V

Tổ chức Lưu trữ Dữ liệu

Nội dung chi tiết

- ***Thiết bị lưu trữ***
- ***Mẫu tin***
  - Chiều dài cố định
  - Chiều dài động
- ***Tổ chức lưu trữ mẫu tin***
  - Trong block
  - Trong file

- ***Có nhiều thiết bị lưu trữ : đĩa cứng cố định, đĩa quang, đĩa di động , băng từ… Trong đó đĩa cứng cố định là thiết bị lưu trữ chính yếu vì :***
  - Dung lượng lớn
  - Độ bền cao, lưu trữ lâu dài
  - Chi phí I/O thấp
- ***Mặt đĩa chia làm nhiều vành tròn đồng tâm gọi là track, một track chi làm nhiều đoạn đều nhau gọi là block (1 block gồm nhiều sector). Một vài block hợp thành 1 cluster (hay page)***
  - Block là đơn vị để lưu trữ và luân chuyển DL giữa đĩa và buffer để xử lý

Thiết bị lưu trữ (tt)

- ***Để chuyển block DL giữa đĩa và buffer có các bước :***

Công việc	*Thời gian thực hiện*	
	Với Client	*Với Server*
Định vị đầu đọc / ghi đúng track cần truy xuất	*7-10 msec*	*3-8 msec*
Định vị block cần truy xuất trên track ấy	*Phụ thuộc tốc độ xoay của đĩa, trung bình 2msec*	
Chuyển DL giữa đĩa và buffer	*Phụ thuộc kích thước và số lượng block chuyển*	

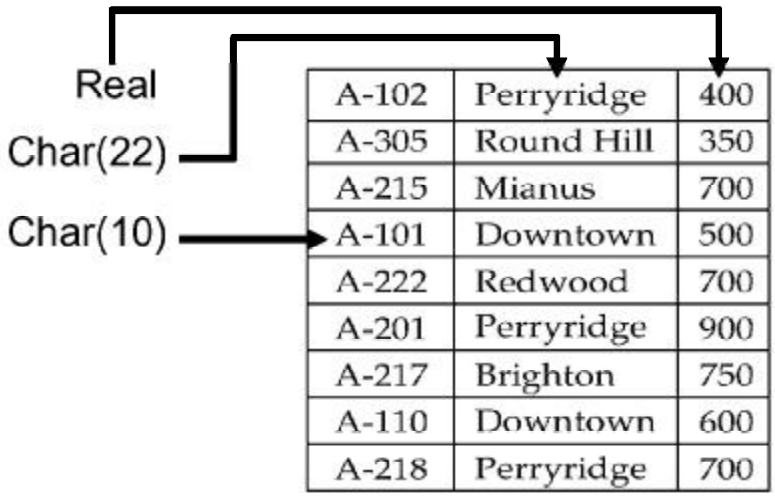
- ***Nhận xét : Truy xuất các block liên tiếp sẽ nhanh hơn các block rời rạc. Mỗi lần đọc/ghi nên truy xuất nhiều block***

- ***Thiết bị lưu trữ***
- ***Mẫu tin***
  - Chiều dài cố định
  - Chiều dài động
- ***Tổ chức lưu trữ mẫu tin***
  - Trong block
  - Trong file

Mẫu tin

- ***Tập hợp các dữ liệu có liên quan với nhau tạo thành một mẫu tin***
- ***Ví dụ***
  - Mẫu tin TaiKhoan có những thông tin
    - SoTaiKhoan
    - ChiNhanh
    - SoDu
- ***Có 2 loại mẫu tin***
    - Mẫu tin có chiều dài cố định (Fixed-Length Record)
    - Mẫu tin có chiều dài động (Variable-Length Record)

- ***Xét 1 tập tin gồm các mẫu tin TaiKhoan nêu trên***



A-102	Perryridge	400
A-305	Round Hill	350
A-215	Mianus	700
A-101	Downtown	500
A-222	Redwood	700
A-201	Perryridge	900
A-217	Brighton	750
A-110	Downtown	600
A-218	Perryridge	700

- ***Giả sử***
  - 1 char : 1 byte
  - Real : 8 bytes
- ***Vậy***
  - 1 mẫu tin account : 40 bytes
  - 40 bytes đầu tiên là mẫu tin thứ 1
  - 40 bytes tiếp theo là mẫu tin thứ 2…

Mẫu tin có chiều dài cố định (tt)

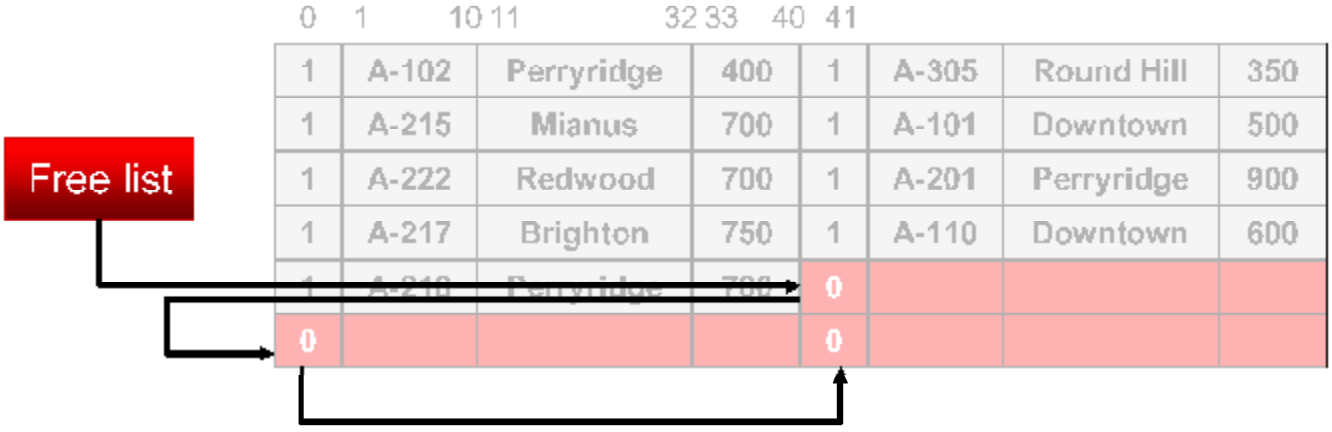
- ***Record header chứa***
  - Lược đồ của mẫu tin
  - Chiều dài của mẫu tin (không nhất thiết có)
  - Thời gian sau cùng cập nhật mẫu tin
- ***Block header chứa***
  - Con trỏ nối các block có liên quan với nhau
  - Thông tin về mối quan hệ giữa các mẫu tin trong block
  - Block ID
  - Thời gian sau cùng truy xuất block

- ***Ví dụ***
  - Mỗi mẫu tin có thêm 1 bit thông tin
    - = 0 : Mẫu tin ấy đã bị xóa
    - = 1 : Mẫu tin ấy vẫn còn tồn tại
  - Danh sách các mẫu tin trống (free list)

0	1	10	11	32	33	40	41
1	A-102	Perryridge	400	1	A-305	Round Hill	350
1	A-215	Mianus	700	1	A-101	Downtown	500
1	A-222	Redwood	700	1	A-201	Perryridge	900
1	A-217	Brighton	750	1	A-110	Downtown	600
1	A-218	Perryridge	700	0			
0				0			

Mẫu tin có chiều dài cố định (tt)

- ***Ví dụ***
  - Mỗi mẫu tin có thêm 1 bit thông tin
    - = 0 : Mẫu tin ấy đã bị xóa
    - = 1 : Mẫu tin ấy vẫn còn tồn tại
  - Danh sách các mẫu tin trống (free list)

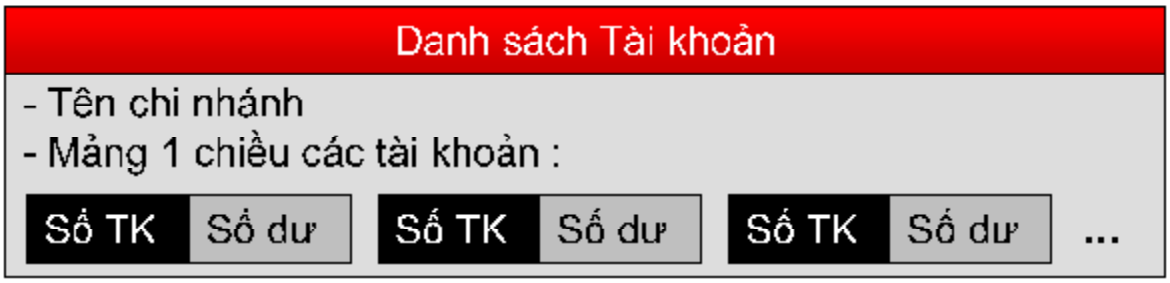


- ***Hủy mẫu tin***
  - Đánh dấu xóa vào bit thông tin
  - Đưa mẫu tin bị đánh dấu xóa vào free list
- ***Thêm một mẫu tin***
  - Hoặc thêm vào những mẫu tin bị đánh dấu xóa hoặc thêm vào cuối tập tin
  - Cập nhật lại free list
- ***Tìm kiếm***
  - Quét tuần tự trên tập tin

Nội dung chi tiết

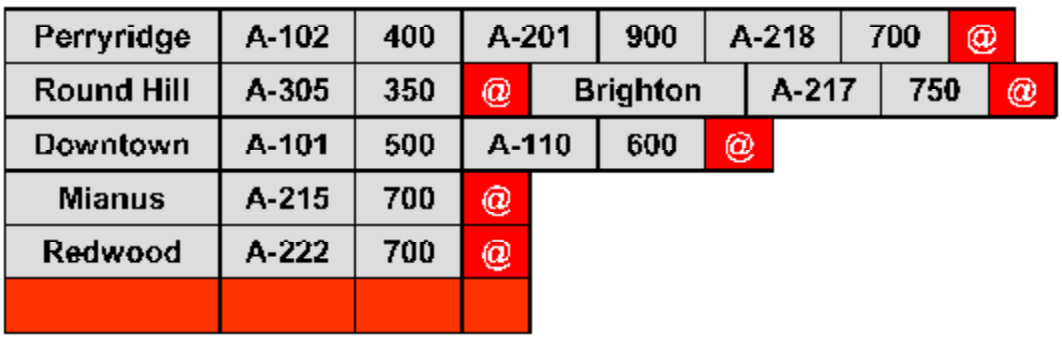
- ***Thiết bị lưu trữ***
- ***Mẫu tin***
  - Chiều dài cố định
  - Chiều dài động
- ***Tổ chức lưu trữ mẫu tin***
  - Trong block
  - Trong file

- ***Trong DBMS, mẫu tin có chiều dài động***
  - Lưu trữ nhiều loại mẫu tin trong 1 tập tin
  - Các loại mẫu tin chứa các trường có chiều dài động
- ***Có 2 cách biểu diễn***
  - Byte-String
  - Fixed-Length
- ***Xét tập tin gồm các mẫu tin Danh sách Tài khoản***



Mẫu tin có chiều dài động (tt)

- ***Cách biểu diễn Byte-String***
  - Cuối mỗi mẫu tin có 1 byte ký tự đặc biệt cho biết kết thúc mẫu tin
  - Dẫn đến tình trạng phân mảnh khi không sử dụng lại được không gian trống sau khi xóa 1 mẫu tin
  - Phải xử lý nhiều khi việc cập nhật làm cho 1 mẫu tin dài ra hay ngắn đi



Cách biểu diễn Byte-String (tt)

- Nếu số lượng thuộc tính của mẫu tin nhiều nhưng số lượng thuộc tính thực sự mang giá trị (≠ null) trong 1 mẫu tin lại ít thì
  - Lưu các cặp <Tên thuộc tính,Giá trị thuộc tính>
    - *Tên thuộc tính có thể được quy ước thành số cho ngắn gọn*
  - Dùng các ký tự đặc biệt (không bao giờ xuất hiện trong tên và giá trị thuộc) tính để đánh dấu các vị trí quan trọng
    - *Giữa tên thuộc tính và giá trị thuộc tính*
    - *Giữa các thuộc tính khác nhau*
    - *Giữa các mẫu tin khác nhau*

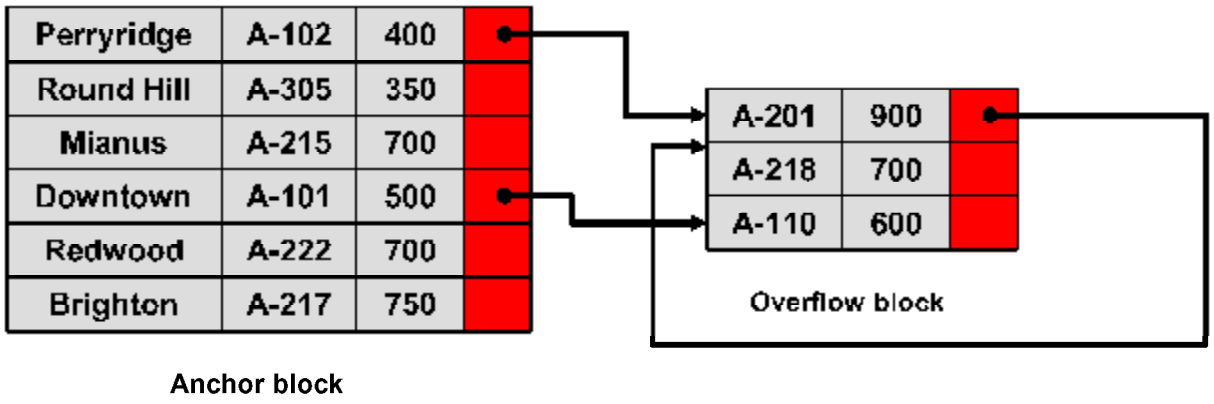
Mẫu tin có chiều dài động (tt)

Cách biểu diễn Fixed-Length

- Sử dụng 1 hay nhiều mẫu tin có chiều dài cố định biểu diễn cho những mẫu tin có chiều dài động
- Có 2 kỹ thuật
  - Kỹ thuật Reserved space : Xác định chiều dài max-length mà mẫu tin có thể đạt đến (chiều dài này phải đảm bảo không bao giờ dài thêm được nữa). Cài đặt như mẫu tin có chiều dài cố định là max-lenghth

Perryridge	A-102	400	A-201	900	A-218	700
Round Hill	A-305	350				
Mianus	A-215	700				
Downtown	A-101	500	A-110	600		
Redwood	A-222	700				
Brighton	A-217	750				

- ***Cách biểu diễn Fixed-Length***
  - Kỹ thuật Pointer : Dùng con trỏ để liên kết các mẫu tin có chiều dài cố định và tạo thành mẫu tin chiều dài động

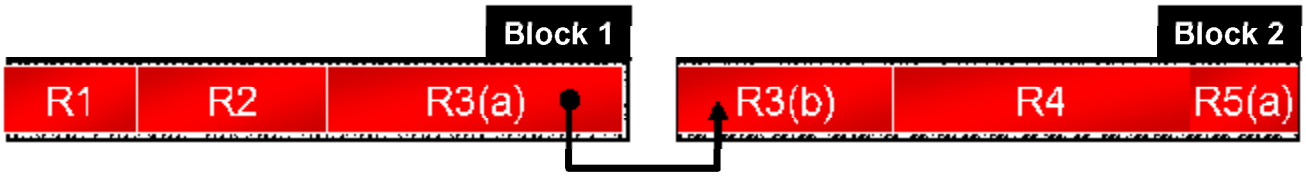


Nội dung chi tiết

- ***Thiết bị lưu trữ***
- ***Mẫu tin***
  - Chiều dài cố định
  - Chiều dài động
- ***Tổ chức lưu trữ mẫu tin***
  - Trong block
  - Trong file

- ***Mẫu tin có chiều dài không đồng nhất***
- ***Block lại có chiều dài đồng nhất và cố định***
- ***Có hai cách sắp xếp các mẫu tin vào các block***
  - Unspanned : Một mẫu tin chỉ nằm trong 1 block, chấp nhận bỏ đi những không gian dư trong block vì không đủ lưu thêm 1 mẫu tin

  - Spanned : Mẫu tin có thể bắt qua nhiều block, có các dấu hiệu nhận biết để liên kết các mảnh khác nhau của một mẫu tin



Spanned hay Unspanned

- ***Unspanned***
  - Đơn giản
  - Tốn nhiều không gian lưu trữ
- ***Spanned***
  - Cần thiết khi kích thước của mẫu tin lớn hơn kích thước của block

- ***Thiết bị lưu trữ***
- ***Mẫu tin***
  - Chiều dài cố định
  - Chiều dài động
- ***Tổ chức lưu trữ mẫu tin***
  - Trong block
  - Trong file

Tổ chức mẫu tin trên tập tin

- ***Dữ liệu trong 1 file cũng được phân ra các block (file block). Có nhiều cách phân bố các file block trên các block vật lý của đĩa***
  - Liên tục : Các block của file định vị liên tiếp trên các block của đĩa
    - Xử lý nhanh : Khi 1 block đang được xử lý thì block kế đã được chuyển sẵn vào buffer
    - Khó khăn khi file tăng trưởng
  - Không liên tục : Các file block liên kết nhau bằng con trỏ
    - Xử lý chậm
    - Dễ mở rộng file
  - Phối hợp liên tục và không liên tục : Liên tục với các block trong cùng cluster, không liên tục với các cluster

- ***File header chứa các thông tin cần thiết để mát tính có thể đọc và hiểu file***
  - Địa chỉ các block trong file
  - Mô tả định dạng các mẫu tin
    - Nếu tập tin chứa mẫu tin có chiều dài cố định : Lưu độ dài và thứ tự thuộc tính trong mẫu tin
    - Nếu tập tin chứa mẫu tin có chiều dài cố động : Lưu mã cho tên thuộc tính, các ký tự đặc biệt…
  - Sau khi 1 (hay 1 số) block được đọc vào buffer, các thông tin trong file-header sẽ được dùng để đọc mẫu tin

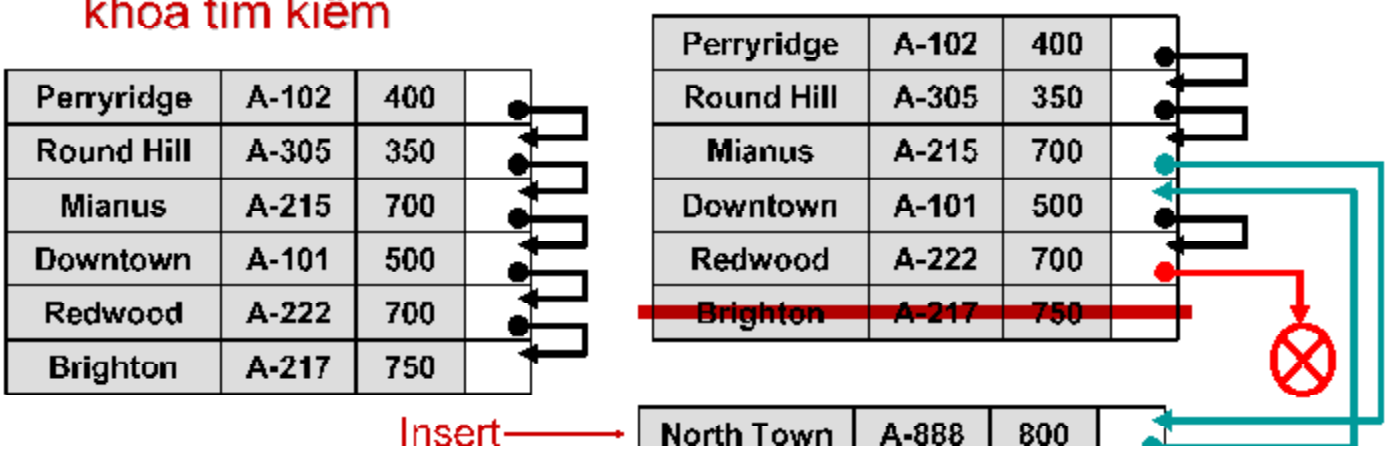
Tổ chức mẫu tin trên tập tin (tt)

- ***Cho 1 tập các mẫu tin, có các cách tổ chức các mẫu tin trên tập tin như sau :***
  - Không được sắp - Heap
  - Được sắp - Sequential
  - Gom cụm - Clustering
  - Băm - Hashing
  - Chỉ mục - Indexing
  - Cây cân bằng - BTree

- ***Mẫu tin mới được chèn vào bất cứ chỗ nào trống trong file. Do đó thứ tự dữ liệu lưu trong file không theo nội dung dữ liệu mà theo trình tự thêm mẫu tin vào file***
  - Thuận lợi
    - Thao tác file nhanh chóng
  - Hạn chế
    - Tìm kiếm phải duyệt → Không tối ưu. Đặc biệt lá với tập tin có kích thước lớn.
    - Chỉ thích hợp cho tập tin có kích thước nhỏ và một tập tin chỉ lưu dữ liệu của một quan hệ

Được sắp (Sequential)

- ***Các mẫu tin được tổ chức lưu trữ tuần tự theo 1 thứ tự nào đó, thông thường theo thuộc tính khóa tìm kiếm (search-key) thông qua một hệ thống pointer***
  - Khóa tìm kiếm không nhất thiết là khóa chính hay siêu khóa. Nên lưu trữ vật lý theo thứ tự khóa → phải đọc ít block
  - Mỗi mẫu tin có 1 con trỏ trỏ đến mẫu tin khác theo thứ tự khóa tìm kiếm



- ***Thực hiện câu truy vấn : Xét 2 quan hệ Lớp và Sinh viên, cho biết ngày mở và họ tên các Sinh viên của mỗi Lớp***
- ***Nếu các bộ của quan hệ Lớp và Sinh viên nằm gần nhau trong 1 block thì***
  - Khi 1 bộ của Lớp được đọc thì nguyên cả block chứa bộ này cũng được đưa vào bộ nhớ chính
  - Lúc đó các bộ của Sinh viên có liên quan đến Lớp đã có sẵn và được xử lý ngay một cách hiệu quả
- ***Clustering : Lưu các mẫu tin có liên quan nhau của hai hay nhiều quan hệ trong cùng block***
  - Không ý nghĩa với truy vấn đơn giản hay tần suất thấp
  - Tốt cho truy vấn kết trên các quan hệ ít thêm, xóa, sửa
  - Một block chứa các mẫu tin của nhiều quan hệ

Băm (hashing)

- ***Chia các mẫu tin trong tập tin thành từng lô (bucket) tùy theo giá trị khóa của mẫu tin***
  - Mẫu tin thuộc về lô nào là tùy thuộc vào hàm băm được áp dụng trên giá trị khóa của mẫu tin đó
  - Các mẫu tin có khóa khác nhau có thể được băm vào cùng 1 lô, vì vậy cần phải tìm tuần tự trên lô để định vị mẫu tin
- ***Khi có quá nhiều mẫu tin lưu vào 1 lô, lô sẽ tràn → sử dụng thêm các lô tràn cho 1 lô gốc***

- *Chỉ mục được dùng để truy xuất dữ liệu nhanh hơn*
- *Một tập tin dữ liệu sẽ có 1 hoặc nhiều tập tin chỉ mục kèm theo*
- *Tập tin chỉ mục gồm* search-key pointer
- *Tập tin chỉ mục sẽ nhỏ hơn rất nhiều so với tập tin dữ liệu ban đầu*
- *Tập tin chỉ mục được sắp xếp thứ tự theo khóa tìm kiếm*
- *Chỉ mục giúp tìm kiếm nhanh nhưng sẽ làm chậm việc thêm, xóa, sửa → phải thỏa hiệp*

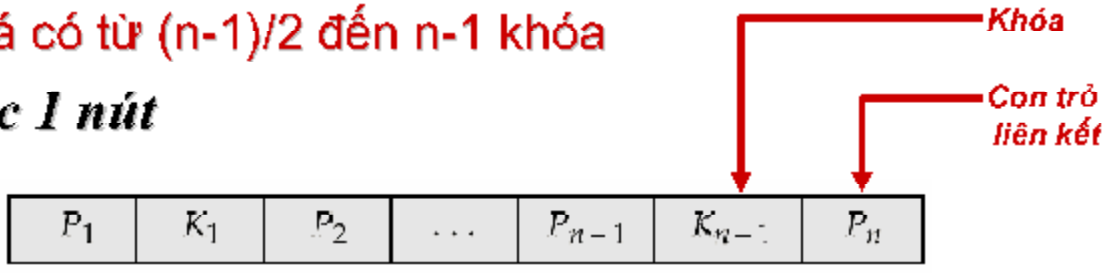
Chỉ mục (tt)

- *Nếu 1 tập tin chứa các mẫu tin đã được sắp thứ tự*
  - Chỉ mục gom cụm (Clustering index)
    - Là chỉ mục có khóa tìm kiếm định nghĩa ra thứ tự sắp xếp các mẫu tin của tập tin gốc
    - Một mẫu tin trong file Index trỏ tới block trong file dữ liệu mà lần đầu tiên giá trị khóa tìm kiếm tương ứng xuất hiện
    - Khi khóa tìm kiếm có ràng buộc duy nhất thì clustering index được gọi là primary index
  - Chỉ mục không gom cụm (Nonclustering index)
    - Là chỉ mục có khóa tìm kiếm đưa ra 1 thứ tự sắp xếp khác với thứ tự tuần tự của tập tin gốc
    - Một mẫu tin trong file Index trỏ tới block/bộ trong file dữ liệu mà lần đầu tiên giá trị khóa tìm kiếm tương ứng xuất hiện

- ***Chỉ mục dày (Dense index)***
  - Tập tin gốc có bao nhiêu giá trị khóa tìm kiếm thì tập tin chỉ mục có bấy nhiêu mẫu tin tương ứng
- ***Chỉ mục thưa (Sparse index)***
  - Tập tin chỉ mục chỉ lưu lại 1 số khóa của tập tin gốc
  - Để xác định vị trí của 1 khóa k
    - Tìm trong tập tin chỉ mục khóa lớn nhất nhưng vẫn bé hơn k
    - Tìm trong tập tin gốc bắt đầu từ địa chỉ vừa xác định trong tập tin chỉ mục
- ***Một file Dữ liệu có thể có nhiều file chỉ mục dày nhưng chỉ có duy nhất 1 file chỉ mục thưa***
- ***Lựa chọn khóa tìm kiếm (thành phần thuộc tính) dựa trên tiêu chí nào ?***

Cây cân bằng (B-Tree)

- ***B-Tree là 1 cây có gốc thỏa điều kiện***
  - Tất cả các đường đi từ nút gốc đến đến nút lá đều bằng nhau
  - Ngoại trừ nút gốc, mỗi nút có từ n/2 đến n cây con (n là bậc của cây)
  - Nút lá có từ (n-1)/2 đến n-1 khóa
- ***Cấu trúc 1 nút***



- ***Khóa tìm kiếm trên 1 nút được sắp thứ tự tăng dần***
  - K1 < K2 < K3 < . . . < Kn–1
- ***Xem lại nội dung B-tree (Cấu trúc DL 2)***