MySQL Index Tutorial - Create, Add & Drop

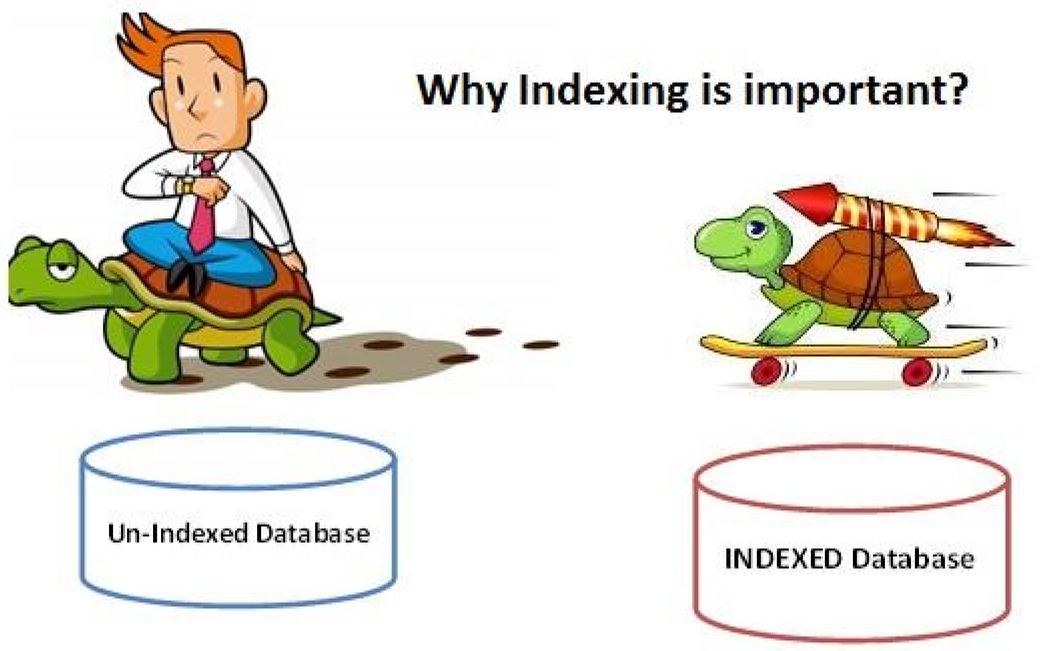
What is an Index?

Indexes in MySQL sort data in an organized sequential way. They are created on the column(s) that will be used to filter the data. Think of an index as an alphabetically sorted list. It is easier to lookup names that have been sorted in alphabetical order than ones that are not sorted.

Using an index on tables that are frequently updated can result in poor performance. This is because MySQL creates a new index block every time that data is added or updated in the table. Generally, indexes should be used on tables whose data does not change frequently but is used a lot in select search queries.

What use an Index?

Nobody likes slow systems. High system performance is of prime importance in almost all database systems. Most businesses invest heavily in hardware so that data retrievals and manipulations can be faster. But there is a limit to hardware investments a business can make. Optimizing your database is a cheaper and better solution.



The slowness in the response time is usually due to the records being stored randomly in database tables. Search queries have to loop through the entire randomly stored records one after the other to locate the desired data. This results in poor performance databases when it comes to retrieving data from large tables. Hence, Indexes are used to sort data to make it easier to search.

Syntax: Create Index

Indexes can be defined in 2 ways

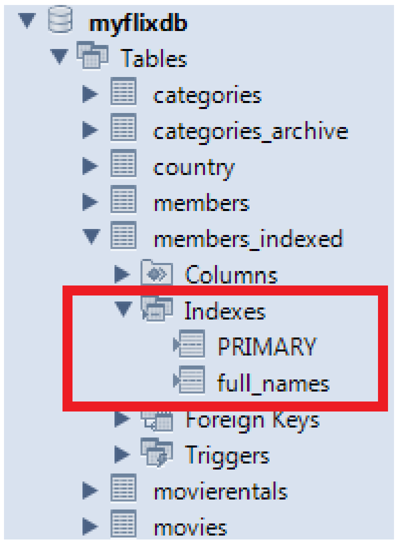
1. At the time of table creation
2. After table has been created

Example:
For our myflixdb we expect lots of searches to the database on full name.

We will add the "full_names" column to Index in a new table "members_indexed".

The script shown below helps us to achieve that.

```
CREATE TABLE `members_indexed` (
  `membership_number` int(11) NOT NULL AUTO_INCREMENT,
  `full_names` varchar(150) DEFAULT NULL,
  `gender` varchar(6) DEFAULT NULL,
  `date_of_birth` date DEFAULT NULL,
  `physical_address` varchar(255) DEFAULT NULL,
  `postal_address` varchar(255) DEFAULT NULL,
  `contact_number` varchar(75) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`membership_number`),INDEX(full_names)
) ENGINE=InnoDB;
```
Execute the above SQL script in MySQL workbench against the "myflixdb".



Refreshing the myflixdb shows the newly created table named members_indexed.

"Note" members_indexed table has "full_names" in the indexes node.

As the members base expands and the number of records increases , search queries on the members_indexed table that use the WHERE and ORDER BY clauses will be much faster compared to the ones performed on the members table without the index defined.

Add index basic syntax

The above example created the index when defining the database table. Suppose we already have a table defined and search queries on it are very slow. They take too long to return the results. After investigating the problem, we discover that we can greatly improve the system performance by creating INDEX on the most commonly used column in the WHERE clause.

We can use following query to add index

```
CREATE INDEX id_index ON table_name(column_name);
```

Let's suppose that search queries on the movies table are very slow and we want to use an index on the "movie title" to speed up the queries, we can use the following script to achieve that.

```
CREATE INDEX `title_index` ON `movies`(`title`);
```

Executing the above query creates an index on the title field in the movies table.

This means all the search queries on the movies table using the "title" will be faster.

Search queries on other fields in the movies table will however still be slower compared to the ones based on the indexed field.

Note: You can create indexes on multiple columns if necessary depending on the fields that you intend to use for your database search engine.

If you want to view the indexes defined on a particular table, you can use the following script to do that.

```
SHOW INDEXES FROM table_name;
```

Let's now take a look at all the indexes defined on the movies table in the myflixdb.

```
SHOW INDEXES FROM `movies`;
```

Executing the above script in MySQL workbench against the myflixdb gives us results on the index created.

Note: The primary and foreign keys on the table have already been indexed by MySQL. Each index has its own unique name and the column on which it is defined is shown as well.

Syntax: Drop index

The drop command is used to remove already defined indexes on a table.

There may be times when you have already defined an index on a table that is frequently updated. You may want to remove the indexes on such a table to improve the UPDATE and INSERT queries performance. The basic syntax used to drop an index on a table is as follows.

```
DROP INDEX `index_id` ON `table_name`;
```

Let's now look at a practical example.

```
DROP INDEX ` full_names` ON `members_indexed`;
```

Executing the above command drops the index with id ` full_names ` from the members_indexed table.

Summary

- Indexes are very powerful when it comes to greatly improving the performance of MySQL search queries.
- Indexes can be defined when creating a table or added later on after the table has already been created.
- You can define indexes on more than one column on a table.
- The SHOW INDEX FROM table_name is used to display the defined indexes on a table.
- The DROP command is used to remove a defined index on a given table