1. What is an Index

An **index** in MySQL is a **data structure** that improves the **speed of data retrieval** operations on a table at the cost of additional storage and slower writes (insert/update/delete).

Think of it like the **index at the beginning of a textbook** — it helps you quickly find the page number for a topic instead of reading every page.

Benefits

a. To **speed up SELECT queries** and WHERE clause lookups.
b. To **optimize JOIN operations**.

Drawbacks

a. Slows write operations such as Inserts, Updates and Deletes.

1.1. Primary Key Index

A primary key is a unique identifier for each row in a table.

When you define a primary key in MySQL, **MySQL automatically creates a unique index** on that column (or columns). This is called the **Primary Key Index**.

Benefits of a Primary Key

1. **Uniqueness**
    - Ensures that no two rows have the same value in the primary key column(s).
    - Example: In a students table, each student_id should be unique.
2. **Fast Lookup (Improved Performance)**
    - The primary key index allows MySQL to **quickly find rows** by their ID.
    - It uses a data structure (usually a **B-tree**) to speed up search and join operations.
3. **No Nulls Allowed**
    - Primary keys **cannot contain NULLs**. Every row must have a value in the primary key column.
4. **Foundation for Relationships**
    - Other tables can use this column in a **foreign key** to refer to this table.
    - It auto-creates the index when you define the key.

1.2. Unique Key Index

A unique index prevents duplicate values in a column. It is similar to a primary key, but with one major difference i.e a table can have only one primary key, but it can have multiple unique indexes

Purpose of a Unique Index

1. **Guarantees uniqueness** of values in one or more columns.
2. **Speeds up searches** (like all indexes do).
3. Can be used when the column should have unique values but **is not the primary identifier** for the row.

Example:

```
CREATE TABLE students (
    student_id INT PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(100),
    UNIQUE (email)
);
```

The above SQL

1. Creates a unique index on the email column.
2. Prevents you from inserting two rows with the same email.

Compare a Unique Key with a Primary Key

Feature	Primary Key	Unique Index
Enforces Uniqueness	Yes	Yes
Number Per Table	Only one	Can be many
Automatically Indexed	Yes	Yes
Used for Relationships	Often used with Foreign Keys	n/a

Some other example columns to Use Unique Index, just so you build the intuition

1. Email addresses
2. Usernames
3. Phone numbers
4. Any column where duplicate values are not allowed, but it's not the main identifier of the row.

1.3. Composite Index

A composite index is a single index that covers two or more columns together. It enables MySQL to look up data faster when it must match several columns at once.

You can think of it like a multi-column address book: Imagine organizing books in a library first by genre, then by author. If someone tells you the genre and the author, you can find the book fast. But if you have only the author's name — it's harder because you'd still have to look through every genre.

Example: Assume we run the below query

```
SELECT * FROM students
WHERE city = 'Bangalore' AND course = 'Python';
```

Instead of using two separate indexes (city, course), we can use a **composite index**:

```
CREATE INDEX idx_city_course ON students(city, course);
```

It works best for:

- WHERE city = '...' AND course = '...'
- WHERE city = '...'
- However, it is **not helpful** for: WHERE course = '...' alone

Because the index starts with city, and MySQL **needs to match the first column** before efficiently using the rest.

Feature	Composite Index
What It Is	Index on **two or more columns**
Use Case	Speeds up queries filtering by multiple columns
Order Matters?	Yes — must match columns **from left to right**
Improves Performance	For multi-column searches
Examples	(city, course), (email, created_date), (user_id, status)

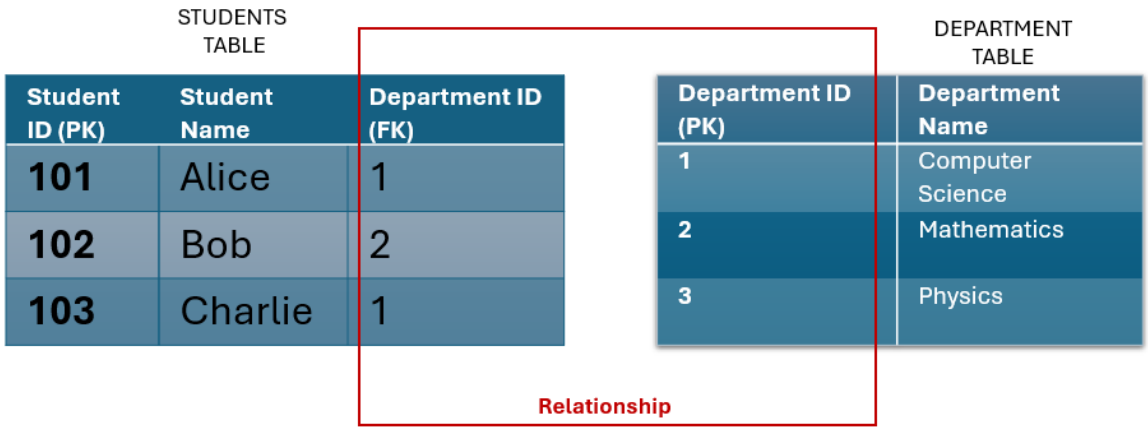
2. Understanding relationship between Primary and Foreign Keys

Primary Key:

1. Uniquely identifies each row in a table.
2. No two rows can have the same primary key value.
3. A table can have only one primary key.
4. Does not take NULL values
5. Enforces data integrity by ensuring each row is distinct

Foreign Key:

1. A field in one table that references the primary key of another table.
2. Establishes a relationship between the two tables.
3. Allows you to relate data in one table to data in another.
4. Enforces referential integrity, meaning the foreign key value must exist as a primary key value in the related table.
5. A table can have multiple foreign keys.



3. Follow Up Questions

Can you try and find answers to

1. What is Referential integrity?
2. What are referential integrity constraints?
3. How do you apply referential integrity constraints?

Can you illustrate the above 3 with the aid of an example?