INDEXES:

- INDEX IS AN DATABASE OBJECT WHICH IS USED TO RETRIEVE DATA FROM A TABLE FASTLY.

- A DATABASE INDEX WILL WORK AS A BOOK INDEX PAGE IN TEXT BOOK.IN TEXT BOOK BY USING INDEX PAGE WE CAN RETRIEVE A PARTICULAR TOPIC FROM A TEXT BOOK VERY FASTLY SAME AS BY USING DATABASE INDEX OBJECT WE CAN RETRIEVE A PARTICULAR ROW FROM A TABLE VAERY FASTLY.

- BY USING INDEXES, WE CAN SAVE TIME AND IMPROVE THE PERFORMANCE OF DATABASE.THESE INDEXES ARE CREATED BY DBA.

- INDEX OBJECT CAN BE CREATED ON A PARTICULAR COLUMN (OR) COLUMNS OF A TABLE AND THESE COLUMNS ARE CALLED AS "INDEX KEY COLUMNS".

- ALL DATABASES ARE SUPPORTING THE FOLLOWING TWO TYPES OF SEARCHING MECHANISMS THOSE ARE,

1. TABLE SCAN(DEFAULT)
2. INDEX SCAN

1.TABLE SCAN:

- IT IS A DEFAULT SCANNING MECHANISM FOR RETRIEVING DATA FROM TABLE.IN THIS MECHANISM ORACLE SERVER IS SCANNING ENTIRE TABLE (TOP - BOTTOM)

EX:

```
SQL> SELECT * FROM EMP WHERE SAL=3000;
```

SOL:

```
 SAL
--------
 800
1600
1250
2975
1250
```

2850

2450

3000 (IN THIS TABLE SCAN WE ARE COMPARING WHERE CONDITION 14 TIMES)

5000

1500

1100

950

3000

1300

2) INDEX SCAN:

- IN INDEX SCAN MECHANISM ORACLE SERVER SCANNING ONLY INDEXED COLUMN FROM A TABLE. IN THIS MECHANISM WE AGAIN FOLLOW THE FOLLWOING TWO METHODS,

I) AUTOMATICALLY / IMPLICITLY:

- WHENEVER WE ARE CREATING A TABLE ALONG WITH "PRIMARY KEY " (OR) "UNIQUE" KEY CONSTRAINT THEN INTERNALLY SYSTEM IS CREATING AN INDEX OBJECT ON THAT PARTICULAR COLUMN AUTOMATICALLY.

EX:

```
SQL> CREATE TABLE TEST1(EID INT PRIMARY KEY, ENAME VARCHAR2(10));

SQL> CREATE TABLE TEST2(SNO INT UNIQUE, NAME VARCHAR2(10));
```

NOTE:

- IF WE WANT TO VIEW INDEX NAME ALONG WITH COLUMN NAME OF A PARTICULAR TABLE THEN WE USE "USER_IND_COLUMNS" DATA DICTIONARY.

EX:

```
SQL> DESC USER_IND_COLUMNS;

SQL> SELECT COLUMN_NAME, INDEX_NAME FROM USER_IND_COLUMNS WHERE TABLE_NAME='TEST1';
```

COLUMN_NAME INDEX_NAME

--------------------- ---------------------

EID SYS_C005501

SQL> SELECT COLUMN_NAME, INDEX_NAME FROM USER_IND_COLUMNS WHERE TABLE_NAME='TEST2';

COLUMN_NAME INDEX_NAME

--------------------- ---------------------

SNO SYS_C005502

II) MANUALLY / EXPLICITLY:

- WHEN USER WANT TO CREATE AN INDEX OBJECT ON A PARTICULAR COLUMN/(S) THEN WE FOLLOW THE FOLLOWING SYNTAXS,

TYPES OF INDEXES:

1. B - TREE INDEX (DEFAULT INDEX)

- **SIMPLE INDEX**
- **COMPOSITE INDEX**
- **UNIQUE INDEX**
- **FUNCTIONAL BASED INDEX**

2. BITMAP INDEX

SIMPLE INDEX:

- WHEN WE CREATED AN INDEX ON A SINGLE COLUMN THEN WE CALLED AS SIMPLE INDEX.

SYNTAX:

CREATE INDEX <INDEX NAME> ON <TN> (<COLUMN NAME>);
EX:

SQL> CREATE INDEX SIND ON EMP(SAL);

INDEX CREATED.

EX:

SQL> SELECT * FROM EMP WHERE SAL=3000;

SOL:

B-TREE (BINARY TREE)

(<) |LP| 3000 |RP| (>=)

|

LP| 2975 | RP LP | 5000 | RP | |

2850|*, 2450|*, 1600|*, 1500|* | 3000 |*, *| 1300|*, 1250|*, *, 1100|*,

950|*, 800|*

NOTE: IN INDEX SCAN WE ARE COMPARING 3 TIMES.WHICH IS MUCH FASTER THAN TABLE SCAN (14 TIMES COMPARING). HERE " * " IS REPRESENT ROWID.

COMPOSITE INDEX:

- WHEN WE CREATED AN INDEX ON MULTIPLE COLUMNS THEN WE CALLED AS COMPOSITE INDEX.

SYNTAX:

CREATE INDEX <INDEX NAME> ON <TN> (<COLUMN NAME1>, <COLUMN NAME2>, ...);

EX:

SQL> CREATE INDEX CIND ON EMP (DEPTNO, JOB); INDEX

CREATED.

NOTE: ORACLE SERVER USES ABOVE INDEX WHEN "SELECT" QUERY WITH WHERE CLAUSE IS BASED ON LEADING COLUMN OF INDEX,I.E (DEPTNO).

EX:

SQL> SELECT * FROM EMP WHERE DEPTNO=10;(INDEX SCAN)

SQL> SELECT * FROM EMP WHERE DEPTNO=10 AND JOB='CLERK';(INDEX SCAN)

SQL> SELECT * FROM EMP WHERE JOB='CLERK';(TABLE SCAN)

UNIQUE INDEX:

- WHEN WE CREATE AN INDEX BASED ON "UNIQUE CONSTRAINT" COLUMN IS CALLED UNIQUE INDEX.UNIQUE INDEX DOES NOT ALLOW DUPLICATE VALUES.

SYNTAX:

CREATE UINQUE INDEX <INDEX NAME> ON <TN> (<COLUMN NAME>);

EX:

SQL> CREATE UNIQUE INDEX UIND ON DEPT(DNAME); INDEX

CREATED.

TESTING:

SQL> INSERT INTO DEPT VALUES (50,'SALES','HYD') ERROR

AT LINE 1:

ORA-00001: UNIQUE CONSTRAINT (SCOTT.UIND) VIOLATED.

NOTE: PRIMARY KEY COLUMNS AND UNIQUE COLUMNS ARE AUTOMATICALLY INDEXED BY ORACLE.

FUNCTIONAL BASED INDEX:

- WHEN WE CREATE AN INDEX BASED ON FUNCTION THEN WE CALLED AS FUNCTIONAL BASED INDEX.

SYNTAX:

CREATE INDEX <INDEX NAME> ON <TN>(<FUNCTION NAME>(COLUMN NAME));
EX:

SQL> CREATE INDEX IND4 ON EMP(UPPER(ENAME)); INDEX

CREATED.

SQL> SELECT * FROM EMP WHERE UPPER(ENAME)='SCOTT';(INDEX SCAN)

2. BITMAP INDEX:

- BITMAP INDEX IS CREATED ON DISTINCT VALUES OF A PARTICULAR COLUMN.GENERALLY BITMAP INDEXES ARE CREATED ON LOW CARDINALITY OF COLUMNS.

- WHEN WE CREATE BITMAP INDEX INTERNALLY ORACLE SERVER IS PREPARING BITMAP INDEXED TABLE WITH BIT NUMBERS ARE 1 AND 0. HERE 1 IS REPRESENT CONDITION IS TRUE WHERE AS 0 IS REPRESENT CONDITION IS FALSE.

CARDINALITY:

- IT REFERES TO THE UINQUENESS OF DATA VALUES CONTAINE IN PARTICULAR COLUMN OF TABLE.

HOW TO FIND CARDINALITY OF A COLUMN:

$$\text{CARDINALITY OF COLUMN} = \frac{\text{NO. OF DISTINCT VALUES OF A COLUMN}}{\text{NO. OF ROWS IN A TABLE}}$$

EX:

CARDINALITY OF EMPNO = 14

14

CARDINALITY OF EMPNO IS "1" ----(CREATING BTREE INDEX)
EX:

CARDINALITY OF JOB = 5

CARDINALITY OF JOB = 0.35 ------ (CREATING BIT MAP INDEX)

SYNTAX:

CREATE BITMAP INDEX <INDEX NAME> ON <TN>(<COLUMN NAME>);

EX:

CREATE BITMAP INDEX BITIND ON EMP(JOB);

EX:

SELECT * FROM EMP WHERE JOB='MANAGER';

BITMAP INDEXED TABLE

==========

JOB 1 2 3 4 5 6 7 8 9 10 11 12 13 14

========== ==========

CLERK 1 0 0 0 0 0 0 0 0 1 1 0 1

---------- ----------

SALESMAN 0 1 1 0 1 0 0 0 0 1 0 0 0 0

---------- ----------

MANAGER 0 0 0 1 0 1 1 0 0 0 0 0 0 0

---------- ----------

ANALYST 0 0 0 0 0 0 0 1 0 0 0 0 1 0

---------- ----------

PRESIDENT 0 0 0 0 0 0 0 0 1 0 0 0 0 0

==========

NOTE: HERE "1" IS REPRESENTED WITH ROWID OF A

PARTICULAR ROW IN A TABLE.

NOTE:

- IF WE WANT TO VIEW INDEX NAME ALONG WITH INDEX TYPE THEN WE USE "USER_INDEXES" DATADICTIONARY.

EX:

```
SQL> DESC USER_INDEXES;

SQL> SELECT INDEX_NAME, INDEX_TYPE FROM USER_INDEXES  WHERE

TABLE_NAME='EMP';

INDEX_NAME INDEX_TYPE

---------- ----------

SIND NORMAL(B-TREE)

BITIND BITMAP

FIND FUNCTION-BASED NORMAL(B-TREE) UIND NORMAL(B-TREE)

CIND NORMAL(B-TREE)
```

HOW TO DROP AN INDEX:

SQL> DROP INDEX <INDEX NAME>;

EX:

```
SQL> DROP INDEX SIND;

SQL> DROP INDEX BITIND;
```