Tối Ưu Hóa Câu Lệnh Bằng Covering Index

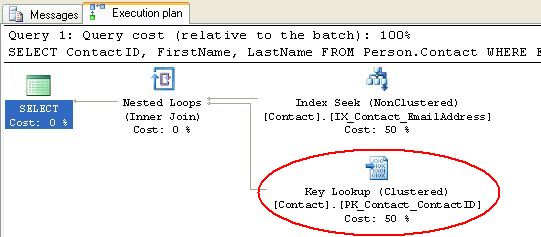
[Vũ Huy Tâm](http://www.sqlviet.com/blog)

Khi một non-clustered index được dùng để thực thi một câu lệnh, ta thường thấy trong kế hoạch thực thi thao tác *Key Lookup* (hoặc Bookmark Lookup), là thao tác mà hệ thống sau khi tìm kiếm trên cây index nhảy tới bản ghi tương ứng trong bảng để lấy các trường dữ liệu cần trả về:

USE AdventureWorks

GO

SELECT ContactID, FirstName, LastName

FROM Person.Contact

WHERE EmailAddress = 'kristina1@adventure-works.com'

Ta thấy index IX\_Contact\_EmailAddress trên trường EmailAddress đã được sử dụng (thao tác Index Seek), và [câu lệnh đạt được hiệu năng tốt hơn rất nhiều so với quét bảng](http://www.sqlviet.com/blog/index-giup-tang-hieu-nang-thuc-hien-nhu-the-nao) khi không có index. Tuy nhiên thao tác Key Lookup ở đó vẫn chiếm tới một nửa chi phí câu lệnh. Một kỹ thuật có thể giúp tối ưu hơn nữa cho câu lệnh này, là loại bỏ Key Lookup bằng cách đưa các trường dữ liệu cần thiết vào cây index. Khi đó hệ thống chỉ cần tìm trên index và trả kết quả về cho câu lệnh mà không cần phải truy nhập vào bảng. Index lúc đó được gọi là *covering index* cho câu lệnh.

Ở phiên bản SQL Server 2000 trở về trước, cách làm duy nhất để đạt được điều này là tạo index trên tất cả các cột cần tìm, như ở ví dụ trên là tạo một index phức hợp gồm ba trường EmailAddress, FirstName và LastName. Tuy nhiên cách làm này không phải luôn khả thi, ví dụ khi index trên một mình trường EmailAddress là cần thiết để duy trì ràng buộc duy nhất trên trường này; đưa thêm các trường khác vào index làm phá vỡ ràng buộc này. Hoặc khi trường cần đưa vào có kiểu VARCHAR(MAX) – kiểu dữ liệu này không thể tạo được index.

Phiên bản SQL Server 2005 bắt đầu bổ sung thêm lựa chọn INCLUDE trong lệnh CREATE INDEX để tăng khả năng *covering* của index:

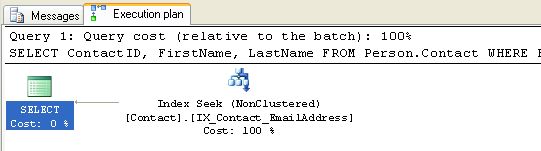
CREATE NONCLUSTERED INDEX IX\_Contact\_EmailAddress

ON Person.Contact(EmailAddress)

INCLUDE (FirstName,LastName)

WITH (DROP\_EXISTING  = ON) -- xóa index nếu đã được tạo trước đó

Lệnh trên vẫn tạo index trên trường EmailAddress, nhưng đồng thời “ký gửi” hai trường FirstName và LastName vào đó. Cần lưu ý là hai trường này không thuộc về khóa index. Ví dụ nếu đây là unique index thì trường EmailAddress vẫn phải duy nhất. Khi thực hiện lại câu lệnh SELECT ở trên, kế hoạch thực thi đã thay đổi:



Như vậy Key Lookup đã biến mất, tức là bước truy nhập vào bảng đã bị loại bỏ, vì tất cả các trường dữ liệu mà câu lệnh yêu cầu đã được tìm thấy ngay tại index. Điều này cũng có nghĩa là chi phí câu lệnh được giảm đi một nửa. Thực tế đây là mức tối ưu nhất mà một câu lệnh có thể đạt được khi dùng non-clustered index. Một ưu điểm nữa khi chỉ cần đọc index là nó làm giảm tranh chấp trên bảng với các câu lệnh khác đang đồng thời truy nhập vào bảng đó.

Cũng từ kế hoạch thực thi trên ta có thêm một nhận xét là trường ContactID, vốn là khóa chính trong bảng, cũng được đọc từ cây index mà không cần quay sang bảng. Sở dĩ như vậy là vì, trường khóa chính luôn được lưu trên cây index để dùng làm con trỏ đến bản ghi tương ứng trong bảng (khi bảng không có khóa chính thì một giá trị định danh bản ghi, RowID, được dùng).

Lưu ý là kỹ thuật trên cũng làm tăng chi phí cho việc UPDATE và INSERT dữ liệu. Ví dụ khi bạn cập nhật FirstName và LastName, không những bản ghi trong bảng mà cả node index tương ứng cũng cần được cập nhật theo. Bạn cần thử nghiệm kỹ càng xem hiệu năng của cả hệ thống có bị ảnh hưởng không trước khi áp dụng kỹ thuật này.

**Phiên bản áp dụng**: SQL Server 2005 trở lên