Q: In this query:

SELECT name FROM large\_table WHERE name = 'ahmed;

If there is no index on the column name and if the table has these columns in order { id, age, city, name, dep}, while performing the full table scan will it pass through id, age, and city values or it knows where the 'name' is located and will directly go to it and skip the other columns

A: In most traditional database systems, when performing a full table scan, the database reads each row sequentially, and for each row, it reads all columns in the order they appear in the table schema. This is a general description of how full table scans work in many databases.

**However**, it's essential to recognize that database systems can vary in their behavior, and advanced optimizations or storage mechanisms may be in place in some systems to optimize query execution. Therefore, the specific behavior of a database during a full table scan can depend on the database system, its configuration, and any optimizations it employs.

If you have specific questions about the behavior of a particular database system, such as MariaDB, I recommend referring to the official documentation or consulting with database experts who are familiar with the intricacies of that system. They can provide precise information about how that database handles full table scans and column access during query execution.

Q: What is the impact of database Index on regular expression queries in MySQL?

A:<https://stackoverflow.com/questions/19896935/impact-of-database-index-on-regular-expression-queries-in-mysql> (take a look at the related topics section)

Q: Explain Bitmap Index Scan vs Index Scan vs Table Scan.

ChatGPT A:

In the context of relational database systems, such as PostgreSQL or Oracle, the terms "Bitmap Index Scan," "Index Scan," and "Table Scan" refer to different methods used by the database engine to retrieve data from tables. Let's explore each of these methods in detail:

**- Table Scan:**

Description**:** A table scan, also known as a sequential scan or full table scan, involves scanning the entire table to find the required rows. It reads each row in sequence, regardless of the values in indexed columns.

When to Use: Table scans are typically used when there is no suitable index for the query or when a large portion of the table needs to be accessed. However, they can be inefficient for large tables.

Example of a table scan:

SELECT \* FROM employees WHERE department\_id = 10;

**- Index Scan:**

Description: An index scan involves scanning the index of a table to locate the rows that satisfy the query conditions. Once the index identifies the rows, it retrieves them from the actual table.

When to Use: Index scans are beneficial when a query can be satisfied using an existing index. They can significantly speed up query performance, especially for large tables.

Example of a table scan:

SELECT \* FROM employees WHERE employee\_id = 1001;

**- Bitmap Index Scan:**

**Description:** A bitmap index scan is a type of index scan that uses a bitmap to combine multiple index conditions efficiently. It is particularly useful when dealing with multiple conditions (AND or OR) and can reduce the overhead of combining multiple index results.

When to Use: Bitmap index scans are effective when a query involves multiple indexed conditions that can be combined using bitmaps.

Example of a table scan:

SELECT \* FROM employees WHERE department\_id = 10 AND salary > 50000;

It's **important** to note that the database optimizer decides which method to use based on factors such as available indexes, the complexity of the query, and statistics about the distribution of data in the tables.

Understanding and analyzing query execution plans can provide insights into which method the database engine has chosen for a particular query. This information is often obtained using database-specific tools or by examining the execution plan generated by the database engine for a given query.

Q: Explain Index Scan vs Index Only Scan.

ChatGPT A:

In relational database systems, particularly PostgreSQL, the terms "Index Scan" and "Index Only Scan" refer to two different methods the database engine uses to retrieve data from an index. Let's explore each of these methods:

* **Index Scan:**

**Description:** An Index Scan involves reading data from an index structure and then retrieving the corresponding rows from the table based on the index information. This scan may require accessing both the index and the actual table to fetch the necessary data.

**When to Use:** Index Scans are used when the query requires fetching columns that are not present in the index. If the query needs to retrieve multiple columns or if the conditions in the WHERE clause involve non-indexed columns, an Index Scan is likely to be used.

**Example of an Index Scan:**

SELECT \* FROM employees WHERE department\_id = 10;

* **Index Only Scan:**

**Description:** An index-only scan, on the other hand, is a more optimized form of index retrieval. It occurs when all the columns needed for a query are present in the index itself, and there is no need to access the actual table. This can significantly reduce the I/O (Input/Output) overhead associated with fetching data from the table.

**When to Use:** Index-only scans are beneficial when the index contains all the columns needed for a query, and there is no need to fetch additional data from the table. This is often the case when using covering indexes.

**Example of an Index-Only Scan**

SELECT department\_id FROM employees WHERE department\_id = 10;

**Key Differences:**

* In an Index Scan, the database engine needs to access both the index and the table to fetch the required data. This can result in additional I/O operations, especially if the table is large.
* In an index-only scan, the database engine can satisfy the query entirely from the index itself without needing to access the table. This can lead to improved performance, as it reduces the amount of data that needs to be read from disk.

Q: Creating an index on a json column (scenarios and performance)