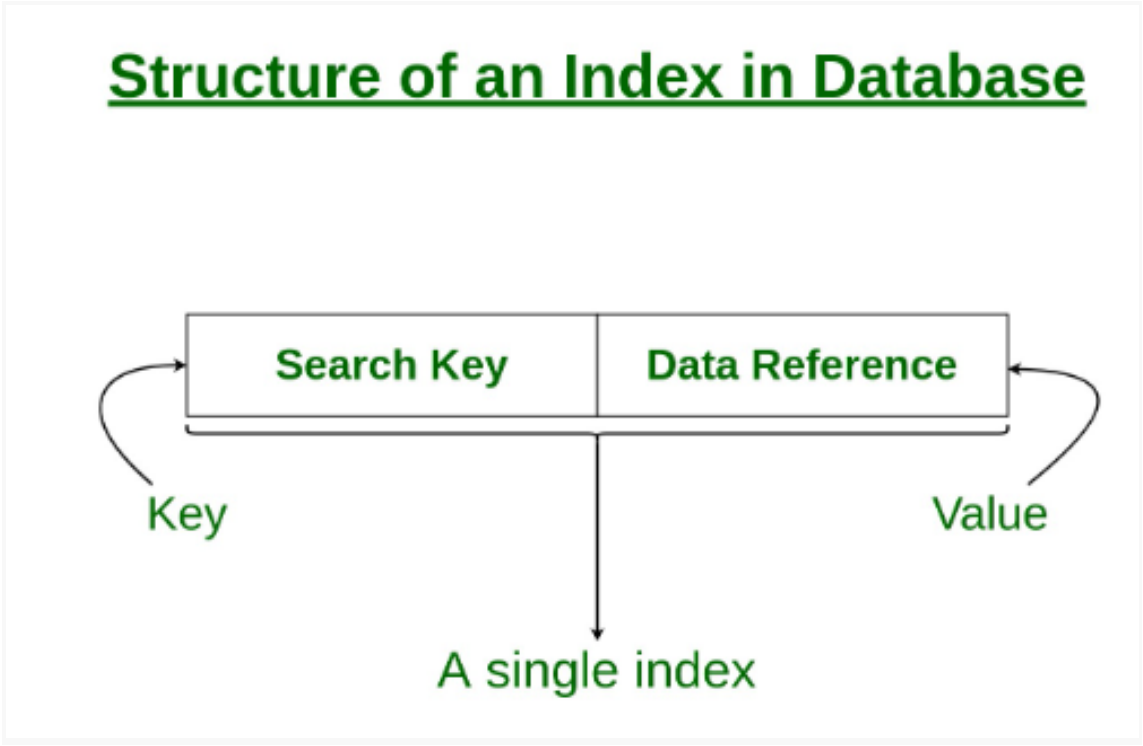
Indexing:

Indexes are used to find rows with specific column values quickly. Without an index, MySQL must begin with the first row and then read through the entire table to find the relevant rows. The larger the table, the more this costs. If the table has an index for the columns in question, MySQL can quickly determine the position to seek to in the middle of the data file without having to look at all the data. This is much faster than reading every row sequentially.




```
// Creating Index
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

```
// Dropping Index
ALTER TABLE table_name DROP INDEX index_name;
```

Show Indexes

```
SHOW INDEX FROM OrderItem;

EXPLAIN SELECT * FROM OrderItem;

EXPLAIN SELECT * FROM OrderItem WHERE OrderID = 551;

EXPLAIN SELECT * FROM OrderItem WHERE ProductID = 62;
```

Removing an Index:

ALTER TABLE Table_Name **DROP INDEX** Index_Name;

Renaming the indexes:

Syntax:

ALTER INDEX old_Index_Name RENAME **TO** new_Index_Name;

Altering an Index:

ALTER INDEX Index_Name **ON** Table_Name REBUILD;

When should indexes be created:

- A column contains a wide range of values.
- A column does not contain a large number of null values.
- One or more columns are frequently used together in a where clause or a join condition.

- If you most often insert, update, and delete records, then the fewer indexes associated with the table, the better the performance.
- If you most often retrieve records, you must look further to define the criteria for retrieving records and create indexes to improve the performance of these retrievals.

When should indexes be avoided:

- The table is small.
- The columns are not often used as a condition in the query.
- The column is updated frequently.

Before you create indexes for a database table, consider how you will use the table. The two most common operations on a table are to:

1) Insert, update, and delete records
2) Retrieve records

– If you most often insert, update, and delete records, then the fewer indexes associated with the table, the better the performance. This is because the driver must maintain the indexes as well as the database tables, thus slowing down the performance of record inserts, updates, and deletes. It may be more efficient to drop all indexes before modifying a large number of records, and re-create the indexes after the modifications.

– If you most often retrieve records, you must look further to define the criteria for retrieving records and create indexes to improve the performance of these retrievals.

Rules to help you decide which indexes to create:

1) If your record retrievals are based on one field at a time , create an index on these fields.

2) If your record retrievals are based on a combination of fields, look at the combinations.

3) If the comparison operator is OR , an index does not help performance. Therefore, you need not create one.

4) If the retrieval conditions contain both AND and OR comparison operators, you can use an index if the OR conditions are grouped.

5) If the AND conditions are grouped, an index does not improve performance.