# Database

**Index**

# **1.** Phân biệt cluester index và non-cluster index ? So sánh về perfomance khi sử dụng 2 loại index này ? Giải thích nguyên nhân**?**

### **Clustered Index:**

1. **Cấu trúc**:
   * **Clustered index** xác định thứ tự vật lý của dữ liệu trong bảng, nghĩa là các hàng của bảng được lưu trữ trên đĩa theo đúng thứ tự mà chỉ mục chỉ định.
   * Một bảng chỉ có thể có **một** clustered index vì không thể lưu trữ dữ liệu theo nhiều thứ tự khác nhau.
2. **Cách hoạt động**:
   * Khi bạn tạo một clustered index trên một cột (hoặc tập hợp các cột), dữ liệu sẽ được sắp xếp lại theo thứ tự của cột đó.
   * Mỗi nút lá của cây chỉ mục chứa **dữ liệu thực tế** của bảng, vì vậy khi truy vấn dữ liệu thông qua clustered index, hệ thống không cần thực hiện tìm kiếm thêm (no lookup) vì chỉ mục chứa luôn dữ liệu.
3. **Ưu điểm**:
   * Truy vấn các phạm vi dữ liệu liên tiếp (range query) rất nhanh vì dữ liệu đã được sắp xếp.
   * Truy vấn dựa trên chỉ mục trả về nhanh chóng, vì không cần thực hiện thêm bước tìm kiếm dữ liệu (no additional lookup).
4. **Nhược điểm**:
   * Việc **INSERT, UPDATE**, và **DELETE** có thể tốn kém hơn khi phải sắp xếp lại dữ liệu vật lý theo thứ tự của clustered index.
   * Chỉ có thể có **một** clustered index trên mỗi bảng do dữ liệu chỉ có thể được sắp xếp theo một cách.

### **Non-Clustered Index:**

1. **Cấu trúc**:
   * **Non-clustered index** không thay đổi thứ tự vật lý của dữ liệu trong bảng, nó chỉ tạo ra một bảng chỉ mục riêng biệt với các con trỏ trỏ đến vị trí thực tế của dữ liệu trong bảng.
   * Một bảng có thể có **nhiều** non-clustered index vì nó không ảnh hưởng đến việc lưu trữ dữ liệu thực tế.
2. **Cách hoạt động**:
   * Non-clustered index tạo ra một cấu trúc giống cây (tree) có chứa giá trị chỉ mục và con trỏ đến vị trí của dữ liệu thực tế trong bảng (gọi là **row locator** hoặc **RID**).
   * Khi một truy vấn sử dụng non-clustered index, hệ thống phải thực hiện một bước bổ sung để tìm ra vị trí vật lý của dữ liệu trong bảng (điều này gọi là **lookup**).
3. **Ưu điểm**:
   * Có thể tạo nhiều non-clustered index trên cùng một bảng để tối ưu hóa các truy vấn khác nhau.
   * Non-clustered index không làm thay đổi cách dữ liệu được lưu trữ vật lý, do đó không ảnh hưởng nhiều đến các hoạt động như INSERT, UPDATE, hoặc DELETE.
4. **Nhược điểm**:
   * Non-clustered index truy cập dữ liệu chậm hơn clustered index vì cần thực hiện thêm bước tìm kiếm (lookup) sau khi tìm thấy giá trị chỉ mục.
   * Tốn nhiều không gian lưu trữ hơn, vì mỗi non-clustered index sẽ lưu trữ cả các con trỏ đến dữ liệu thực tế.

### **Giải thích nguyên nhân:**

* **Truy vấn nhanh hơn với Clustered Index**: Clustered index lưu trữ dữ liệu theo thứ tự chỉ mục, do đó, khi truy vấn dữ liệu, hệ thống có thể truy cập trực tiếp vào dữ liệu mà không cần bước bổ sung để tìm kiếm vị trí dữ liệu. Điều này đặc biệt hiệu quả cho các truy vấn phạm vi (ví dụ: SELECT \* FROM employees WHERE salary BETWEEN 5000 AND 10000), vì dữ liệu liên quan được lưu trữ liên tục trên đĩa.
* **Lookup cần thiết với Non-Clustered Index**: Non-clustered index chỉ lưu giá trị chỉ mục và con trỏ đến dữ liệu thực tế, vì vậy sau khi tìm thấy giá trị trong non-clustered index, hệ thống cần thực hiện thêm một bước để lấy dữ liệu từ bảng. Điều này làm cho non-clustered index chậm hơn so với clustered index, đặc biệt là với các truy vấn có phạm vi lớn.
* **INSERT/UPDATE/DELETE chậm hơn với Clustered Index**: Vì dữ liệu phải được sắp xếp theo thứ tự của clustered index, bất cứ khi nào bạn chèn hoặc cập nhật một hàng, hệ thống có thể cần sắp xếp lại dữ liệu trên đĩa, điều này có thể tốn thời gian. Trong khi với non-clustered index, vì thứ tự vật lý không quan trọng, các thao tác này thường nhanh hơn.

### **Khi nào nên sử dụng Clustered Index và Non-Clustered Index:**

* **Clustered Index**:
  + Phù hợp với các bảng mà dữ liệu thường được truy vấn theo thứ tự (ví dụ: các truy vấn phạm vi).
  + Thích hợp cho các cột thường xuyên được sử dụng để tìm kiếm và trả về nhiều dữ liệu liên tục (ví dụ: cột ngày, cột ID tuần tự).
  + Không nên sử dụng clustered index trên các cột có giá trị thường xuyên thay đổi, vì sẽ tốn chi phí sắp xếp lại dữ liệu.
* **Non-Clustered Index**:
  + Phù hợp cho các bảng có nhiều truy vấn tìm kiếm với các điều kiện khác nhau, nơi cần nhiều chỉ mục.
  + Sử dụng tốt cho các bảng lớn với các cột không cần thiết phải sắp xếp lại dữ liệu vật lý.
  + Thích hợp cho các truy vấn chính xác (exact match query) như tìm theo ID, username, v.v.

# 2. Có giới hạn việc đánh bao nhiêu index trong 1 table hay không ? Giải thích?

### **Giới hạn số lượng chỉ mục (Index) trong một bảng:**

Hầu hết các hệ quản trị cơ sở dữ liệu (DBMS) như MySQL, SQL Server, Oracle, PostgreSQL đều cho phép bạn tạo nhiều **Non-Clustered Index** trên một bảng. Tuy nhiên, mỗi DBMS đều có **giới hạn tối đa** về số lượng chỉ mục bạn có thể tạo. Số lượng chỉ mục tối đa phụ thuộc vào cấu hình cụ thể của từng hệ quản trị cơ sở dữ liệu, nhưng thường là rất lớn (từ vài trăm đến vài nghìn chỉ mục).

### **Lý do hạn chế số lượng chỉ mục:**

Mặc dù các DBMS hiện đại cho phép tạo ra nhiều chỉ mục trên một bảng, việc **tạo quá nhiều chỉ mục không phải lúc nào cũng là giải pháp tốt**, bởi vì có một số vấn đề về hiệu suất và tài nguyên:

#### **1. Tăng chi phí ghi (Write cost):**

* Mỗi khi bạn thực hiện một thao tác **INSERT**, **UPDATE**, hoặc **DELETE**, DBMS phải **cập nhật tất cả các chỉ mục liên quan**. Nếu bảng có nhiều chỉ mục, việc cập nhật mỗi chỉ mục sẽ tốn thêm tài nguyên và thời gian, làm chậm hiệu suất ghi dữ liệu.

#### **2. Tăng dung lượng lưu trữ:**

* Mỗi chỉ mục sẽ chiếm một lượng không gian lưu trữ trong cơ sở dữ liệu. Khi số lượng chỉ mục tăng lên, dung lượng lưu trữ cần thiết sẽ tăng theo, có thể dẫn đến việc tiêu tốn nhiều bộ nhớ hoặc dung lượng đĩa hơn.

#### **3. Hiệu suất truy vấn không phải lúc nào cũng được cải thiện:**

* Tạo quá nhiều chỉ mục có thể **gây khó khăn cho trình tối ưu hóa truy vấn** (query optimizer) trong việc lựa chọn chỉ mục phù hợp, vì không phải chỉ mục nào cũng luôn hiệu quả cho mọi loại truy vấn. Hệ thống có thể tốn thời gian để đánh giá chỉ mục nào nên được sử dụng, gây chậm trễ cho truy vấn.

#### **4. Overhead cho bảo trì chỉ mục:**

* Các chỉ mục cần được bảo trì, và các thao tác như **rebuild** hoặc **reorganize** chỉ mục sẽ tốn nhiều thời gian và tài nguyên hơn khi bạn có quá nhiều chỉ mục trên một bảng.

# 3. Việc đánh index dựa trên cơ sở nào?

Việc tạo chỉ mục (index) trong cơ sở dữ liệu dựa trên nhiều yếu tố và cần phải cân nhắc kỹ lưỡng để đạt được hiệu suất tối ưu cho các truy vấn. Dưới đây là những yếu tố chính cần xem xét khi quyết định đánh chỉ mục:

### **1. Mẫu truy vấn (Query Patterns)**

* **Cột thường xuyên được sử dụng trong WHERE clause**: Nếu một cột thường xuyên được sử dụng để lọc dữ liệu trong các truy vấn, thì nên tạo chỉ mục cho cột đó. Ví dụ, nếu bạn thường xuyên truy vấn theo CustomerID, thì nên tạo chỉ mục cho cột này.
* **Cột trong JOIN clause**: Các cột được sử dụng trong các phép nối giữa các bảng (JOIN) cũng nên được lập chỉ mục để cải thiện hiệu suất của các truy vấn JOIN.
* **Cột trong ORDER BY và GROUP BY**: Nếu một cột thường xuyên được sử dụng để sắp xếp (ORDER BY) hoặc nhóm (GROUP BY), việc tạo chỉ mục có thể cải thiện hiệu suất truy vấn.

### **2. Tần suất cập nhật (Update Frequency)**

* **Cột ít thay đổi**: Chỉ mục trên các cột ít thay đổi (như mã số khách hàng, địa chỉ email, v.v.) thường sẽ có lợi hơn so với chỉ mục trên các cột thường xuyên được cập nhật. Việc cập nhật các chỉ mục sẽ tăng chi phí ghi (write performance), do đó, các cột ít thay đổi sẽ tối ưu hơn.

### **3. Loại dữ liệu và độ phân tán (Data Type and Distribution)**

* **Phân phối dữ liệu**: Nếu dữ liệu trong một cột có độ phân tán cao (nhiều giá trị duy nhất), thì chỉ mục sẽ có hiệu quả hơn. Ngược lại, nếu một cột có ít giá trị duy nhất (như giới tính với các giá trị "Nam" và "Nữ"), việc tạo chỉ mục có thể không cần thiết.
* **Kiểu dữ liệu**: Một số kiểu dữ liệu có thể được tối ưu hóa tốt hơn với các loại chỉ mục nhất định. Ví dụ, chỉ mục b-tree thường được sử dụng cho dữ liệu có thứ tự, trong khi chỉ mục hash có thể được sử dụng cho các truy vấn tìm kiếm chính xác.

### **4. Khả năng mở rộng và hiệu suất tổng thể (Scalability and Overall Performance)**

* **Tăng trưởng của bảng**: Nếu bảng có khả năng tăng trưởng lớn, bạn cần phải cân nhắc số lượng chỉ mục có thể ảnh hưởng đến hiệu suất. Việc lập chỉ mục cần được theo dõi và điều chỉnh theo thời gian khi kích thước bảng thay đổi.
* **Tác động đến hiệu suất ghi**: Mặc dù chỉ mục có thể tăng tốc độ truy vấn, nhưng chúng cũng có thể làm giảm hiệu suất ghi (write performance). Bạn cần đánh giá kỹ lưỡng xem việc lập chỉ mục có mang lại lợi ích lớn hơn cho hiệu suất truy vấn so với tác động tiêu cực đến việc ghi.

### **5. Các chỉ mục hiện có (Existing Indexes)**

* **Kiểm tra các chỉ mục hiện tại**: Trước khi tạo chỉ mục mới, hãy xem xét các chỉ mục hiện có để tránh tạo ra những chỉ mục trùng lặp hoặc không cần thiết. Việc có quá nhiều chỉ mục có thể làm giảm hiệu suất tổng thể.

# 4.Làm thế nào để biết 1 câu query đã sử dụng index hay chưa ?

### **1. SQL Server**

* **Sử dụng Execution Plan**: Bạn có thể xem kế hoạch thực thi của truy vấn để xác định xem chỉ mục có được sử dụng hay không.
  + Trong SQL Server Management Studio (SSMS), bạn có thể làm như sau:
    1. Mở cửa sổ truy vấn.
    2. Nhấp vào biểu tượng "Include Actual Execution Plan" hoặc sử dụng phím tắt Ctrl + M.
    3. Chạy truy vấn của bạn.
    4. Kiểm tra tab "Execution Plan" để xem cách SQL Server thực thi truy vấn. Nếu có chỉ mục được sử dụng, bạn sẽ thấy một biểu tượng chỉ mục trong kế hoạch.

### **2. MySQL**

* **Sử dụng EXPLAIN**: Bạn có thể sử dụng từ khóa EXPLAIN trước truy vấn để xem kế hoạch thực thi.

### **3. PostgreSQL**

* **Sử dụng EXPLAIN**: Tương tự như MySQL, bạn có thể sử dụng từ khóa EXPLAIN
* Kết quả sẽ hiển thị chi tiết về cách PostgreSQL thực hiện truy vấn. Nếu có chỉ mục được sử dụng, bạn sẽ thấy thông tin liên quan đến chỉ mục trong kết quả.

### **4. Oracle**

* Sử dụng EXPLAIN PLAN
* Câu lệnh này sẽ hiển thị kế hoạch thực thi, cho bạn biết chỉ mục nào đã được sử dụng.

### **5. Kiểm tra với SHOW INDEX**

* Bạn cũng có thể sử dụng lệnh SHOW INDEX (trong MySQL) để kiểm tra các chỉ mục hiện có trong bảng
* Điều này giúp bạn xác nhận rằng chỉ mục bạn đang mong đợi có tồn tại trên bảng.