**INDEXES**

INDEX is a schema object, which a pointer locates the physical address

of data.

INDEX used by the Oracle Server to speed up the retrieval, manipulate of rows.

**Specification of an INDEX:**

• INDEX can be created explicitly or automatically.

• INDEX is activated when indexed column is used in where clause.

• INDEX Necessity of disk I/O by using an indexed path to locate data quickly.

• INDEX is used and maintained automatically by the Oracle Server.

• When you drop a table or column, corresponding indexes are also dropped.

**INDEX Creation is of Two Types:**

1)Automatic:

• A unique index is created automatically when you define

a PRIMARY KEY or UNIQUE constraint in a table

definition.

2) Manual:

• Users can create nonunique indexes on columns to speed up access to the rows.

• One Table more than one Index can be created, but this does not mean that, more the Indexes lead to faster performance.

• Each DML operations that is committed on table on a table with Indexes means that Indexes must be updated.

• USER\_INDEXES hold the details of Indexes.

**Process Of INDEXES:**

* Indexing involves forming a two-dimensional matrix completely independent of the table on which the index is being created.
* The two-dimensional matrix will have a
* Single Column
* Address Field
* **Single Column** will hold sorted data, extracted from the table column(s) on which the index is created.
* **Address Filed** identifies the location of the record in the Oracle database. (**ROWID**)

**Note:**

* The records in the index are stored in the ascending order of the INDEX column(s).

**Types Of INDEXES:**

**NORMAL INDEXES:**

They are default Indexes.

They are created with primary key or unique key.

**Unique:**

• Specify UNIQUE to indicate that the value of the column or columns upon which the Index is based must be UNIQUE.

**BITMAP INDEXES:**

•They store ROWIDs associated with a key value as a Bitmap.

**COMPOSITE INDEX:**

•If we define an index on more than one column, it is called Composite

Index.

**FUNCTION BASED INDEXES:**

* When we create Index on column with function it is called function-based index.

• Enable query to evaluate values returned by expression.

**PRE REQUISITES:**

* The table or CLUSTER to be INDEXED must he in the Own Schema.
* INDEX object privilege should be available on the table to INDEXED.
* Create any Index SYSTEM privilege must be available.
* UNLIMITED TABLESPACE system privilege or SPACE QUOTA on TABLE SPACES must be available.

**Select table\_name,tablespace\_name from user\_tables**

**Where table\_name= 'EMP';**

**Restriction On INDEX Columns:**

An INDEX cannot be created on Columns or ATTRIBUTES whose type is...

* LONG
* LONG RAW
* LOB
* REF

**Syntax:**

Create [UNIQUE] INDEX IndexName

OR [BITMAP] ON

TableName(Column\_name[,ColumnName....]) TABLESPACE TableSpaceName;

**Restrictions:**

• We cannot specify both UNIQUE and BITMAP.

• UNIQUE Cannot be specified for a domain Index

**Restictions:**

BITMAP cannot be specified when creating a global portioned Index.

We cannot specify both UNIQUE and BITMAP.

BITMAP cannot be specified for a DOMAIN INDEX

**Simple Index Example:**

Sql>Create INDEX SalIdx On Emp(Sal)

Sql>Create INDEX Dnoldx ON Dept(Dname)

**Creating Unique INDEXES:**

Sql>Create Unique Index Eno\_Unq\_Idx On Emp(Empno)

**Creating Composite Unique Indexes:**

• Composite Index is an Index on multiple Columns.

Sql>Create Unique Index Eno\_Ename\_CINX On Emp(Empno,Ename);

**Creating Function Based INDEXES:**

• A function-based index is an index-based expression.

• The index expression is built from table columns, constants, SQL functions, and user-defined functions.

• Function-based indexes defined with the UPPER(Column\_Name) or LOWER(Column\_Name) allow case-insensitive searches.

• To ensure that the Oracle Server uses the index rather than performing a full table scan, be sure that the value of the function is Not Null in subsequent queries.

• The oracle Server treats indexes with columns marked DESC as function-based indexes.

**Example:**

Sql>Create INDEX upper\_dept\_name\_idx ON Dept(UPPER(Dname));

Sql>Select \* From Dept Where UPPER(Dname) = 'SALES';

Note: The Function Based Indexes are used only when the Query

Statement is executed through the specified function.

**Bitmap Indexing:**

• Specify BITMAP to indicate that INDEX has to be create with a BITMAP for each DISTINCT KEY.

• BITMAP Indexes store the ROWID's associated with a key value as a BITMAP.

• BITMAP Indexes should be used only when the data is infrequently updated.

• BITMAP Indexes add to the cost of all data manipulation transaction against the tables they INDEX.

• The ORACLE OPTIMIZER can dynamically convert Bitmap Indexes to ROWID's during the query processing.

* Bitmap index will provide the highest optimizer.

**Example:**

Sql>Create Bitmap Index EmpBitMapJob On Emp(Job);

**When to Create An Index:**

• A Column contains a wide range of values.

• A Column contains a large number of NULL values.

• One or more columns are frequently used together in a WHERE clause or a join condition.

• The table is large and most queries are expected to retrieve less than 2 to 4% of the rows.

• The Column is used frequently in the **WHERE** clause or **Join Condition**.

**When Not to Create an Index:**

* The table is small.
* The columns are not often used as a condition in the **Query**.
* Most queries are expected to retrieve more than 2 to 4% of the rows in the table.
* The table is updated frequently
* The indexed columns are referenced as part of an **expression**.

**Removing an Index:**

• Remove an index from the data dictionary by using the DROP INDEX command.

DROP INDEX <Index\_Name>;