Indexing is an important feature of MySQL. It supports different type of indexes like primary key index, unique index, normal index and full-text index. Indexes help to speed up the retrieval of data from MySQL database server. When retrieving the data from a database table, MySQL first checks whether the index of table exists; If yes it will use index to select exact physical corresponding rows without scanning the whole table.

**Creating Indexes:**

Mostly we create index when creating table. Any column in creating table statement declared as PRIMARY KEY, KEY, UNIQUE or INDEX will be indexed automatically by MySQL. In addition, you can add indexes to the tables which has data. The statement to create index in MySQL is as follows:

**CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index\_name  
USING [BTREE | HASH | RTREE]  
ON table\_name (column\_name [(length)] [ASC | DESC],…)**

**Unique:** Refers to an index where all rows of the index must be unique. That is, the same row may not have identical non-NULL values for all columns in this index as another row. As well as being used to speed up queries, UNIQUE indexes can be used to enforce restraints on data, because the database system does not allow this distinct values rule to be broken when inserting or updating data.

**FULLTEXT:** indexes are different from all of the above, and their behaviour differs significantly between database systems. FULLTEXT indexes are only useful for full text searches done with the MATCH() / AGAINST() clause, FULLTEXT is typically implemented internally using b-trees (allowing for selecting, sorting or ranges starting from left most column) or hash tables (allowing for selection starting from left most column).

Where the other index types are general-purpose, a FULLTEXT index is specialised, in that it serves a narrow purpose: it's only used for a "full text search" feature.

**Spatial:** Spatial Index is like an ordinary index with this difference that Spatial objects are not 1D data points rather are in higher dimension space (e.g. 2D) and thus Ordinary indexes such as BTree are not appropriate for indexing such data. The well-known spatial Index technique is R-tree. Spatial index are used for indexing geo-objects - shape.

In above statement UNIQUE specify that MySQL will create a constraint that all values in the index must be distinct. Duplicated NULL is allowed in all storage engine except BDB.

The FULLTEXT index is supported only by MyISAM storage engine and only accepted columns which have data type is CHAR,VARCHAR or TEXT.

The SPATIAL index supports spatial column and available in MyISAM storage engine. In addition, the column value must not be NULL.

Then you name the index using index types such as BTREE, HASH or RTREE also based on storage engine

In below index create statement, I have created index to emp\_no column on employees table to make the record retreaval faster. The SQL statement to create index is as follows:

**CREATE INDEX emp\_no ON employees(emp\_no)**

##### **Advantages of MySQL Indexes**

1- Indexes make search queries much faster.

2- Indexes like primary key index and unique index help to avoid duplicate row data.

3- Full-text indexes in MySQL, users have the opportunity to optimize searching against even large amounts of text located in any field indexed as such.

##### **Disadvantages of MySQL indexes**

Actually a separate file created when a new index created on the table column. that file stored only the field you’re interested in sorting on. So when we create index, it takes up disk space. but because of creating index on every column in every possible combination, the index file would grow much more quickly than the data file. In the case when a table is of large table size, the index file could reach the operating system’s maximum file size.

The index also slow down the speed of writing queries, such as INSERT, UPDATE and DELETE. AS MySQL has to internally maintain the pointers to the inserted rows in the actual data file, so there is a performance price to pay in case of above said writing queries because every time a record is changed, the indexes must be updated.

So Indexes are important to speed in large MySQL databases. it doesn’t matter how small your table, a 100000-row table scan will never be fast. So If you have a site with a 100000-row table, you should really spend time analyzing possible indexes and possibly consider rewriting queries to optimize your application.