SQL Indexing Strategies

Anish Chakravorty
Follow me on LinkedIn

What Are Indexes in SQL?

Indexes are like a table of contents in a book—they help your database find data quickly without scanning the entire table.

Benefits of Indexing:

- Faster data retrieval.
- Optimized sorting and filtering.
- Better performance for large datasets.

💡 Think of indexes as shortcuts for SQL queries!

Types of Indexes

1. Clustered Index:

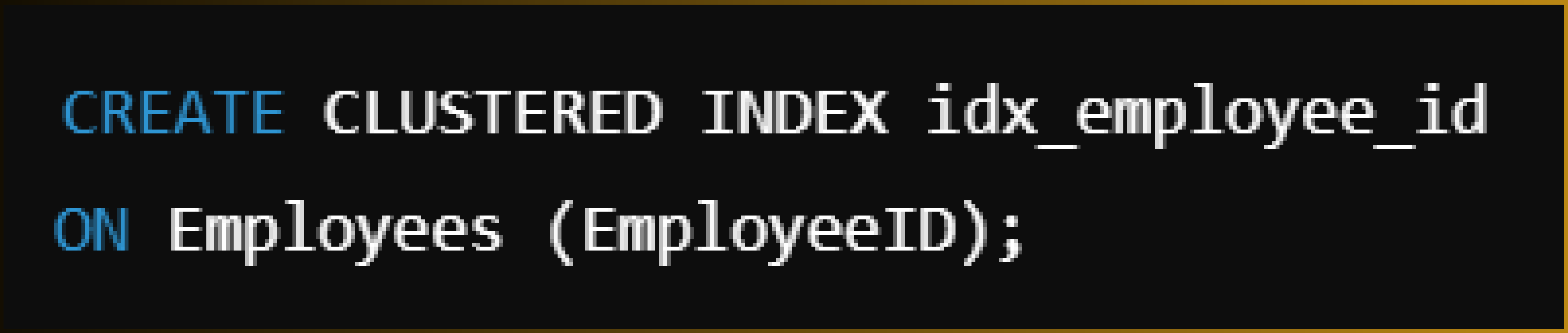
- Determines the physical order of data in the table.
- Only one per table (usually on the primary key).
- Faster for range queries.

2. Non-Clustered Index:

- Separate from the actual data; includes pointers to the data.
- Multiple non-clustered indexes allowed per table.
- Great for frequently filtered or sorted columns.

Clustered Index Explained

- The table data is stored physically in the order of the indexed column.
- Perfect for range queries (e.g., dates, IDs).

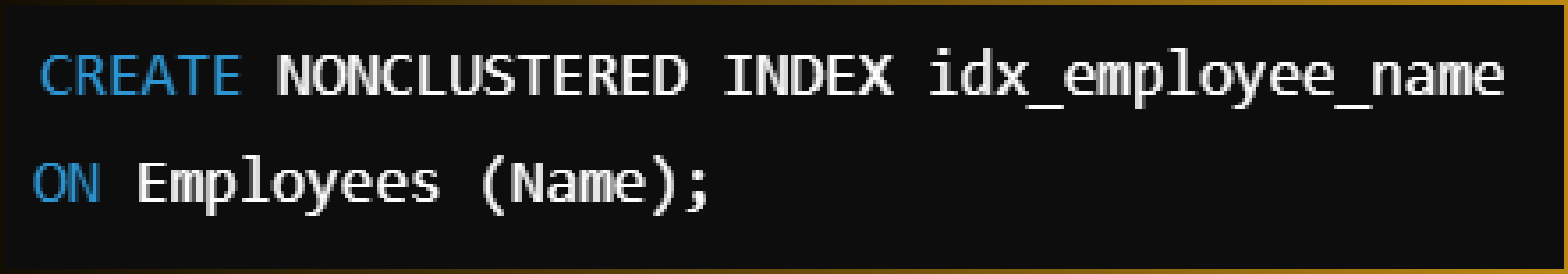


```sql
CREATE CLUSTERED INDEX idx_employee_id
ON Employees (EmployeeID);
```

Pro Tip: Use clustered indexes for primary keys or frequently queried ranges.

Non-Clustered Index Explained

- The index contains a separate structure with pointers to the actual data.
- Allows multiple indexes for different queries.



```sql
CREATE NONCLUSTERED INDEX idx_employee_name
ON Employees (Name);
```

Pro Tip: Use non-clustered indexes for columns in WHERE, ORDER BY, or JOIN clauses.

Key Differences

Feature	Clustered Index	Non-Clustered Index
Physical Data Order	Matches the index	Separate from the index
Number Per Table	Only 1	Multiple allowed
Performance	Faster for range queries	Faster for selective filters
Use Case	Primary Key	Frequently filtered columns

When to Use Clustered Indexes?

- Primary keys and unique identifiers.
- Columns with sequential or range-based data (e.g., dates, IDs).
- Tables with frequent range queries.

When to Use Non-Clustered Indexes?

- Columns frequently used in WHERE or ORDER BY.
- Supporting columns for complex joins.
- Large tables requiring multiple search patterns.

Real-Life Optimization Example

Scenario: Find employees earning more than 100,000.

Without Index: (Full Table Scan)

```sql
SELECT * FROM Employees WHERE Salary > 100000;
```

With Index: (Non-Clustered Index on Salary)

```sql
CREATE NONCLUSTERED INDEX idx_salary ON Employees (Salary);
SELECT * FROM Employees WHERE Salary > 100000;
```

Result: Query runs 5x faster!

Key Takeaways

- Clustered Index: Use for primary keys or sequential data.
- Non-Clustered Index: Use for frequently filtered or sorted columns.
- Index wisely to balance performance and storage costs.

Follow me to get more **Information** and content like this.

Anish Chakravorty

Follow me on LinkedIn