# --------LECTURE- 19--------

## 1 What is an INDEX?

**Simple definition**

An **index** is a **separate data structure** that helps the database **find rows faster**, without scanning the entire table.

**Real-life analogy 🕮**

Think of a **book**:

- Without index → you read page by page
- With index → you jump directly to page number

👉 Database table = book
👉 Index = book index page

---

## 2 Why indexes are used

*Indexes in MySQL are used to speed up data retrieval (SELECT queries)*

**WHY exactly?**

Without index:

- Database checks **every row**
- This is called **Full Table Scan**

With index:

- Database jumps to **only matching rows**
- Much fewer reads

**Example**

SELECT * FROM employee WHERE eid = 103;

- Without index → check all employees
- With index → jump directly to eid = 103

---

## 3 "Index is a data structure" — WHAT structure?

*An index is a data structure*

But **what kind?**

**Most common structure = B-TREE**

Why B-tree?

- Balanced
- Sorted
- Fast search
- Fast insert/delete

Time complexity:

- Search: **O(log n)**
  instead of **O(n)**

That's the real reason indexes are fast.

---

## 4 Pros of Index

### ✓ Faster SELECT

Because fewer rows are scanned.

### ✓ Faster WHERE

Because index helps locate rows directly.

### ✓ Faster JOIN

Because matching keys is faster.

### ✓ Faster ORDER BY

Because index is already sorted.

---

## 5 Cons of Index (VERY important)

*Slows down INSERT, UPDATE, DELETE*

**WHY?**

Because:

- Data changes → index must change
- Extra work for database

Example:

INSERT INTO employee VALUES (201, 'Amit');

Database does:

1. Insert row into table
2. Update index structure
3. Rebalance B-tree if needed

👉 More indexes = more overhead

---

## 6 When to AVOID index

### ❌ Small tables

Why?

- Table scan is already fast
- Index overhead is unnecessary

### ❌ Low-cardinality columns

example: gender

Values:

- Male
- Female

Index doesn't help much because:

- Too many rows match
- Database still reads many rows

---

## 7 Types of Index

### 1 PRIMARY KEY INDEX

- Automatically created
- UNIQUE
- NOT NULL
- Only one per table

eid INT PRIMARY KEY

Fastest lookup.

---

### 2 UNIQUE INDEX

- No duplicate values
- NULL allowed (depends on DB)

```
CREATE UNIQUE INDEX idx_email ON employee(email);
```

Ensures data quality + fast search.

---

### 3 NORMAL INDEX

- Allows duplicates
- Used only for performance

```
CREATE INDEX idx_cadd ON employee(cadd);
```

---

### 4 COMPOSITE INDEX

Index on **multiple columns**

```
CREATE INDEX idx_esal_cadd ON employee(esal, cadd);
```

Order matters
This index works for:

- esal
- esal + cadd
  ✕ Not for only cadd

---

### 8 SELECT examples explained

**Example 1**

```
SELECT * FROM employee WHERE eid = 103;
```

Fast because:

- eid is PRIMARY KEY
- Uses index lookup

---

**Example 2**

SELECT * FROM employee WHERE cadd = 'Noida';

Slow because:

- No index
- Full table scan

---

## 9 SHOW INDEX FROM employee

This shows:

- Index name
- Column name
- Cardinality

**Cardinality = ?**

👉 Number of **unique values**

Higher cardinality = better index

---

## 10 What is EXPLAIN?

**Simple meaning**

EXPLAIN tells:

"HOW the database plans to run this query"

It does **NOT run** the query.

---

**Example**

EXPLAIN SELECT * FROM employee WHERE esal > 20000;

---

**Important type values (MySQL)**

⚫ **type = ALL**

- Full table scan
- Worst performance

🟤 **type = ref**

- Index used

- Filtered rows

This tells you if index is helping or not.

---

## 1️⃣1️⃣ Composite Index explained properly

CREATE INDEX idx_esal_cadd ON employee(esal, cadd);

**How database stores it:**

Sorted first by:

1. esal
2. then cadd

Works well for:

WHERE esal > 20000

WHERE esal > 20000 AND cadd = 'Noida'

Does NOT work for:

WHERE cadd = 'Noida'

---

## 1️⃣2️⃣ Why B-TREE index is used

Your note:

*Index use BTREE Algorithm*

**What B-TREE gives:**

- Balanced tree
- Fast search
- Ordered data

Used for:

- =
- <
- >
- BETWEEN
- LIKE 'abc%'

❌ Not good for:

- LIKE '%abc'

---

**FINAL**

- Table = raw data
- Index = shortcut structure
- SELECT = read operation
- INSERT/UPDATE = maintenance cost
- EXPLAIN = X-ray of query