Day 10: SQL Notes Basic to Advanced

Todays Topic: SQL Indexes

- Types of SQL Indexes
- Creating Indexes
- Dropping Indexes
- Clustered vs. Non-Clustered Indexes

SQL Indexes

SQL indexes are database objects that improve the speed of data retrieval operations on a table at the cost of additional storage and maintenance overhead.

They work similarly to indexes in books, where they help you quickly locate specific information without scanning the entire content.

- Improves query performance by speeding up data retrieval.
- Creates a data structure to quickly locate rows in a table.
- Can be applied to one or more columns in a table.
- Helps in optimizing search operations, but may impact write performance.

10.1 Types of Indexes:

- **Clustered Index:** Determines the physical order of data in the table. A table can have only one clustered index. The primary key is usually a clustered index by default.

- **Non-Clustered Index:** Contains a separate structure from the table's data, with pointers to the actual data rows. A table can have multiple non-clustered indexes.

- **Unique Index:** Ensures that all values in the index key are unique. It is often used to enforce uniqueness constraints on columns.

- **Composite Index:** An index on multiple columns, which helps in queries that filter or sort based on those columns.

- **Full-Text Index:** Used for efficient text searches within large text columns, such as finding words or phrases in a TEXT field.

- **Spatial Index:** Used for spatial data types in databases that support geographic data, enabling fast querying of spatial information.

10.2 Creating Indexes:

Creating indexes in SQL involves defining an index on one or more columns of a table to improve query performance. Here's a step-by-step guide to creating various types of indexes:

1. Basic Index Creation

To create a basic index on a single column, use the following syntax:

```sql
CREATE INDEX index_name
ON table_name (column_name);
```

Example:

```sql
CREATE INDEX idx_employee_lastname
ON employees (lastname);
```

2. Unique Index

A unique index ensures that all values in the indexed column(s) are unique. It is often used to enforce uniqueness constraints.

Syntax:

```sql
CREATE UNIQUE INDEX index_name
ON table_name (column_name);
```

10.2 Creating Indexes:

Creating indexes in SQL involves defining an index on one or more columns of a table to improve query performance. Here's a step-by-step guide to creating various types of indexes:

1. Basic Index Creation

To create a basic index on a single column, use the following syntax:

```
CREATE INDEX index_name
ON table_name (column_name);
```

Example:

```
CREATE INDEX idx_employee_lastname
ON employees (lastname);
```

2. Unique Index

A unique index ensures that all values in the indexed column(s) are unique. It is often used to enforce uniqueness constraints.

Syntax:

```
CREATE UNIQUE INDEX index_name
ON table_name (column_name);
```

Example:

```sql
CREATE UNIQUE INDEX idx_employee_email
ON employees (email);
```

3. Composite Index

A composite index (or multi-column index) is created on multiple columns. It is useful for queries that filter or sort on multiple columns.

Syntax:

```sql
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

Example:

```sql
CREATE INDEX idx_employee_department_salary
ON employees (department_id, salary);
```

4. Clustered Index

A clustered index determines the physical order of data in the table. A table can have only one clustered index.

Syntax:

```sql
CREATE CLUSTERED INDEX index_name
ON table_name (column_name);
```

Example:

```
CREATE CLUSTERED INDEX idx_employee_id
ON employees (employee_id);
```

Note: In some databases like MySQL, the primary key is automatically used as the clustered index if you don't explicitly define one.

5. Full-Text Index

A full-text index is used for efficient searching of text within large text columns.

SQL Syntax (SQL Server):

```
CREATE FULLTEXT INDEX ON table_name (column_name)
KEY INDEX index_name;
```

Example:

```
CREATE FULLTEXT INDEX ON articles (content)
KEY INDEX idx_article_id;
```

6. Spatial Index

A spatial index is used for spatial data types to enable fast querying of geographic data.

SQL Syntax (SQL Server):

```
CREATE SPATIAL INDEX index_name
ON table_name (column_name);
```

Example:

```
CREATE SPATIAL INDEX idx_location
ON locations (geo_data);
```

10.3 Dropping an Index

Dropping indexes is a straightforward process, but it's important to understand the implications on your database performance and structure. Here’s how you can drop indexes in various SQL databases:

Syntax:

```
DROP INDEX index_name ON table_name;
```

Example:

```
DROP INDEX idx_employee_lastname ON employees;
```

Considerations Before Dropping an Index

- Impact on Performance: Dropping an index can slow down queries that used it.
- Dependencies: Verify the index is not part of a primary key or unique constraint.
- Testing: Test performance changes in a staging environment first.

10.4 Clustered vs. Non-Clustered Indexes

Aspect	Clustered Index	Non-Clustered Index
Definition	Determines the physical order of data in the table.	Creates a separate structure from the table's data.
Number per Table	Only one clustered index per table is allowed.	Multiple non-clustered indexes can be created on a table.
Data Storage	Data rows are stored in the order of the clustered index key.	Indexes are stored separately from the data rows.
Index Structure	The index is the actual table with sorted data.	The index contains pointers to the actual data rows.
Key Usage	Typically used on primary keys or unique columns.	Can be used on any column(s) to improve query performance.
Performance	Efficient for range queries and sorting operations.	Efficient for exact matches and queries involving joins.
Impact on Data	Alters the physical arrangement of the data on disk.	Does not alter the physical arrangement of the data.
Query Speed	Often faster for queries involving sorting and range queries.	Speeds up searches, especially for columns frequently queried.
Storage Overhead	Can lead to fragmentation of data.	Requires additional storage for the index structure.
Maintenance	Updates to the index can be costly in terms of performance.	Maintenance involves updating the index separately from the data.
Typical Use Case	Best for columns frequently used in sorting or range queries.	Best for columns frequently used in search conditions or joins.

SQL Has Been Completed
Total 95 Pages

From Tomorrow I'll Post Python Notes Series

Download the PDF from My Telegram Channel

Curious Coder
124,362 subscribers

share link
https://t.me/Curious_Coder
description
Do join for coding resources, Handwritten notes & Quizzes! 🧑‍💻