Values.
- Two OR More Columns Are Frequently Used Together in A Where Clause OR Join
- The Table is Large And Most Queries Are Expected To Retrieve Less Than 2 To 4 % of The Rows.













## When Not To Create An Index:

- If The Table is Too Small.
- The Columns Are Not Often Used As Condition in The Query.
- Most Queries Are Expected To Retrieve More Than 2 To 4% of The Rows.
- The Table is Updated Frequently.

## **Querying For Indexes:**

- The Indexes Can Be Confirmed From The USER\_INDEXES Data Dictionary.
- The Column That Are Involved in An Index By Querying USER\_IND\_COLUMNS.

SQL>SELECT UC.TABLE\_NAME TabName, UC.COLUMN\_NAME ColName, I.INDEX NAME IDXName FROM USER IND COLUMNS UC, USER INDEXES UI WHERE UC.INDEX NAME = UI.INDEX NAME;







