Indexes in Oracle SQL

1. What is an Index?

An index is a database object that stores a sorted copy of specific columns
to speed up searches. Instead of scanning all rows, Oracle uses the index for faster lookups.

Example Without an Index (Full Table Scan):

```
SELECT * FROM employees WHERE last_name = 'Smith';
```

Example With an Index:

```
CREATE INDEX idx_lastname ON employees(last_name);
```

---

2. Types of Indexes in Oracle

2.1 B-Tree Index (Default)

- The most common index type.
- Efficient for exact match and range-based queries (> , < , BETWEEN).

Example:

```
CREATE INDEX idx_salary ON employees(salary);
```

2.2 Unique Index

- Ensures values in a column are unique.

Example:

```
CREATE UNIQUE INDEX idx_emp_id ON employees(emp_id);
```

2.3 Bitmap Index

- Ideal for columns with few unique values (low cardinality).

Example:

```
CREATE BITMAP INDEX idx_gender ON employees(gender);
```

2.4 Composite (Multi-Column) Index

- An index on multiple columns.
Example:
CREATE INDEX idx_dept_salary ON employees(department_id, salary);

2.5 Function-Based Index
- Indexes a calculated expression.
Example:
CREATE INDEX idx_upper_lastname ON employees(UPPER(last_name));

2.6 Invisible Index
- Index is not used automatically but can be enabled when needed.
Example:
CREATE INDEX idx_hidden ON employees(email) INVISIBLE;
ALTER INDEX idx_hidden VISIBLE;

---

3. When to Use Indexes
Use an index when:
- The column is frequently used in WHERE, JOIN, or ORDER BY.
- The column has high cardinality (many unique values).
- Queries filter a small percentage of rows.

Avoid indexes when:
- The table is small (full table scan is faster).
- The column has low cardinality (use Bitmap Index).
- You perform frequent INSERT, UPDATE, DELETE (Indexes slow down DML).

---

4. How to Check Indexes on a Table
View Indexes:
SELECT INDEX_NAME, COLUMN_NAME FROM USER_IND_COLUMNS WHERE TABLE_NAME = 'EMPLOYEES';

---

5. Dropping and Rebuilding Indexes
Drop an Index:
```
DROP INDEX idx_salary;
```

Rebuild an Index:
```
ALTER INDEX idx_salary REBUILD;
```

---

6. Indexing Best Practices
- Index frequently searched columns.
- Use composite indexes for multi-column filtering.
- Use Function-Based Indexes when filtering with functions.
- Use Bitmap Indexes for low-cardinality columns.
- Monitor index performance with EXPLAIN PLAN.

Summary:
- Indexes speed up queries but slow down DML (INSERT, UPDATE, DELETE).
- B-Tree indexes are best for high-cardinality columns.
- Bitmap indexes are best for low-cardinality columns.
- Function-Based Indexes help with queries using functions.
- Use EXPLAIN PLAN to check index usage.