**Indexes**

Indexes are database objects which you can create over one or more columns of a table.

Creating index over columns of a table can help you in increasing the performance of data retrieval.

Indexes are logically and physically independent of the data. Thus we can drop and create indexes with no effect in the table or other indexes.

If table does not have index then one value will be checked against each row of the table which decreases the performance of the query.

But index maintain the ordered data in the ascending order. It reduces number of comparisons.

Index is two dimensional table . it contains two part address part(rowid) and data part.

Rowid is the pseduocolumn it contains the physical address of data.

**Rowid has 3 components**

Fileno(table no) datablockno recordno

|  |  |
| --- | --- |
| **index** | |
| Address part | Data part |

**syntax**

Create [unique] index index\_name on table\_name(column\_name[,column\_name…]);

**Type of indexes**

1. **Simple index -**  indexes created on single column.

**B-Tree Index** – is an ordered list of values divided into ranges. B-Tree index associates a key with a row or range of rows as well as provides an excellent retrieval performance for a wide range of queries.

1. **Composite index** – indexes created on multiple columns.
2. **Unique index** - creating index on the unique column.
3. **Reverse index** – index maintain ordered data in ascending order. If we want to fetch highest value always then we create reverse index on that column.
4. **Functional based index** – indexes created on the expressions used in the WHERE clause of the query.
5. **Bitmap index**  - bits are (0,1) . if there are less distinct values then we create bitmap index(example Gender ,status etc. where cardinality is less)

**Examples :-**

1. **Simple index :-**

Create index idx\_sal on emp(sal);

Select \* from emp where sal>10000;

Before creating index ,query goes for FULL table scan.

After creating index it goes for index range scan and increases the performance.

1. **Composite index :-**

Create index idx\_sal\_job on emp(sal,job);

Select \* from emp where sal>1000 and job=’ANALYST’;

Before creating index ,query goes for FULL table scan.

After creating index it goes for index range scan and increases the performance.

Incase we are using below query

Select \* from emp where job=’ANALYST’;

It will go for by index rowid and SKIP SCAN. Since the job is second column in index creation.

Select \* from emp where sal>1000;

Will go for Select \* from emp where sal>1000.

1. **Unique index**

Create unique index idx\_ename on emp(ename);

Select \* from emp where ename=’WARD’;

Before creating index ,query goes for FULL table scan.

After creating index it goes for index range scan and increases the performance.

1. **Reverse index :**

Create index idx\_Sal on emp(sal) reverse;

Before creating index ,query goes for FULL table scan.

After creating index it goes for index range scan and increases the performance.

1. **Functional based index :-**

Create index idx\_lastname on emp(upper(ename));

Before creating index ,query goes for FULL table scan.

After creating index it goes for index range scan and increases the performance.

1. **Bitmap index :-**

Create bitmap index on emp(gender);

Before creating index ,query goes for FULL table scan.

After creating index it goes for index range scan and increases the performance.

**Data dictionary**

1. User\_indexes
2. All\_indexes
3. Dba\_indexes

Select \* from user\_indexes where table\_name=’EMP’;

We can find all the indexes created on the table.

Index\_type column gives what kind of indexes are created like normal,functional,reverse etc.

**NOTE:-**

1. To search for entry in index, some sort of binary search algorithm is used.
2. When data cardinality is high then b-tree index is perfect.
3. Index makes DML slow, as far each record thre should be one entry in the index for indexex column.
4. If not necessary we should avoid creating indexes.

To drop index

**Drop index index\_name;**

To drop all indexes on table

Begin

For I in(select index\_name from user\_indexes where table\_name=’EMP’)

Loop

Execute immediate ‘DROP index ‘||i.index\_name;

End loop;

End;

/

Or

Select ‘drop index ‘ || index\_name from user\_indexes where table\_name=’EMP’;

And run output separately.