fingerTips

Module-10

Indexes in SQL

1) Indexes in SQL

In SQL, indexes are data structures that provide quick access to rows within a table. They are used to improve the performance of queries by reducing the amount of data that needs to be scanned and filtered.

When a query is executed against a table, the database engine scans through the rows of the table to find the ones that match the query's criteria. Without an index, the database engine has to perform a full table scan, which can be slow and resource-intensive, especially for large tables.

Indexes work by creating a separate structure that stores a copy of some or all of the table's data in a way that is optimized for search and retrieval. The index contains pointers to the corresponding rows in the table, so when a query is executed, the database engine can use the index to quickly locate the relevant rows without scanning the entire table.

Indexes can be created on one or more columns in a table, and they can be either unique or non-unique. A unique index ensures that no two rows in the table have the same value for the indexed column(s), while a non-unique index allows duplicate values.

2) Understand indexes in SQL using real life example

Let's say we have a book with hundreds of pages and you want to find a specific topic, such as "How to make pizza". Without an index, you would have to flip through the pages one by one, which can be time-consuming and frustrating.

However, if the book has an index at the beginning or end, we can quickly find the page number where the topic is discussed, and then go directly to that page without having to search through the entire book. The index serves as a map that tells you where to find specific information in the book, and saves you time and effort.

Indexes in SQL also works in similar way.

3) Applications of indexes in SQL

Some common applications of indexes in SQL are:

i. Improving query performance: By reducing the amount of data that needs to be scanned and filtered, indexes can help to speed up queries, especially for large tables.
ii. Enforcing uniqueness constraints: Unique indexes can be used to enforce uniqueness constraints on one or more columns in a table.
iii. Optimizing joins: Indexes can be used to optimize joins between tables by allowing the database engine to quickly locate the matching rows.
iv. Supporting full-text search: Full-text indexes can be used to enable full-text search capabilities in a database, allowing users to search for specific words or phrases within text fields.

Indexes are an essential tool for optimizing the performance of SQL queries, and are widely used in a variety of applications, including e-commerce, finance, healthcare, and more.

4) Creating index on menu_id column of table menu

By creating an index on the "menu_id" column or the combination of columns "menu_id", "restro_id", and "food_id", we can improve the performance of queries that involve filtering or sorting on these columns.

If we want to retrieve the menu items for a particular restaurant with a specific food item, we can use the following SQL query:

CREATE INDEX idx_menu_id ON menu_1 (menu_id);

Now, since we have created an index on the "menu_id" column of the "menu" table, we run a query that filters on that column, the database engine will use the index to quickly find the matching rows.

Example:

```
SELECT f_id, price
FROM menu_1
WHERE menu_id = 39;
```

5) Create an index on the combination of columns "menu_id", "r_id", and "f_id" of table menu

We have created an index on the combination of columns "menu_id", "r_id", and "f_id", and we run a query that filters or sorts on those columns, the database engine will use the index to optimize the query.

```
CREATE INDEX idx_menu_restro_food ON menu (menu_id, restro_id, food_id);
```

Example:

```
SELECT menu_id, r_id, f_id, price
FROM menu_1
WHERE r_id = 3
AND f_id = 7
ORDER BY price DESC;
```