What is an Index?

An index is a database object that is created on a table to improve the speed of data retrieval operations. It works much like an index in a book, allowing the database to quickly find rows that match certain criteria without scanning the entire table.

Creating an Index

Let's create a table and then add an index to it.

Creating the Books Table

```
CREATE TABLE Books (
    BookID INT AUTO_INCREMENT PRIMARY KEY,
    Title VARCHAR(255) NOT NULL,
    PublisherID INT,
    ISBN VARCHAR(13) UNIQUE NOT NULL,
    YearPublished YEAR CHECK (YearPublished >= 1450 AND YearPublished <= YEAR(CURDATE())),
    Genre VARCHAR(50),
    CopiesAvailable INT DEFAULT 1 CHECK (CopiesAvailable >= 0)
);
```

Adding Indexes to Improve Query Performance

Create an index on the Title column

```
CREATE INDEX idx_title ON Books (Title);
```

Create an index on the ISBN column

```
CREATE UNIQUE INDEX idx_isbn ON Books (ISBN);
```

How Indexes Improve Query Performance

Indexes are particularly useful for columns that are frequently searched, such as Title and ISBN in the Books table. Here’s how an index improves performance:

Speeding Up SELECT Queries

When you run a query to find books by title:

SELECT * FROM Books WHERE Title = 'The Great Gatsby';

Without Index: The database engine performs a full table scan, checking each row to see if the Title matches 'The Great Gatsby'. This is slow, especially if the table has a large number of rows.

With Index: The database uses the idx_title index to quickly locate the rows where the Title is 'The Great Gatsby', significantly speeding up the query.

Ensuring Uniqueness and Quick Searches on Unique Columns

The ISBN column is unique, and searching by ISBN should be fast:

SELECT * FROM Books WHERE ISBN = '9781234567897';

Without Index: Again, a full table scan is required.

With Index: The database uses the idx_isbn index, which is a unique index, to quickly find the row with the specified ISBN.

Impact of Indexes on Write Operations

While indexes improve read performance, they can negatively impact write performance (INSERT, UPDATE, DELETE) because the index must be updated whenever the table data changes.

INSERT Operations

The database needs to insert the data into the table and also update the index.

More indexes mean more overhead during inserts.

UPDATE Operations

If an indexed column is updated, the database must update the index.

This can slow down updates, especially for large tables with many indexes.

DELETE Operations

Similar to updates, the database must remove entries from the indexes when rows are deleted.

This adds to the overhead.

Dropping an Index

If an index is no longer needed or is negatively impacting performance, you can drop it.

Drop the index on the Title column

DROP INDEX idx_title ON Books;

Drop the index on the ISBN column

DROP INDEX idx_isbn ON Books;

Impact on Query Execution Before and After Dropping Index

Before Dropping Index

Using EXPLAIN to Analyze Query Plan

EXPLAIN SELECT * FROM Books WHERE Title = 'The Great Gatsby';

The query optimizer uses the idx_title index.

The query plan shows an index lookup, which is faster than a full table scan.

➔ After Dropping Index

Using EXPLAIN to Analyze Query Plan Again

EXPLAIN SELECT * FROM Books WHERE Title = 'The Great Gatsby';

Without the idx_title index, the query optimizer reverts to a full table scan.

The query plan shows a full table scan, which is slower, especially with large datasets.