Mục lục

[I. Khái niệm và thuật ngữ chỉ mục: 2](#_Toc309759931)

[1. Các loại chỉ mục cơ bản: 2](#_Toc309759932)

[2. Các phương pháp truy vấn đối với index 3](#_Toc309759933)

[II. Qui tắc chỉ mục theo kinh nghiệm: 3](#_Toc309759934)

[III. Quyết định lựa chọn chỉ mục: 4](#_Toc309759935)

[IV. Chọn lựa chỉ mục kết hợp: 7](#_Toc309759936)

[1. Phép kết vòng lặp lồng (nested-loop join): 7](#_Toc309759937)

[2. Phép kết vòng lặp lồng block (block nested-loop join) 9](#_Toc309759938)

[3. Phép kết vòng lặp lồng chỉ mục (indexed nested-loop join) 9](#_Toc309759939)

[4. Phép kết kết hợp với sắp xếp: 10](#_Toc309759940)

[5. Phép kết hash: 11](#_Toc309759941)

[THỦ THUẬT VÀ CÁC QUAN NIỆM DÀNH CHO CHUYÊN GIA CƠ SỞ DỮ LIỆU 12](#_Toc309759942)

**SELECTING INDEXES**

# Khái niệm và thuật ngữ chỉ mục:

Chỉ mục là một trong các công cụ cơ bản được sử dụng trong hệ quản trị cơ sở dữ liệu để phân chia hiệu suất cao cho nhiều câu truy vấn thường sử dụng của hệ thống. Chỉ mục có mục đích đa dạng trong phạm vi hiệu suất: tìm kiếm nhanh thông tin cụ thể và phạm vi dữ liệu, sự ép buộc tính duy nhất và v.v…

Các mức độ tinh chỉnh cơ sở dữ liệu:

* **mức đầu tiên** (thấp nhất) trong tinh chỉnh cơ sở dữ liệu là mức phần cứng. Hiện tượng thắt cổ chai có thể được giảm hoặc loại bỏ bằng cách tăng tài nguyên, chẳng hạn làm CPU chạy nhanh hơn, tăng dung lượng bộ nhớ (không gian bộ nhớ đệm), thêm đĩa, sử dụng hệ thống RAID hoặc stripping (phân vùng) dữ liệu dọc các đĩa nhằm tăng khả năng xử lí song song
* **mức thứ hai** là trong chính hệ thống cơ sở dữ liệu. Hiệu suất hệ thống đôi khi có thể được cải thiện bằng cách tăng kích thước buffer hoặc thay đổi khoảng cách giữa các checkpoint.
* **mức thứ ba** (cao nhất) là thực hiện ở mức lược đồ và các giao tác. Giao tác có thể được thiết kế lại để đạt tốc độ cũng như trong tinh chỉnh truy vấn được thực hiện bởi hệ thống. Lập trình viên phải viết cẩn thận các câu truy vấn SQL vì cơ chế tinh chỉnh của DBMS không hoàn hảo. Các bảng có thể không đạt dạng chuẩn để đạt hiệu suất cao nhưng bất lợi trong xóa và cập nhật các bất thường. Cuối cùng quyết định về các khung nhìn, phân vùng, và lựa chọn chỉ mục được thực hiện ở bước này.

## Các loại chỉ mục cơ bản:

* **Chỉ mục cây B+** : chỉ mục bảng đơn giản dựa trên một hoặc nhiều thuộc tính, thường được dùng để ràng buộc tính duy nhất của khóa ngoại
* **Chỉ mục bảng Hash**: Một chỉ mục giúp ánh xạ giá trị khóa chính tương ứng địa chỉ block dữ liệu trong cơ sở dữ liệu cho các dữ liệu chưa được sắp xếp.
* **Chỉ mục kết hợp**: Một chỉ mục dựa trên nhiều hơn một thuộc tính hoặc khóa
* **Chỉ mục phân nhóm (chỉ mục cluster**): Bất kì cấu trúc chỉ mục nào cũng có dữ liệu được truy vấn theo cách index được tổ chức vì thế dữ liệu với các giá trị khóa tương tự hoặc giống nhau sẽ được lưu trữ chung. Phân nhóm dường như giảm thiểu đáng kể thời gian I/O đối với truy vấn và lưu trữ.
* **Chỉ mục bao phủ (chỉ mục duy nhất):** một chỉ mục với đầy đủ thông tin thỏa các truy vấn cụ thể bằng chính nó, hay nói cách khác là truy vấn thỏa yêu cầu khi chỉ cần tìm kiếm trong chỉ mục mà không cần tìm trong cơ sở dữ liệu
* **Chỉ mục bitmap:** một tập hợp các mảng bit để hình thành một chỉ mục có thể được sử dụng để tìm kiếm nhanh chỉ mục thứ cấp hoặc tìm kiếm trong kho dữ liệu. Mỗi mảng bit đại diện cho một giá trị khác nhau của thuộc tính, và độ dài của mỗi mảng là số dòng trong bảng.
* **Chỉ mục dày và thưa:** một chỉ mục dày chứa con trỏ đến mỗi dòng trong bảng, một chỉ mục thưa chứa hầu hết một con trỏ đến một block hoặc page dữ liệu trong bảng

## Các phương pháp truy vấn đối với index

Mỗi hệ cơ sở dữ liệu có các phương pháp truy vấn có sẵn được sử dụng để quét các index và dữ liệu, và mỗi hệ thống được cài đặt để tối ưu tìm kiếm dữ liệu cho mỗi loại truy vấn cụ thể. Các phương pháp cơ bản thường sử dụng là:

* **Quét bảng** (table scanning): khi chỉ mục không cần thiết
* **Quét chỉ mục**(index scanning): đối với chỉ mục phân nhóm và không phân nhóm (clustered and nonclusted index)
* **Quét chỉ mục duy nhất** (index only): quét một chỉ mục bao phủ
* **Chỉ mục block hoặc dòng** (block or row index ANDing): đối với entry chỉ mục kết hợp lại cho các truy vấn đa điểm
* **Nạp trước danh sách:** sự sắp xếp trên các record ID được kèm theo việc nạp trước các dòng đã sắp xếp từ block hoặc trang dữ liệu, cho phép các phép toán tuần tự trong tương lai trên dữ liệu này được nạp lên trước

# Qui tắc chỉ mục theo kinh nghiệm:

1. **Đặt chỉ mục trên mỗi khóa chính và hầu hết các khóa ngoại trên cơ sở dữ liệu:** hầu hết các phép kết xảy ra trên khóa chính và khóa ngoại, do đó hãy đặt chỉ mục trên khóa chính và khóa ngoại bất cứ khi nào có thể.

Chúng ta lưu ý rằng ở hầu hết các hệ quản trị cơ sở dữ liệu, khi khóa chính được khai báo trong SQL, chỉ mục cũng được tạo ra. Khóa ngoại nên được xem xét trong phạm vi sử dụng như một trường tìm kiếm, và trong quá trình xóa khóa chính. Cụ thể là, khi có một số lượng nhỏ giá trị có thể (như giới tính, địa chỉ state), chỉ mục có thể khá lớn và bị ảnh hưởng khi xóa khóa chính.

1. **Thuộc tính thường được dùng trong mệnh đề WHERE là ứng viên tiềm năng cho chỉ mục.** Mệnh đề WHERE được dùng đối với thuộc tính tìm kiếm trong câu truy vấn và là ứng viên cho chỉ mục khi vị từ là một phép bằng. Những thuộc tính vừa được dùng để hạn chế số dòng mục tiêu hoặc phép kết của 2 bảng. Trong cả hai trường hợp, chỉ mục được xem xét để tăng tốc tính toán

Một ngoại lệ đối với qui tác này khi vị từ là phép không bằng như “WHERE cost <> 4”. Chỉ mục ít khi được dùng trong phép không bằng vì có nhiều sự chọn lựa.

1. **Sử dụng chỉ mục cây B+ cho cả truy vấn bằng và truy vấn phạm vi.**
2. **Chọn cẩn thận một cluster index cho mỗi bảng.** Một cluster index cho biết cách dữ liệu được lưu trữ vật lý như thế nào đối với một bảng cụ thể. Việc cài đặt một cluster index trên khó chính rất quan trọng, cho phép tìm kiếm phạm vi và duy trì một thứ tự giúp xử lý nhóm hiệu quả.
3. **Tránh hoặc xóa các chỉ mục dư thừa.** Tránh các chỉ mục được cài đặt trên cùng cột hoặc hầu hết các cột đều giống nhay, nó sẽ làm truy vấn phức tạp hơn.
4. **Chỉ thêm chỉ mục khi thật sự cần thiết.**
5. **Thêm hoặc xóa cột chỉ mục đối với chỉ mục kết hợp để cải thiện hiệu suất. Đừng thay đổi cột khóa chính.**
6. **Sử dụng thuộc tính đối với chỉ mục cần chú ý khi nào chúng thường được cập nhât.**
7. **Duy trì chỉ mục trên nền tảng thông thường, chỉ xóa chỉ mục khi chúng thật sự ảnh hưởng hiệu suất**. Duy trì chỉ mục có thể giảm hiệu suất của các dòng lệnh đáng kể: INSERT, UPDATE, DELETE, IMPORT, LOAD.
8. **Tránh các cực trị trong bản số và phân phối giá trị.**
9. **Chỉ mục bao phủ (covering index – index only) rất có ích nhưng không được lạm dụng**. Chỉ mục bao phủ có thể được dùng để tìm kiếm dữ liệu, tráng I/O với database
10. **Sử dụng chỉ mục bitmap đối với lượng dữ liệu lớn, đặc biệt là trong data warehouse**

# Quyết định lựa chọn chỉ mục:

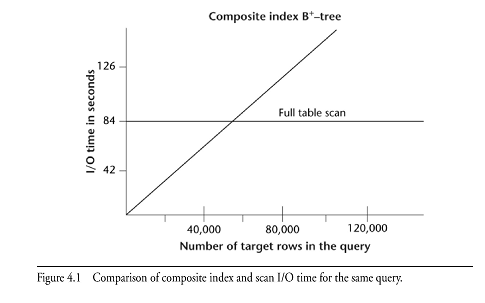
Chúng ta cần xem xét các chỉ mục mở rộng gia tăng và tìm hiểu liệu có cách nào tốt hơn hay không, ví dụ hiệu suất có được cải thiện đáng kể với mỗi index mới hay không. Không có bất kì tiêu chuẩn cụ thể nào ở đây hết, nhưng chúng ta phải cài đặt độc lập cho mỗi hệ thống cơ sở dữ liệu và người dùng.

Chúng ta cũng cần xem xét các thao tác cập nhật. Đôi khi một chỉ mục mới cải thiện hiệu suất truy vấn nhưng lại làm giảm hiệu năng cập nhật. Sự kết hợp giữa hiệu suất truy vấn và cập nhật sẽ quyết định một chỉ mục có ích hay không. Điều này là sự cân bằng thường gặp trong thiết kế chỉ mục, và đòi hỏi chúng ta ghi lại các cập nhật nào là cần thiết, tần số và bảng nào mà chúng tác động đến.

* **Quyết định thiết kế 1: Bảng này cần thiết một chỉ mục hay không, và nếu cần thì nên cài đặt chỉ mục trên từ khóa tìm kiếm nào?** Xuất phát điểm hợp lý cho thiết kế chỉ mục là khai báo kháo chính và chỉ mục trên khóa ngoại, bất chấp hỗn hợp truy vấn bạn đang thực hiện. Đối với các ứng dụng đòi hỏi trong phạm vi tần số, lượng truy cập dữ liệu nhiều, và/hoặc độ ưu tiên, thêm hoặc sửa chỉ mục cần phải được xem xét nếu hiện tượng thắt cổ chai xuất hiện.

Nếu quyết định cần chỉ mục, hãy xem xét mệnh đề WHERE ( và AND) của truy vấn để quyết định thuộc tính nào có thể được dùng cho khóa tìm kiếm chỉ mục. Mỗi truy vấn có thể liên quan đến nhiều thuộc tính khác nhau và chỉ mục nên được chọn cho mỗi truy vấn mà có thể cải thiện đáng kể hiệu suất cơ sở dữ liệu. Các truy vấn đòi hỏi quét toàn bảng có thể không cần bất kì chỉ mục nào. Chỉ mục tăng tốc cho hơn một câu truy vấn sẽ có giá trị nhất.

* **Quyết định thiết kế 2: Khi nào cần từ khóa tìm kiếm đa thuộc tính (kết hợp) và nên chọn cái nào?** Một truy vấn đa điểm liên quan mệnh đề WHERE với nhiều thuộc tính. Khi loại truy vấn này thực hiện, chỉ mục cây B+ kết hợp nên được sử dụng để truy vấn hiệu quả tập hợp các dòng thỏa điều kiện truy vấn. Phân tích sự cân bằng giữa quét toàn bảng và chỉ mục B+ đa thuộc tính được thể hiện trong hình bên dưới



* **Quyết định thiết kế 3: Nên sử dụng chỉ mục dày hay thưa?** Khi số dòng nhỏ tương ứng với kích thước page, chỉ mục thưa tốt hơn. Một chỉ mục dày sẽ có nhiều entry hơn và sẽ thường yêu cần thêm mức chỉ mục nữa để có thể truy xuất dữ liệu hơn chỉ mục thưa. Mặt khác, chỉ mục dày có thể kết hợp một cách hiệu quả để tạo ra chỉ mục kết hợp cho truy vấn đa điểm. Khi số dòng lớn tương ứng kích thước trang lớn, chỉ mục thưa ít hiệu quả hơn. Một ưu thế khác của chỉ mục thưa là nó có thể trở thành chỉ mục bao phủ cho một số truy vấn nhất định
* **Quyết định thiết kế 4: Khi nào có thể dùng chỉ mục bao phủ?** Đôi khi một chỉ mục kết hợp có thể được sử dụng như chỉ mục bao phủ đối với một số truy vấn nhất định có thể đạt yêu cầu khi chỉ tìm kiếm trên index. Ví dụ, mọt chỉ mục trên attribute, make, và model có thể thỏa câu truy vấn sau:

**SELECT make, model, vin, year**

**FROM carOnLot**

**WHERE feature = ‘catalytic converter’;**

* **Quyết định thiết kế 5: Nên sử dụng chỉ mục phân nhóm (clustered index) hay không?** Các qui tắc cơ bản của cluster index:
* Chỉ có một cluster index được cài trên một bảng, hay nói cách khác sẽ có nhiều xung đột giữa các dòng thỏa nhiều truy vấn sử dụng các chỉ mục khác nhau
* Nếu một chỉ mục có thể được dùng như một chỉ mục bao phủ, thì không cần phân nhóm nữa, vì câu truy vấn có thể đạt yêu cầu chỉ cần truy vấn chỉ mục
* Truy vấn phạm vi, truy vấn đa điểm và truy vấn đơn điểm trên các giá trị không phải khóa chính có thể được lợi từ phân nhóm. Mỗi loại truy vấn liên quan đến truy xuất nhiều dòng, và bất kì sự phân nhóm nào cũng sẽ cải thiện hiệu suất. Nếu ta có vài lựa chọn, phân tích sự cân bằng có thể cần để quyết định chọn lựa nào sẽ tối ưu hiệu suất.
* **Quyết định thiết kế 6: Một chỉ mục có còn thích hợp khi cập nhật được thực hiện? Đâu là sự cân bằng giữa truy vấn và cập nhật đối với mỗi lựa chọn chỉ mục?** Xem xét sự cân bằng từ viễn cảnh chi phí / lợi ích, xem xét thời gian I/O với tất cả giao tác liên quan đến truy vấn và các cập nhật của mỗi bảng mục tiêu đối với chỉ mục đã cho, tính đến tần số mỗi truy vấn và cập nhật trong một khoảng thơi gian cố định:

***Lợi ích của chỉ mục = thời gian I/O (tất cả truy vấn không có index) – thời gian I/O (tất cả truy vấn có index)***

***Chi phí của chỉ mục = thời gian I/O (tất cả cập nhật có index) – thời gian I/O (tất cả cập nhật không có index)***

Chú ý mỗi cập nhật có 2 bước: truy vấn để truy xuất đến dòng được cập nhật và thao tác cập nhật dòng đó. Clustered index dường như có chi phí cập nhật cao hơn so với nonclustered index, và nên được xem xét cẩn thận

**Qui tắc cơ bản**: tạo một chỉ mục nếu lợi nhiều hơn hại (lợi ích nhiều hơn chi phí). Qui tắc này có một ngoại lệ phải xem xét, bởi vì nó chỉ dựa trên thời gian I/O. Độ ưu tiên truy vấn có thể cao hơn độ ưu tiên cập nhật, đặc biệt nếu cập nhật được xử lý đợt trong giờ không cao điểm trong ngày, nghĩa là bạn có thể sống với việc hoãn cập nhật lại vài giờ. Trong trường hợp này, lợi ích nhất định thấp hơn chi phí, bởi vì được đo bằng thời gian I/O, vì vậy mà độ ưu tiên là một yếu tố quan trọng hơn.

Mặt khác, nếu cập nhật phải thực hiện ngay lập tức, truy vấn bị hoãn lại thì lợi ích đương nhiên lơn hơn chi phí khi xác định sử dụng chỉ mục

* **Quyết định thiết kế 7: Làm sao biết đã chọn lựa đúng chỉ mục?** Chúng ta có thể thực hiện một vài phân tích cân bằng hiệu suất như đo thời gian I/O cần thiết để đáp ứng một tập các truy vấn trên bảng hoặc một tập các bảng và chỉ mục. Để so sánh được thì dữ liệu cần được lấy trước và sau cài đặt chỉ mục. Nếu không có hiệu quả nào, hoặc có tình trạng giảm hiệu năng, hãy xem xét các phương án tốt hơn

# Chọn lựa chỉ mục kết hợp:

Phân tích của chúng ta được dựa trên giả sử nếu có một phép kết giữa một bảng mà khóa ngoại của nó khớp với khóa chính của bảng khác, m đại diện cho số dòng trong bảng có khóa chính, n đại diện cho số dòng trong bảng có khóa ngoại. Nếu phép kết không giữa 2 bảng như trên, thì thiết kế của bảng nào có m dòng và bảng nào có n dòng là tuỳ ý.

## Phép kết vòng lặp lồng (nested-loop join):

Chiến lược lồng vòng lặp là một phương pháp kết cơ bản. Vòng lặp ngoài là một phép quét tuần tự bảng đầu tiên R, và với mỗi dòng trong bảng R, vòng lặp trong được thực hiện, một phép quét tuần tự bảng thứ hai S. Tham số cơ bản là số dòng, m và n, trong hai bảng của phép kết và thứ tự vật lý các dòng trong mỗi bảng. Độ phức tạp là O(mn) vì lặp 2 lần. Mã giả của phép kết:

**For each tuple r in R do**

**For each tuple s in S do**

**If r and s satisfy the join condition**

**Then output the tuple <r,s>**

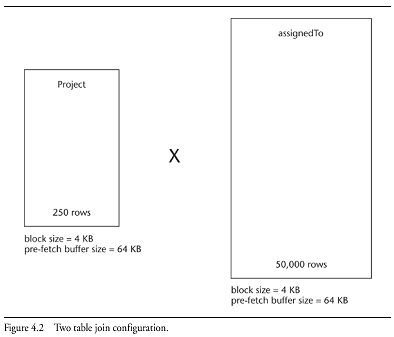
Chúng ta giả sử mỗi bảng được lưu trong không gian đĩa vật lý kề nhau để có lợi cho quét toàn bảng. Chi phí thời gian thực hiện chiến lược này còn tùy thuộc vào bảng nào được chọn cho vòng lặp ngoài và vòng lặp trong

***Ví dụ: giả sử có bảng Employee với 10,000 dòng, bảng Project với 250 dòng, và bảng AssignedTo với 50,000 dòng. Ta có câu truy vấn sau (không tính thời gian hiển thị kết quả của phép kết)***

***SELECT p.projectName, p.projectLeader, a.empId***

***FROM project AS p, assignedTo AS a***

***WHERE p.projectName = a.projectName;***

****

***Biết kích thước block: 4KB***

***Kích thước bộ đệm nạp trước: 64KB***

***Thời gian truy xuất block (giả sử xét ổ cứng IBM U320): 3,6ms (seek time), thời gian xoay trung bình là 2ms, và tốc độ chuyển tải là 320 MB/sec.***

*Số buffer cần nạp trước (64KB) cho bảng assignedTo*

*= ceiling (( 50,000 dòng x 100bytes/dòng) /64KB)*

*= ceiling (5,000,000/65,536) = 77*

*Số buffer cần nạp trước cho bảng project*

*= ceiling((250 dòng x 200 bytes/dòng) / 64KB)*

*= ceiling (50,000 /65,536)*

*= 1*

Trong đó ceiling là hàm lấy mức tối đa của số thực

**Vòng lặp lồng trường hợp 1:** assignedTo là bảng vòng lặp ngoài

*Thời gian I/O phép kết = quét assignedTo 1 lần, quét bảng project n lần*

*= ( 77 buffers + 50,000 lần quét x 1 buffer) x 5.8ms*

*= 290.4 sec*

**Vòng lặp lồng trường hợp 2:** project là bảng lặp ngoài

*Thời gian I/O phép kết = quét project 1 lần, quét bảng assignedTo m lần*

*= (1 buffer + 250 lần quét x 77 buffer) x 5.8ms*

*= 19,251 buffer x 5.8ms*

*= 111.7 sec*

Một cách tổng quát bảng nhỏ hơn thường được chọn là bảng ngoài sẽ có lợi hơn trong thực thi truy vấn. Chú ý chiến lược này không tận dụng bất kì sự sắp xếp dòng hoặc phương pháp chỉ mục nào. Chúng ta cũng nên tránh các phép kết dựa trên các thuộc tính không khóa

## Phép kết vòng lặp lồng block (block nested-loop join)

Phép kết lồng vòng lặp block chỉ là sự đa dạng của phép kết lồng vòng lặp trong đó bảng vòng lặp trong được quét mỗi lần một block của bảng vòng lặp ngoài thay vì mỗi lần một dòng. Khi một block của bảng vòng lặp được lưu trong buffer, thuật toán có thể quét tất cả dòng trong block mà không có thao tác I/O. Xử lý này có thể giảm đáng kể số I/O cần thiết cho phép kết và thích hợp hơn khi tiếp cận vòng lặp lồng cơ bản. Hãy xem xét lại ví dụ bên trên

**Trường hợp 1:** assignedTo là bảng vòng lặp ngoài

*Thời gian I/O phép kết = quét assignedTo 1 lần, quét project 77 lần nạp trước vào buffer*

*= ( 77 buffer + 77 lần quét x 1 buffer) x 5.8 ms*

*= 0.89 sec*

**Trường hợp 2:** project là bảng vòng lặp ngoài

*Thời gian I/O phép kết = quét project 1 lần, quét assigned m lần*

*= ( 1 buffer + 1 lần quét x 77 buffer) x 5.8 ms*

*= 45 sec*

Cấu hình vòng lặp lồng block nhanh hơn so với cấu hình vòng lặp lồng cơ bản. Chúng ta lưu ý rằng tạo bảng project là bảng vòng lặp ngoài là chiến lược tốt hơn cho cả 2 cấu hình

## Phép kết vòng lặp lồng chỉ mục (indexed nested-loop join)

Phương pháp này là một thay thế có lợi cho phép kết vòng lặp lồng block khi sự chọn lựa trong phép kết thấp vì thế chỉ có một tập hợp nhỏ các dòng của bảng kết cần được truy xuất.

**Ví dụ 4.2;**

**SELECT p.projectName, p.projectLeader, a.empId**

**FROM project AS p, assignedTo AS a**

**WHERE p.projectName = a.projectName**

**AND p.projectName = ‘financial analysis’;**

**Trường hợp 1:** quét bảng khóa ngoại 1 lần và chỉ mục đến bảng khóa chính một lần

Chiến lược cơ bản là quét toàn bộ bảng chứa khóa ngoại, và với mỗi giá trị thuộc tính kết thỏa yêu cầu, xác định dòng duy nhất tương ứng trong bảng khóa chính thông qua chỉ mục duy nhất. Sẽ có thao tác quét bảng khóa ngoại và (h +1) lần truy xuất block của khóa chính thông qua chỉ mục duy nhất đối với mỗi dòng. Giả sử số dòng mục tiêu ntr = 100 – dòng thỏa bảng khóa ngoại khớp với một dòng trong bảng khóa chings. Giả sử h = 2

*Thời gian I/O phép kết= quét toàn bảng khóa ngoại (assignedTo) + chỉ mục đến những dòng thỏa của bảng khóa chính (project)*

*= (77 buffer + (h+1) truy xuất block (buffer)) x 5.8ms*

*= ( 77 + 3) x 5.8 ms*

*= 0.46 sec*

**Trường hợp 2**: chỉ mục đến bảng khóa chính, và sau đó chỉ mục đến bảng khóa ngoại

Một phương pháp khác là sử dụng cây B+ hoặc chỉ mục hash đến bảng khóa chính để tìm các giá trị khóa chính thỏa điều kiện, sau đó sử dụng chỉ mục B+ kết hợp đến bảng khóa ngoại tìm các dòng chứa giá trị khóa ngoại trùng với giá trị khóa chính. Đối với trường hợp này, giả định h = 3, một block đòi hỏi con trỏ ntr = 100 dòng khóa ngoại mục tiêu được cho như trong trường hợp 1

*Thời gian I/O phép kết = chỉ mục đến bảng khóa chính + chỉ mục đến bảng khóa ngoại*

*= ( (h +1) + [h + 1 + ntr]) x 5.8ms*

*= (4 + 104 ) 5.8 ms*

*= 0.63 sec*

## Phép kết kết hợp với sắp xếp:

Chiến lược phép kết kết hợp sắp xếp lợi dụng các dòng theo thứ tự tương tự cách xử lý theo đợt thực hiện. Nếu bảng được sắp xếp trên các cột kết, thì chỉ cần một phép quét tuần tự từng bảng để hoàn thành phép kết. Nếu một hoặc cả hai bảng chưa được sắp xếp trên cột kết, thì mỗi bảng chưa sắp xếp sẽ được sắp xếp trước khi phép trộn thực thi. Thậm chí khi tính cả chi phí sắp xếp, thuật toán này vẫn nhanh hơn vòng lặp lồng. Nếu truy vấn không đòi hỏi sự sắp xếp(tức trong cây truy vấn có order by) thì phép kết kết hợp sắp xếp sẽ hiệu quả hơn phép kết vòng lặp lồng block và ngược lại

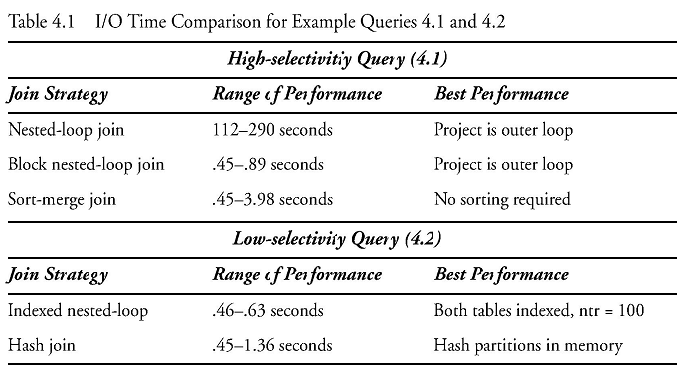
## Phép kết hash:

Chiến lược kết hash cũng hiệu quả cho phép kết ít sự chọn lọc. Chiến lược cơ bản là xây dựng một bảng hash trong bộ nhớ, nếu có thể, hoặc trên đĩa nếu không đủ bộ nhớ, bằng cách quét mỗi bảng kết và băm giá trị thuộc tính các cột thỏa yêu cầu để phân vùng. Thuật toán thay thế còn được sử dụng để băm input nhỏ hơn trong hai input của phép kết vào bộ nhớ và sau đó đọc input lớn hơn khớp với bảng hash trong bộ nhớ. Chúng ta giả sử thuật toán đầu tiên ở đây, nơi mà hai phân vùng được tìm kiếm để xác định các thuộc tính bẳng (lưu ý các thuộc tính khác nhau có thể băm(hash) vào cùng nơi, vì vậy cần thiết phải tìm kiếm để kiểm tra chúng, nhưng các giá trị thuộc tính tương tự sẽ không băm vào các vị trí khác nhau)

Khi một bảng băm được xây dựng, pha thứ hai của thuật toán là truy vấn đến các dòng thật sự thỏa điều kiện từ mỗi bảng (chẳng hạn những dòng từ hai bảng với các giá trị thuộc tính giống nhau ở các phân vùng file hash). Độ phức tạp là O (3m + 2n) vì có 3 pha thực hiện quét: một pha đọc 2 bảng, một pha ghi lại 2 bảng vào các phân vùng bằng giá trị băm, và một để đọc các phân vùng cho phép kết thật sự.

Trong thực thi phép kết hash, thao tác quét bảng cho các phân vùng có thể thực hiện không thường xuyên miễn là phân vùng hash file trong bộ nhớ vẫn không bị thay đổi đối với chuỗi các truy vấn

**Tóm lại, đối với truy vấn có nhiều sự lựa chọn, cả phép kết vòng lặp lồng block và phép kết trộn với sắp xếp đều thực hiện tốt. Đối với phép kết có ít sự lựa chọn, phép kết vòng lặp lồng index và phép kết hash làm việc tốt, với phép kết hash có ít giá trị hơn cho cận trên đối với các truy vấn cho trước.**



# THỦ THUẬT VÀ CÁC QUAN NIỆM DÀNH CHO CHUYÊN GIA CƠ SỞ DỮ LIỆU

* **Thủ thuật 1: Số lượng và kích thước chỉ mục thay đổi.** Tuy nhiên theo tổng quát, chỉ mục thường chiếm khoảng 10% - 20% không giân đĩa cơ sở dữ liệu. Hơn 25% đĩa được sử dụng cho chỉ mục nên bị phản đối và nên tiến hành xem xét lần hai một cách nghiêm túc.
* **Thủ thuật 2: Chỉ mục giúp truy vấn nhưng làm hại thao tác ghi. Vì vậy cơ sở dữ liệu truy vấn** nhiều có thể phạm sai lầm trên chỉ mục, và cơ sở dữ liệu có cả nhiều hoạt động insert/delete/update, hoặc đối với cơ sở dữ liệu nào mà hiệu suất của các hoạt động ghi được chú trọng, nên bảo toàn số lượng chỉ mục chúng định nghĩa cho hệ thống.
* **Thủ thuật 3: Ôn lại toàn bộ qui tắc chỉ mục dựa trên kinh nghiệm đã liệt kê ở trên.**