Indexing is a very useful technique which helps in optimizing the search time in database query. Indexing improves database performance by minimizing the number of disc visits required to fulfil a query. It is a data structure technique used to locate and quickly access data in databases.

Several database fields are used to generate indexes. The main key or candidate key of the table is duplicated in the first column, which is the Search key. To speed up data retrieval, the values are also kept in sorted order. It should be highlighted that sorting the data is not required. The second column is the Data Reference or Pointer which contains a set of pointers holding the address of the disk block where that particular key value can be found.

* **Access Types**: This refers to the type of access such as value based search, range access, etc.
* **Access Time**: It refers to the time needed to find particular data element or set of elements.
* **Insertion Time**: It refers to the time taken to find the appropriate space and insert a new data.
* **Deletion Time**: Time taken to find an item and delete it as well as update the index structure.
* **Space Overhead**: It refers to the additional space required by the index.

There are several different index types used by databases, including primary, secondary, clustered, and non-clustered indexes. Based on the particular needs of the database system, each form of index offers benefits and drawbacks.

* **Improved Query Performance:** Indexing enables faster data retrieval from the database. The database may rapidly discover rows that match a specific value or collection of values by generating an index on a column, minimising the amount of time it takes to perform a query.
* **Efficient Data Access:** Indexing can enhance data access efficiency by lowering the amount of disk I/O required to retrieve data. The database can maintain the data pages for frequently visited columns in memory by generating an index on those columns, decreasing the requirement to read from disk.
* **Optimized Data Sorting:**Indexing can also improve the performance of sorting operations. By creating an index on the columns used for sorting, the database can avoid sorting the entire table and instead sort only the relevant rows.
* **Consistent Data Performance:** Indexing can assist ensure that the database performs consistently even as the amount of data in the database rises. Without indexing, queries may take longer to run as the number of rows in the table grows, while indexing maintains roughly consistent speed.
* By ensuring that only unique values are inserted into columns that have been indexed as unique, indexing can also be utilized to ensure the integrity of data. This avoids storing duplicate data in the database, which might lead to issues when performing queries or reports.

Overall, indexing in databases provides significant benefits for improving query performance, efficient data access, optimized data sorting, consistent data performance, and enforced data integrity

"I am using a dataset provided by the government, which includes information on animal hospitals nationwide. The dataset consists of five columns: hospital number, unique identification number of the local government, licensing date, full address, and business name. It is an Excel file with a total of 9687 rows, which include duplicate entries of the same hospitals.

To create a table, I removed the licensing date column as it is considered unnecessary for general citizens searching for hospital existence or location information. I used Oracle 11 as the DBMS and utilized a Docker image to set up the environment. I created a new user named 'saeyeon' instead of the system user. I removed any duplicate hospital entries or rows with missing information, resulting in a table with 8714 rows.

The primary key is set as 'id,' and the 'address' column is designated as a unique key. I set the 'address' column as unique to consider hospitals with the same address as the same business entity. Additionally, I created indexes on the 'name' and 'address' columns. The 'address' index is based on the first word of the address.

The table without indexes only has the 'name' and 'address' indexes, while the rest remains the same.

I will examine the search results for an animal hospital named 'Heemang Animal Hospital.' Without any indexes, a full scan is performed, and the IO cost and CPU cost are as shown in the right picture. Now, let's consider the case with indexes. An index scan is performed instead of a full scan, and we can observe a decrease in both CPU cost and IO cost compared to the previous case. However, it seems that the improvement is not as significant as expected. Let's look at another example.

This time, I searched for rows where the first word of the address value is 'Seoul.' Without an index, a full scan is again performed. Now, let's see the case with indexes. As shown in the picture, we can observe a much larger difference in CPU cost and IO cost compared to the previous example.

As demonstrated, indexing plays a crucial role in achieving efficient performance. With indexes, scans can be much more efficient, leading to cost savings. Additionally, the benefits of indexing become more prominent as the dataset size increases or when executing more complex queries."