Một khung nhìn (view) là một câu lệnh SQL được lưu trữ trong cơ sở dữ liệu với một **tên xác định**.

Một view thực sự là **một thành phần của một bảng** dưới dạng truy vấn SQL được xác định trước.

Một view có thể chứa **tất cả các hàng của một bảng** hoặc chọn **các hàng từ một bảng**. Một view  có thể được tạo từ một hoặc nhiều bảng phụ thuộc vào truy vấn SQL để tạo ra một view.

**View là một loại bảng ảo** cho phép người dùng thực hiện các thao tác sau:

Cấu trúc dữ liệu theo cách mà người dùng tìm thấy tự nhiên hoặc trực quan.

Hạn chế quyền truy cập vào dữ liệu theo cách mà người dùng có thể nhìn thấy và (đôi khi) sửa đổi chính xác những gì họ cần.

Tóm tắt dữ liệu từ các bảng khác nhau có thể được sử dụng để tạo báo cáo.

WITH CHECK OPTION là tùy chọn với câu lệnh CREATE VIEW.

Mục đích của WITH CHECK OPTION là để **đảm bảo** rằng tất cả các **UPDATE và INSERT đều đáp ứng** (các) điều kiện trong định nghĩa view.

Nếu chúng không thỏa mãn (các) điều kiện, thì UPDATE hoặc INSERT trả về lỗi.

Khối mã sau đây có một ví dụ về việc tạo cùng một view CUSTOMERS\_VIEW với WITH CHECK OPTION:

Các **chỉ mục cơ sở dữ liệu** trong MySQL cho phép bạn **tăng tốc hiệu suất** của các câu lệnh truy vấn. Đối với **các bảng nhỏ**, một chỉ mục không giúp được nhiều. Tuy nhiên, nếu bạn có các bảng có lượng dữ liệu lớn, các chỉ mục có thể cải thiện đáng kể hiệu năng.

Các hành vi sau đây là tất cả các dấu hiệu cho thấy các bảng có thể được hưởng lợi từ việc thêm các chỉ mục:

Trang web của bạn tải rất chậm hoặc hoàn toàn không.

Một ứng dụng không thể kết nối với cơ sở dữ liệu của nó.

Các truy vấn cơ sở dữ liệu được sao lưu.

            Nếu bạn gặp bất kỳ vấn đề nào trong số này, bạn nên phân tích các truy vấn cơ sở dữ liệu của mình và xem xét thêm các chỉ mục.

            Để xác định bảng nào có thể có lợi từ việc sử dụng chỉ mục, bạn phải **phân tích các truy vấn cơ sở dữ liệu của mình**. Câu lệnh **EXPLAIN SELECT** giúp bạn làm điều này:

Khái niệm về chỉ mục (**index**) trong mysql

Chỉ mục (Index) là **bảng tra cứu đặc biệt** mà Database Search Engine có thể **sử dụng để tăng nhanh thời gian và hiệu suất** thu thập dữ liệu. Hiểu đơn giản, **một chỉ mục** là **một con trỏ** tới **dữ liệu trong một bảng**. Một chỉ mục trong một Database là tương tự như một chỉ mục trong Mục lục của cuốn sách.

Ví dụ, nếu bạn muốn tham chiếu tất cả các trang trong một cuốn sách về một chủ đề nào đó, đầu tiên bạn nghĩ ngay đến mục lục của nó, mà liệt kê tất cả các chương, chủ đề theo thứ tự và sau đó được tham chiếu tới một hoặc nhiều trang cụ thể. Khi đã có mục lục của cuốn sách, việc tìm kiếm trang sách đó sẽ nhanh chóng và đỡ tốn công hơn rất nhiều so với việc bạn phải lật từng trang.

Khi nào cần đánh chỉ mục cho bảng

Bất cứ khi nào bạn **thay đổi cấu trúc bảng** **dữ liệu** bạn đều cần Add Index lại, giống như khi bạn thay đổi nội dung quyển sách, bạn cần phải làm lại mục lục. Vậy hãy cân nhắc, nếu Database của bạn sử dụng **INSERT hay UPDATE nhiều hơn là SELECT** thì Index chỉ **làm chậm** thêm mà thôi.

Lưu ý: Index giống như các tab. Chúng làm cho nó nhanh hơn để bạn có thể tìm thấy những dữ liệu bạn cần phù hợp với một số tiêu chí nhất định. Để tạo chỉ mục tốt bạn cần phải suy nghĩ về cách bạn sẽ được truy vấn dữ liệu như thế nào. Khi bạn có thông tin này, bạn sẽ có thể tạo index và nó sẽ tăng tốc độ truy vấn của bạn ngay lập tức.

Ở các post trước, các tác giả đã nói cụ thể về cú pháp sử dụng index nên trong post này, tôi xin nói thêm về các loại index trong mysql

MySQL cung cấp **3 kiểu index** khác nhau cho data đó là **B-Tree, Hash** và **R-Tree** index. Hiểu rõ về các kiểu index này sẽ giúp chúng ta tạo ra các loại index trong bảng một cách hiệu quả hơn. Do R-Tree được sử dụng cho các loại dữ liệu không gian spacial data và thường ít khi gặp phải nên bài viết này chúng ta sẽ tập chung so sánh 2 loại index là B-Tree và Hash:

**B-Tree index** có các đặc điểm sau:

Dữ liệu index trong **B-Tree được tổ chức và lưu trữ theo dạng cây(tree),** tức là có **root, branch, leaf**. Giá trị của các node được tổ chức tăng dần từ trái qua phải.

Khi truy vấn dữ liệu thì việc tìm kiếm trong B-Tree là 1 quá trình đệ quy, bắt đầu từ root node và tìm kiếm tới branch và leaf, đến khi tìm được tất cả dữ liệu - thỏa mãn với điều kiện truy vấn thì mới dùng lại.

B-Tree index được sử dụng cho những column trong bảng khi muốn tìm kiếm 1 giá trị nằm trong khoảng nào đó. Ví dụ: tìm kiếm những sinh viên có điểm Toán từ 5-9.

**Hash Index**

Hash index có các đặc điểm sau:

Hash index có một vài đặc điểm khác biệt so với Btree index.

Dữ liệu index được tổ chức theo dạng **Key - Value** được liên kết với nhau.

Khác với B-Tree, thì **Hash index** chỉ nên **sử dụng** trong các **biểu thức toán tử** là **= và <>.** Không sử dụng cho toán từ tìm kiếm 1 khoảng giá trị như **> hay < .**

Không thể tối ưu hóa toán tử ORDER BY bằng việc sử dụng Hash index bởi vì nó không thể tìm kiếm được phần từ tiếp theo trong Order.

Toàn bộ nội dung của Key được sử dụng để tìm kiếm giá trị records, khác với B-Tree một phần của node cũng có thể được sử dụng để tìm kiếm.

Hash có tốc độ nhanh hơn kiểu Btree.

Các chỉ mục **không** nên được sử dụng trong **các bảng nhỏ.**

**Không** nên khi mà Bảng **thường xuyên** có các hoạt động **update, insert.**

Các chỉ mục **không** nên được sử dụng trên các **cột** mà chứa một số lượng lớn **giá trị NULL.**

**Không** nên dùng chỉ mục trên các **cột** mà thường xuyên **bị sửa đổi**.