1. **Index là gì?**

* Index là một cấu trúc dữ liệu dùng để đánh chỉ mục cho các row của bảng giống như index của một cuốn sách, nó giúp cho storage engine tìm kiếm các row một cách nhanh chóng.
* Index trong MySQL thường được lưu trữ dưới dạng cấu trúc B-Tree, hoặc cũng có thể dưới dạng R-Tree, hoặc với storage engine MEMORY thì lưu dưới cấu trúc Hash, hoặc Full Text Index.
* Index được sử dụng trong việc SELECT dữ liệu sẽ làm cho hiệu suất nhanh hơn, nhưng đối với các câu lệnh UPDATE, DELETE, INSERT thì sẽ mất nhiều thời gian hơn do phải cập nhật cả Index.
* Nên tạo index cho các trường hợp sau:
  + Các collumn mà thường xuyên được sử dụng trong mệnh đề WHERE.
  + Các column thường xuyên sử dụng để join như là primary key hay foreign key.
  + Các column sử dụng trong mệnh đề Group By hay Order By.
* Index không cần thiết phải tạo cho trường hợp sau:
  + Các Table mà có số lượng rows nhỏ.
  + Các tables mà thường xuyên phải sử dụng các thao tác đọc ghi dữ liệu lên đĩa.
  + Các column không sử dụng trong mệnh đề WHERE.
* Cú pháp tạo index như sau:

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index\_name [index\_type] ON tbl\_name (index\_col\_name,...) [index\_type].

1. **Các loại index.**

Có 2 loại cấu trúc index cơ bản là clustered-index và nonclustered-index.

* **Clustered Index:** 
  + Dữ liệu lưu trên đĩa đã được sắp xếp theo trường index.
  + Đây là cấu trúc Index dùng cho Storage Engine InnoDB.
  + Clusterd index lưu trữ data dưới dạng B-Tree, mỗi nút lá bao gồm cả index và data của record có index tương ứng.
  + Khi ta định nghĩa một pimary key trong table, thì innoDB sẽ sử dụng nó như một clustered index
* **NonClustered Index:** 
  + Là kiểu cấu trúc mà việc sắp sếp các index không mapping với việc sắp xếp dữ liệu trên đĩa.
  + Mỗi node lá của cấu trúc nonclustered index không chứa dữ liệu mà chỉ chứa index và con trỏ trỏ tới địa chỉ thực lưu data trên đĩa(con trỏ này chính là record identifier).
  + MyISAM sử dụng cấu trúc nonclustered index.
  + Có thể tạo một hoặc nhiều nonclusterd index cho một table.

1. **Các dạng Index:**

Dạng Index hay có thể hiểu là cấu trúc lưu trữ dữ liệu của index. MySQL có các dạng index sau: loại B-Tree, R-Tree, Hash, và Full Text.

* **Dạng B-Tree**:
  + Sử dụng cấu trúc B-Tree là dữ liệu đã được sắp xếp theo một thứ tự nhất định dưới dạng cây. Các node lá của cây bao gồm giá trị index và data record(clustered index) hoặc là con trỏ tới nơi lưu data record(nonclustered index).
  + Cấu trúc B-Tree giúp tăng tốc độ truy cập dữ liệu vì storage engine không phải truy cập qua tất cả các node của cây.
* **Dạng R-Tree**:
  + Hỗ trợ đối với storage engine MyISAM.
  + Thường để lập chỉ mục cho các kiểu dữ liệu không gian, dữ liệu về tọa độ địa lý hoặc dữ liệu 3 chiều.
  + Việc sắp xếp các chỉ số theo cấu trúc R-Tree cũng gần giống với B-Tree nhưng phép so sánh cho việc sắp xếp là khác nhau(ở B-Tree chỉ có 2 nhánh là > hoặc < giá trị key).
* **Dạng HASH**:
  + Hash index được xây dựng dựa trên cấu trúc hash table. Với mỗi một row, storage engine sẽ tính ra một hash code cho các column là index. Việc tạo ra hash code sẽ làm giảm lượng bộ nhớ phải lưu trữ giá trị index. Giá sử trường cần đánh index là một trường mà cần nhiều byte lưu trữ thì khi sẽ tốn nhiều bộ nhớ hơn việc tạo ra các hash code.
  + Chỉ hỗ trợ bởi storage engine MEMORY.
  + Index data sẽ lưu lại hash code của key mà đánh index và con trỏ trở tới row tương ứng.
* **Dạng FULL TEXT**:
  + Chỉ hỗ trợ bởi MyISAM.
  + Sử dụng trong các câu query mà tìm kiếm kiểu Full Text.
  + Có thể tạo Full Text Index cho các column có kiểu TEXT, CHAR , hoặc VARCHAR.
  + FULL TEXT Index thường được sử dụng cho các hàm MATCH(), AGAINTS() mà không sử dụng cho mệnh đề WHERE.

1. **Các dạng Scan khi xử lý query:**

Có 2 dạng scan chính là table scan và index scan:

* **Table Scan**: Khi thực thi query dạng table scan thì tức là duyệt toàn bộ các rows trong table một cách tuần tự. Cùng với quá trình duyệt là việc kiểm tra các điều kiện lấy data. Nên dạng table scan thường tốn nhiều chi phí cho việc đọc dữ liệu. Nếu tables mà có kích thước lớn thì dạng scan này sẽ tốn thời gian. Table scan thường được sử dụng trong trường hợp data của table là nhỏ, hoặc sử dụng trong data warehourse nơi mà mặc dù dữ liệu các bảng là lớn, nhưng tất cả đã được phân tích và tổng hợp. Dạng table scan sẽ được thực thi trong các trường hợp sau:
* Không có mệnh đề Where trong câu query.
* Các trường trong mệnh đề Where đều không phải là trường index.
* Thường sử dụng khi mà số lượng các bản ghi trả về là lớn so với tổng số bản ghi của toàn bảng.
* **Index Scan**: Không duyệt table theo từng row mà là duyệt bằng index data. Việc sử dụng index scan trong nhiều trường hợp mang lại hiệu xuất cao. Index Scan thường được sử dụng trong :
  + điều kiện tìm kiếm của mệnh đề WHERE.
  + mệnh đề GROUP BY.
  + mệnh đề ORDER BY.
  + Thường được sử dụng khi lượng row trả về là nhỏ so với số lượng row của toàn bảng.

VD: về việc lựa chọn kế hoạch index scan hay table scan.

Trong DB tạo 2 bảng là hoc\_sinh(ID, Name, Age, Address) và nhan\_vien(ID, Name, Age, Address) có dữ liệu giống hệt nhau. Tạo index trên trường name cho bảng hoc\_sinh, bảng nhan\_vien thì không tạo index. Thực hiện câu truy vấn như sau đối với từng bảng:

* hoc\_sinh: SELECT \* FROM hoc\_sinh h where h.name = 'lampq';
* nhan\_vien: SELECT \* FROM nhan\_vien h where h.name = 'lampq';

Chạy 2 câu lệnh trên cho kết quả câu lệnh thứ nhất chạy mất thời gian0.0010s, câu lệnh thứ 2 cho kết quả chạy là 0.0011s

* + - kết quả cho thấy là việc tạo index cho hiệu suất nhanh hơn so với bảng không có index.