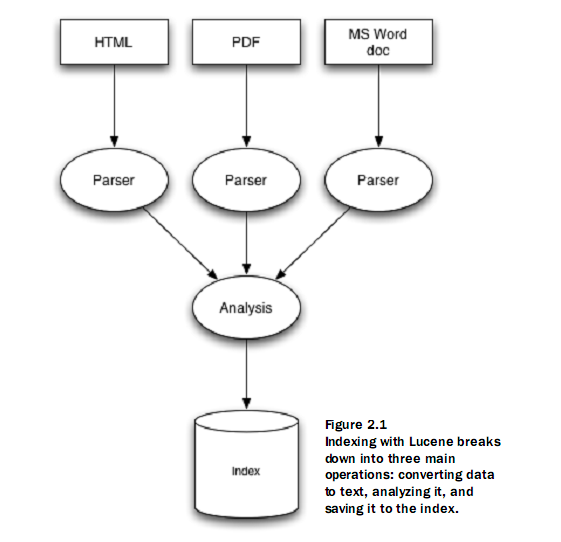
**LUCENE IN ACTION**

# CHAPTER 2: INDEXING

## Quy trình đánh chỉ mục



### Convert to text

Để tiến hành index được trong Lucene,thì trước hết ta phải chuyển đổi dữ liệu thành dạng văn bản thuần túy(plain text,như file .txt chẳng hạn).Điều này là quan trọng,bởi vì dữ liệu được lưu trữ dưới nhiều dạng file khác nhau (pdf,word,excel,powerpoint,html…),trong khi đó Lucene chỉ hỗ trợ bạn index các Field dạng String , Date hoặc đối tượng Reader mà thôi.Vấn đề chuyển đổi thành dạng plain text sẽ được nói rõ hơn trong phần Chương 7

### Analysis

Mỗi khi bạn chuẩn bị cho việc index và tạo ra đối tượng Document với các Field,thì Lucene sẽ phân tích dữ liệu này sao cho phù hợp nhất với việc index.Để làm điều này,Lucene sẽ phân chia dữ liệu thành các chuỗi hoặc là các kí tự thông qua việc lựa chọn các toán tử thực thi trên chúng.Chẳng hạn như việc bạn phân tích thành các kí tự thường,hoặc bỏ đi các từ ngữ không có nghĩa…Chi tiết về việc phân tích cùng các lớp API sẽ được đề cập đến trong Chương 4

### Wrting index

Sau khi dữ liệu được phân tích ,nó sẽ sẵn sang cho việc index.Lucene sẽ chứa dữ liệu này theo cấu trúc inverted index(chỉ mục có thể nghịch đảo ).Cấu trúc này sẽ có hiệu quả để tiết kiệm dung lượng ổ đĩa và cho phép tìm kiếm nhanh hơn các từ khóa trong quá trình search.Nguyên tắc của nó là thay vì phải tìm kiếm các từ nào chứa trong tài liệu đó thì với cấu trúc này sẽ tối ưu hóa việc tìm ra câu trả lời “tài liệu nào chứa từ khóa này”.

# Các toán tử đánh chỉ mục cơ bản

Lucene hỗ trợ các toán tử giúp thực hiện việc đánh chỉ mục như:

* Thêm tài liệu mới (Document) cùng các trường (Fields): Keyworks,UnIndexed,UnStored và Text.Trong mỗi tài liệu lại có thể có chứa nhiều Fields cùng tồn tại,và trong mỗi Fields lại có thêm nhiều giá trị khác nhau
* Xóa tài liệu ra khỏi chỉ mục : (Remove Documents )Sử dụng lớp IndexReader với phương thức delete () ta có thể dễ dàng xóa bỏ tài liệu được chỉ định ra khỏi chỉ mục.Lucene sẽ xem như các tài liệu này được đánh dấu như là sẽ xóa.Tuy nhiên việc này chỉ có thể thực hiện khi gọi hàm close().
* Hủy bỏ việc xóa tài liệu (Undeleting Documents): Như đã đề cập,việc thực sự xóa bỏ tài liệu khi gọi phương thức close() của lớp IndexReader.Do đó trươc khi thực sự xóa tài liệu này thì chỉ cần gọi phương thức undeleteAll() của cùng lớp IndexReader trước đó sẽ xóa bỏ đi các tài liệu được đánh dấu xóa bỏ.
* Cập nhật tài liệu : Lucene không hỗ trợ ta thưc hiện việc cập nhật tài liệu ,thay vào đó ta sẽ xóa bỏ tài liệu và sau đó thêm lại tài liệu mới thay thế.Chú ý rằng để đảm bảo tốc độ thực thi thì tốt nhất việc xóa bỏ và thêm tài liệu mới nên thực hiện theo khối,không nên xen lẫn giữa việc xóa và thêm tài liệu mới.

# Khuếch đại các tài liệu và các trường(Boosting Documents and Fields)

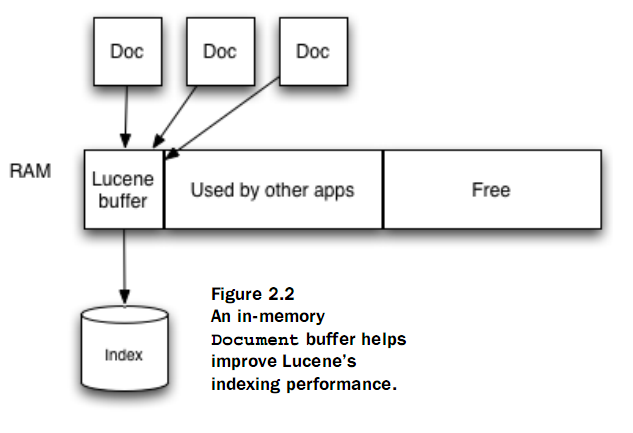
Trong quá trình index không phải tất cả các Tài liệu và trường (Documents and Fields) cũng có độ ưu tiên như nhau,goi là điểm ưu tiên (Score).Nếu như ta có ý định sắp xếp kết quả trả về trong quá trình tìm kiếm theo nhưng tiêu chí nào đó chẳng hạn như là sắp xếp theo điểm ưu tiên thì chúng ta cần phải boost tài liệu hoặc trường cần sắp xếp.

Mặc định tất cả các tại liệu đều không có boost hoặc là boost là 1.0.Do đó muốn thay đổi độ ưu tiên này ta chỉ cần gọi hàm setBoost(float) cho Document hoặc Field nào đó mà ta mong muốn.Độ ưu tiên (độ boost) càng cao thì tài liệu càng quan trọng và ngược lại .

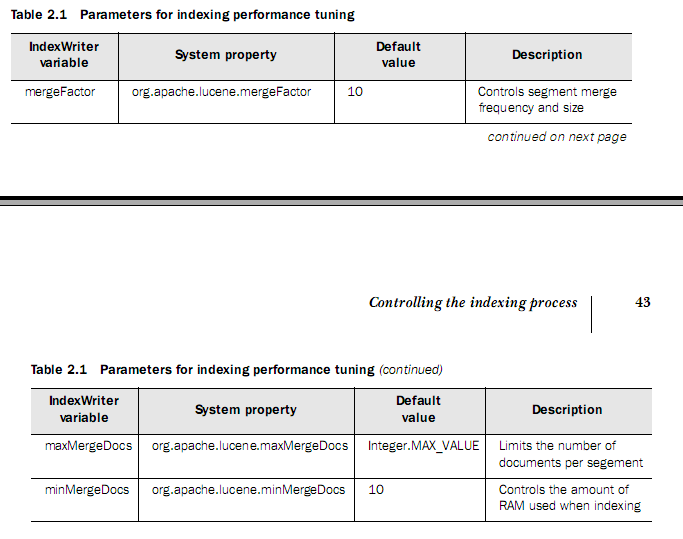
# Điều khiển tiến trình đánh chỉ mục

Với các ứng dụng nhỏ thì việc index dữ liệu là không lớn,ta hoàn toàn có thể mặc định index theo cài đặt của Lucene.Tuy nhiên với các ứng dụng lơn,khi mà dung lượng tại liệu lên đến hàng triệu,thì tốc độ index là điều hết sức quan trọng,thay vì phải mất hàng giờ thì có thể chỉ mất hàng phút mà thôi.

* Điều chỉnh việc thực thi đánh chỉ mục : Trong các ứng dụng đánh chỉ mục điển hình thì trờ ngại chính là việc ghi chỉ mục xuống đĩa.Vì vậy ta phải điều chỉnh làm sao cho Lucene tự “thông minh” trong việc đánh chỉ mục với các tài liệu mới cũng như bổ sung các file có sẵn.



Như ta đã thấy,mỗi khi thêm tài liệu mới vào chỉ mục,chúng sẽ được khởi tạo vào trong bộ nhớ đệm thay vì ghi trực tiếp xuống đĩa.Sử dụng bộ nhớ đệm này có nhiều lí do và quan trong hàng đầu vẫn là cải thiện tốc độ index.Lucene cho phép bạn điều chỉnh các tham số này một cách dễ dàng.



* Các tham số :
  + mergeFactor: tham số cho phép ta điều chỉnh số lượng Document sẽ chứa trong bộ nhớ trước khi ghi vào đĩa cũng như là bao lâu thì sẽ trộn những segment này với nhau ,giá trị mặc đinh là 10.Điều này có nghĩa là số segment trên đĩa sẽ là mũ của 10.

Ví dụ giá trị này là 10 thì có nghĩa là sẽ tạo ra 10 segment,mỗi segment có thể chứa 10 Documents,và khi segment thứ 10 được tao ra thì chúng sẽ được merge thành 1 segment có kích thước là 100. Và cứ như thế.

* + maxMergeDocs :Đây là tham số giới hạn số lượng Document tối đa chứa trong segment trước khi merger các segments với nhau.
  + minMergeDocs : tương tự như maxMergeDocs,song nó chỉ ra có tối thiểu bao nhiêu tài liệu chứa trong segment trước khi merge các segment lại với nhau

Như vậy,ta thấy rằng: nếu tăng giá trị mergeFactor thì Lucene sẽ sử dụng nhiều bộ nhớ chính hơn nhưng mật độ ghi xuống đĩa lại ít thường xuyên hơn,Điều này sẽ tăng tốc độ tiến trình index .Tuy nhiên nó sẽ tạo ra nhiều file index hơn và lại làm ảnh hưởng tới việc tìm kiếm về sau,khi phải truy xuất nhiều file trong thư mục index.maxMergeDocs càng cao thì thích hợp cho các dữ liệu khối hơn.

Tóm lại mergeFactor và maxMergeDocs càng có giá trị lớn thì tốc độ index sẽ cao,song đừng quá lạm dụng vì có thể ảnh hưởng tới việc tìm kiếm.

# Bộ nhớ trong trong quá trình đánh chỉ mục : RAMDirectory

Như đã đề cập,Lucene lưu Document mới vừa thêm vào trong bộ đệm buffer trước khi ghi xuống đĩa.Mặc định Lucene sẽ sử dụng lớp FSDirectory- một lớp cài đặt của Directory.tuy nhiên nếu ta muốn kiểm soát tốt hơn việc sử dụng bộ nhớ và mật độ ghi xuống đĩa thì RAMDirectory là một thay thế.

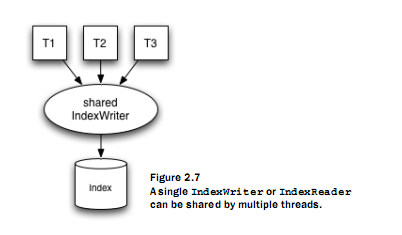
FSDirectory làm việc trực tiếp trên đĩa cứng,trong khi đó RAMDirectory thực thi trong bộ nhớ chính vì thế mà tốc độ sẽ nhanh hơn.Từ đây ta có thể nghĩ ngay tới việc sử dụng kết hợp giữa hai lớp trên để cải thiện tốc độ index.sử dụng RAMDirectory như bộ đệm lưu giữ tài liệu mới thêm vào tới một số lượng nhất định,sau đó đẩy xuống cho lớp FSDirectory thưc thi ghi xuống đĩa.

# Tối ưu hóa việc đánh chỉ mục

Việc tối ưu hóa tiến trình đánh chỉ mục là tiến trình trộn nhiều file chỉ mục lại với nhau để giảm thiểu thời gian đọc chỉ mục tr ong quá trình tìm kiếm.Bằng việc sử dụng API của lucene mà cụ thể là hàm optimize() của đối tượng IndexWriter ta có thể dễ dàng tối ưu điêu này.Tuy nhiên việc làm này chỉ có hiệu quả tăng tốc độ tìm kiếm trên chỉ mục đã có,mà không có tác động tới tốc độ đánh chỉ mục.

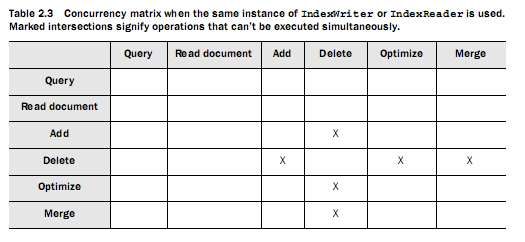
# Tính đồng thời,an toàn tiến tình,ngăn chặn các thực thi

* Các luật đồng thời : Lucene cung cấp cho ta nhiều toán tử liên quan tới việc đánh chỉ mục tài liệu như : xóa,cập nhật.Do đó trong quá trình thực hiện chúng ta phải tuân theo một số luật cụ thể để tránh việc đụng độ trong quá trình thực thi.Điều này là cấn thiết khi mà có nhiều thực thi diễn ra một cách thường xuyên trước những yêu cầu gởi từ web tới ứng dụng của bạn.Sau đây là một số luật cơ bản :
  + Bất kì toán tử chỉ đọc nào cũng có thể thực thi đồng thời,Chẳng hạn là nhiều tiến trình có thể tìm kiếm cùng một chỉ mục tại một thời điểm
  + Bất kì toán tử chỉ đọc nào cũng có thể thực thi đồng thời trong khi một chỉ mục nào đó đang được cập nhật.Ví dụ: người dùng có thể tìm kiếm trong chỉ mục trong khi nó đang được cập nhật,thêm tài liệu mới hoặc là được xóa khỏi chỉ mục
  + Chỉ có duy nhất 1 toán tử cập nhật chỉ mục có thể thực thi tại một thời điểm.Một chỉ mục chỉ có thể được mở bởi chỉ một đối tượng IndexWriter hoặc là IndexReader tại một thời điểm mà thôi.
* Tính an toàn của tiến trình(thread-safety) :Mặc dù ta thấy rằng việc cập nhật chỉ mục tại một thời điểm sử dụng nhiều đối tượng IndexWriter hoặc là IndexReader là không được phép.Song,cùng một đối tượng có thể được chia sẽ bởi nhiều tiến trình khác nhau và tất cả cùng gọi tới những phương thức cập nhật chỉ mục sẽ được đồng bộ hóa sao cho việc thực thi có tính tuần tự.



Lucene luôn đảm bảo rằng các toán tử cập nhật không được chồng lắp nhau.Có nghĩa là trước khi thêm tài liệu mới vào chỉ mục,ta phải đóng tất cả các thể hiện IndexReader trên cùng chỉ mục.Tương tự,trước khi xóa hoặc cập nhật tài liệu trong chỉ mục,ta phải đóng tất cả thể hiện IndexWriter đang mở trên cùng chỉ mục.

Sau đây là bảng mô tả các toán tử có thể thực thi đồng thời hay không:



* Khóa chỉ mục: nhằm tránh gây ra sự đụng độ trong quá trình sử dụng các hàm API của Lucene,Thư viện này đã tạo ra các file lock bên cạnh các segment để đảm bảo rằng việc thực thi một chương trình tại một thời điểm.Mỗi chỉ mục có một tập các file lock.mặc nhiên tất cả các file này được tạo ra trong thư mục tạm được xác định bởi thuộc tính java.io.tmpdir.

Có hai loại file lock được tao ra:

* + File Write.lock : được tạo ra trong quá trình đánh chỉ mục tài liệu.nó nhằm đảm bảo các tiến trình thực thi một cách có tuần tự trong quá trình cập nhật chỉ mục.Hơn nữa nó được tồn tại khi dối tượng IndexWriter đang duy trì và giữ cho tới khi nó đóng mới thôi. Ngoài ra nó tồn tại khi đối tượng IndexReader đang được sử dụng để xóa,hủy bỏ việc xóa,hay cài đặt các mục của trường nào đó.nó còn giúp khóa chỉ mục mà diễn ra lâu hơn mong đợi.
  + File commit.lock : tạo ra trong quá trình merge các segment.Nó được dùng bất kể khi nào những segment đang được đọc hoặc trộn lại với nhau.Nó được nắm giữ bởi đối tượng IndexReader trước khi nó nó đọc các segment và chỉ giải phóng sau khi IndexReader đã mở và đọc các segment
* Vô hiệu hóa tính năng khóa chỉ mục : nhiều khi ta cần vô hiệu hóa tính năng khóa chỉ mục của Lucene.chẳng hạn là ứng dụng cần đọc chỉ mục nằm trên ổ CD-ROM.Có nghĩa là ứng dụng chỉ ở chế độ đọc mà thôi,tức là chỉ tìm kiếm trên đó mà không hề có cập nhật chỉ mục.Để vô hiệu hóa ta chỉ cấn thay đổi thuộc tính disableLuceneLocks thành true là được.