Index Overview

by Sophia

WHAT'S COVERED

This lesson explores the different types of indexes in databases, in two parts. Specifically, this lesson will cover:

1. Understanding Indexes

By default, primary keys and columns with the UNIQUE constraint have indexes created for them. You can also create indexes on any other columns where you think having one might be useful.

The purpose of a database **index** is to improve the efficiency of data retrieval operations. Based on the values of one or more columns, it quickly retrieves specific rows from within a database table. The database system organizes data by creating an index on a column or set of columns to speed up retrieval, search, and filtering.

An index allows the database engine to quickly pinpoint the relevant rows when a query is executed based on indexed columns, reducing the need to search through the entire table. Large datasets are processed more quickly as a result, especially when dealing with large and complex queries. Indexes can also be helpful when running UPDATE and DELETE statements that include filtering criteria.

Even though indexes greatly improve read operations, they can sometimes negatively affect the performance of data modification operations (such as inserts, updates, and deletes) since the indexes must be updated whenever the underlying data changes. The right balance between query performance and data modification efficiency can be achieved by strategically creating indexes for frequently searched and retrieved columns.

EXAMPLE Let us look at a query run on our track table:

```
SELECT * FROM track WHERE album_id = 5;
```

If no indexes on the album_id column are created for this table, the database would have to scan row by row through the entire table to find all of the matching rows. If there were many rows in this table and we only expected to have a few rows returned, this can be quite an inefficient search. Imagine if there were one million rows in the table, and this album_id did not even exist. This means that the database would have to search through all one million rows to identify no matching values. If the database has an index on the album_id column, however, the database would be able to find the matching data much more efficiently.

KEY CONCEPT

The concept of database indexes is quite similar to the alphabetical index at the end of a book. With a book, an individual can quickly scan through the index to find the topics they're interested in and what page to turn to. This is a much faster approach than reading through the entire book to find the content you want. The usefulness of a book index depends on the person who creates the topic list. Similarly, the database developer must determine what indexes could be useful.

After an index is created, the database system uses it automatically to assist with data retrieval and updates it whenever the table data is modified. Indexes can be useful not only in SELECT statements with joins on the indexed columns but also in UPDATE and DELETE commands that have filtering criteria.

We do not want to index every single column in a database, for two reasons. One is that creating an index can take a long time. Another is that maintaining the index uses system resources. Every INSERT, DELETE, and UPDATE statement that affects the index uses memory and CPU resources to keep the index in synch with the table. For that reason, if an index is not frequently used, it should ideally be removed.

TERM TO KNOW

Index

A database object that improves the speed of data retrieval operations on a database table at the expense of additional writes and storage space.

2. Types of Indexes

Optimizing and managing databases effectively requires an understanding of different index types. Understanding each index type's strengths and limitations can help database administrators, and developers make informed decisions when optimizing data access and query performance.

Each index type has its own benefits and drawbacks that make it ideal for certain kinds of searches and not so ideal for others. Database professionals who understand the types can select the best index type for a situation based on the nature of the data and the expected query workload. With this proactive approach, queries are more efficient, data is retrieved more quickly, and database efficiency is improved.

Most databases will implement indexes using one of the following data structures:

Hash Index	Data Types: Suitable for exact match queries. Queries: Efficient for comparisons using the = operator.

	Best For: Columns with uniform distribution and where exact matches are frequent. This type can handle only simple equality operators, but it does so very well.
B-Tree Index	Data Types: Suitable for almost all data types. Queries: Ideal for equality and range queries. Best For: Frequently accessed columns, such as primary keys or columns used in WHERE clauses for equality comparisons. Columns where most records contain a unique value. This is the most common type of index. It organizes data in a hierarchical format, like an upside-down tree. It takes about the same amount of time to access any row, regardless of the index size. For example, with one million rows, it will take fewer than 20 searches to find the data, compared to one million searches if the search were done sequentially.
Bitmap Index	Data Types: Suitable for columns with a low cardinality (a small number of distinct values). Queries: Great for querying Boolean or categorical data. Best For: Columns with few distinct values (e.g., gender, Boolean flags).
Full-Text Index	Data Types: Designed for text data. Queries: Ideal for searching within text fields using keywords or phrases. Best For: Text-heavy columns like article content, product descriptions, etc.
Composite Index	Data Types: Any data types. Queries: Useful for queries involving multiple columns, especially when those columns are often used together in WHERE clauses. Best For: Queries that involve multiple columns in the WHERE clause or involve multiple conditions.
Spatial Index	Data Types: Geospatial data types (points, lines, polygons). Queries: Optimal for spatial queries, such as finding nearby locations or areas within a certain distance. Best For: Geospatial data, like maps, GPS coordinates, etc.

SUMMARY

In this lesson, you **understood that an index** within a database management system speeds up and improves data retrieval operations. Indexed columns make it easier to locate specific data rows based on their values. It minimizes the amount of time it takes to retrieve relevant data by reducing the need for full table scans. Indexes are especially important for maintaining acceptable query response times for large dataset databases. As well as maintaining data integrity, database indexes enforce unique constraints. You learned about the different **types of indexes** and what data types and query types they are optimized for. Some of the types you learned about include B-tree, hash, bitmap, spatial, composite, and full-text.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND FAITHE WEMPEN (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR **TERMS OF USE**.

TERMS TO KNOW

Index

A database object that improves the speed of data retrieval operations on a database table at the expense of additional writes and storage space.