**INDEXING**

**CREATE INDEX** statement in SQL is used to create indexes in tables. The indexes are used to retrieve data from the database more quickly than others. The user cannot see the indexes, and they are just used to speed up queries /searches.

**Note:** Updating the table with indexes takes a lot of time than updating a table without indexes. It is because the indexes also need an update. So, only create indexes on those columns that will be frequently searched against.

-- Indexing

Drop database if exists indexing;

create database indexing;

use indexing;

-- PRIMARY KEY

-- Description: A primary key is a unique index that identifies each row in a table. Only one primary key is allowed per table, and it must be unique and non-null.

-- Automatically Created: Yes, when you define a column as PRIMARY KEY.

CREATE TABLE my\_table(

id INT PRIMARY KEY,

firstname VARCHAR(100),

lastname varchar(100),

age INT,

location POINT NOT NULL SRID 4326, -- Define POINT with SRID

address Text

);

-- Understanding SRID

-- SRID (Spatial Reference System Identifier):

-- SRID is used to define the coordinate system and spatial reference for spatial data.

-- Without an SRID, MySQL cannot fully optimize spatial queries.

-- you should define the GEOMETRY column with a specific SRID.

-- This is often done by using specific spatial types like POINT, LINESTRING, or POLYGON along with an SRID.

-- SRID 4326 is not a fixed number but rather a specific identifier used in spatial databases to define a particular spatial reference system.

-- It refers to the WGS 84 (World Geodetic System 1984) coordinate system, which is commonly used for GPS and global positioning systems.

-- SRID 4326: WGS 84 (latitude/longitude)

-- SRID 3857: Web Mercator (used in web maps like Google Maps)

-- SRID 4269: NAD83 (North American Datum 1983)

-- How to see indexes in table. below are all commands that can show indexing.

SHOW INDEX FROM my\_table;

SHOW KEYS FROM my\_table;

DESCRIBE my\_table;

EXPLAIN my\_table;

SHOW CREATE TABLE my\_table;

SELECT \*

FROM INFORMATION\_SCHEMA.STATISTICS

WHERE table\_schema = 'indexing'

AND table\_name = 'my\_table';

-- 2. UNIQUE INDEX

-- Description: This type of index ensures that all the values in the indexed column(s) are unique.

-- Use Case: When you want to ensure that a column or group of columns has no duplicate values.

-- You create a unique index to both enforce uniqueness and improve the speed of lookups and queries that involve the indexed column(s).

-- Example:

CREATE UNIQUE INDEX idx\_unique\_name ON my\_table (firstname);

-- 3. INDEX (or NON-UNIQUE INDEX)

-- Description: A standard index allows for faster retrieval of data but doesn’t enforce uniqueness.

-- Use Case: When you want to speed up queries on columns that may contain duplicate values.

-- Example:

CREATE INDEX idx\_name ON my\_table (lastname);

-- 4. FULLTEXT INDEX

-- Description: A specialized index used for full-text searches in CHAR, VARCHAR, or TEXT columns.

-- Use Case: When you need to perform full-text searches on large text fields.

-- Example:

CREATE FULLTEXT INDEX idx\_fulltext\_name ON my\_table (address);

-- 5. SPATIAL INDEX

-- you should define the GEOMETRY column with a specific SRID. This is often done by using specific spatial types like POINT, LINESTRING, or POLYGON along with an SRID.

--

-- Description: A spatial index is used for indexing spatial data types like GEOMETRY.

-- Use Case: When working with geographical data (e.g., latitude and longitude) or spatial queries.

-- Example:

CREATE SPATIAL INDEX idx\_spatial\_location ON my\_table (location);

-- 6. COMPOSITE INDEX

-- Description: An index that includes more than one column. This can improve performance when queries filter or sort by multiple columns.

-- Use Case: When you frequently query by multiple columns together.

-- Example:

CREATE INDEX idx\_name\_age ON my\_table (firstname, age);

-- CLUSTERED INDEX Meaning

-- You can have only one clustered index in a table but,

-- you can have one clustered index on multiple columns, and that type of index is called a composite index.

-- Description: In MySQL's InnoDB engine, the primary key is a clustered index, meaning the data is physically stored in the order of the primary key.

-- Use Case: When you want to optimize data retrieval based on the primary key. MySQL automatically creates a clustered index on the primary key.

-- You can create only one clustered index in a table like a primary key. A clustered index is as same as a dictionary where the data is arranged in alphabetical order.

-- Non CLUSTERED INDEX

-- The non-Clustered Index is similar to the index of a book. The index of a book consists of a chapter name and page number,

-- if you want to read any topic or chapter then you can directly go to that page by using the index of that book. No need to go through each and every page of a book.

-- The data is stored in one place, and the index is stored in another place. Since the data and non-clustered index is stored separately,

-- then you can have multiple non-clustered indexes in a table.

-- In a non-clustered index, the index contains the pointer to data.