

---

## Chương 3

# Cấu trúc lưu trữ và tập tin

1

## Nội dung

---

- ☐ Phương tiện lưu trữ vật lý
  - ☐ Đĩa từ
  - ☐ Hệ thống đĩa dự phòng
  - ☐ Tổ chức File
  - ☐ Tổ chức các mẫu tin trong file
  - ☐ Lưu trữ tự điển dữ liệu
  - ☐ Chỉ mục
  - ☐ B-cây
  - ☐ Băm
- 

2

## Phương tiện lưu trữ vật lý

---

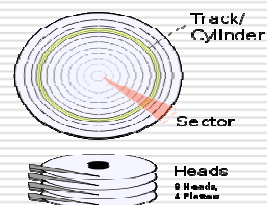
- ☐ Bộ nhớ cache
  - ☐ Bộ nhớ chính
  - ☐ Bộ nhớ flash
  - ☐ Đĩa từ
  - ☐ Đĩa quang
  - ☐ Băng từ
- 

3

## Đĩa từ

---

Gồm nhiều tấm đĩa hình tròn, hai mặt được phủ bằng vật liệu từ tính, thông tin được ghi trên bề mặt đĩa.



4

## Yếu tố đo lường hiệu năng của đĩa

- ❑ Thời gian truy xuất (Access Time): Là khoảng thời gian từ khi yêu cầu đọc-viết được phát đi đến khi bắt đầu truyền dữ liệu.
- ❑ Thời gian tìm kiếm trung bình (Average Seek Time): Là trung bình của thời gian tìm kiếm, được đo lường trên một dãy các yêu cầu ngẫu nhiên.
- ❑ Thời gian tiềm ẩn (Latency Time) : Là thời gian chờ sector được truy xuất xuất hiện dưới đầu đọc/viết.
- ❑ Tốc độ truyền dữ liệu : Là tốc độ mà theo đó dữ liệu có thể được lấy ra từ đĩa hoặc được lưu trữ vào đĩa
- ❑ Thời gian trung bình không sự cố (Mean Time to Failure) : Là lượng thời gian trung bình hệ thống chạy liên tục không có bất kỳ sự cố nào.

5

## Tối ưu hóa truy xuất khối đĩa

- ❑ *Lập lịch trình (Scheduling)*: yêu cầu các khối theo thứ tự mà nó chạy qua dưới đầu đọc-viết. Nếu các khối mong muốn ở trên các trụ khác nhau, ta yêu cầu các khối theo thứ tự sao cho làm tối thiểu sự di chuyển cánh tay đĩa.
- ❑ *Tổ chức khối đĩa kề nhau* : tổ chức các khối trên đĩa theo cách tương ứng gần nhất với cách mà dữ liệu được truy xuất.
- ❑ *Sử dụng RAM dự phòng dùng pin (Battery Backup RAM)* .
- ❑ *Đĩa log (Log Disk)* : Một cách tiếp cận khác để làm giảm thời gian viết lên đĩa là sử dụng đĩa log : Một đĩa được tận hiến cho việc viết một log tuần tự: ghi nhận những thay đổi trên dữ liệu

6

## Cải tiến độ tin cậy thông qua lưu trữ dư thừa

- ❑ Tổ chức nhiều đĩa hoạt động song song lưu trữ dữ liệu dư thừa.
- ❑ Một sự đa dạng các kỹ thuật tổ chức đĩa, được gọi là **RAID** được đề nghị nhằm vào vấn đề tăng cường hiệu năng và độ tin cậy của việc đọc ghi đĩa

7

## Cải tiến hiệu năng thông qua song song

- ❑ Phân chia dữ liệu mức bit
- ❑ Phân chia dữ liệu mức khối

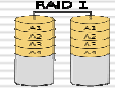
8

## Redundant Arrays of Inexpensive Disks – RAID (1)

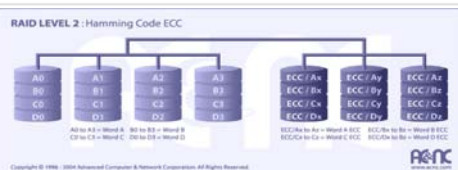
□ Raid 0:



□ Raid 1:



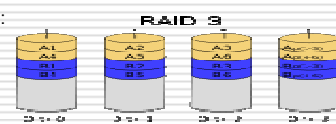
□ Raid 2:



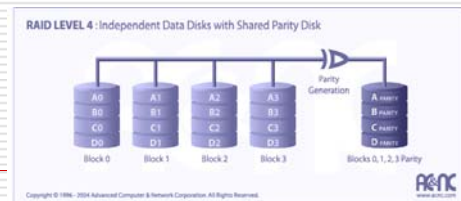
9

## Redundant Arrays of Inexpensive Disks – RAID (2)

□ Raid 3:



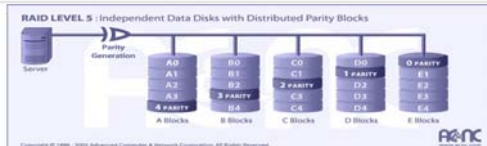
□ Raid 4:



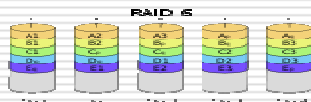
10

## Redundant Arrays of Inexpensive Disks – RAID (3)

□ Raid 5:



□ Raid 6:



11

## Tổ chức file

- Một file được tổ chức logic như một dãy các mẫu tin (record).
- Các mẫu tin này được ánh xạ lên các khối đĩa
- Các khối có kích cỡ cố định được xác định bởi tính chất vật lý của đĩa và bởi hệ điều hành, song kích cỡ của mẫu tin có thể thay đổi.
- Trong CSDL quan hệ, các bộ của các quan hệ khác nhau nói chung có kích cỡ khác nhau

12

## Mẫu tin có độ dài cố định (Fixed-Length Records)

- ❑ type  
depositor = record  
    branch\_name : char(20) ;  
    account\_number : char(12) ;  
    balance : real ;  
end
- ❑ Nêu các vấn đề nảy sinh đối với mẫu tin độ dài cố định?
- ❑ Nêu cách giải quyết vấn đề nảy sinh trên?

13

## Mẫu tin có độ dài cố định (Fixed-Length Records)

- ❑ Các vấn đề nảy sinh đối với mẫu tin độ dài cố định:
- ❑ Khi xoá một mẫu tin thì không gian bị chiếm bởi mẫu tin bị xoá phải được lấp đầy với mẫu tin khác của file hoặc ta phải đánh dấu mẫu tin bị xoá.
- ❑ Nếu kích cỡ khối là bội của 40 thì một số mẫu tin sẽ bắt chéo qua biên khối.

14

## Mẫu tin có độ dài cố định (Fixed-Length Records)

- ❑ Cách giải quyết:
- ❑ Tịnh tiến các mẫu tin ngay sau mẫu tin bị xoá về lấp vị trí mẫu tin bị xoá
  - Cách này mất nhiều thời gian
- ❑ Di chuyển mẫu tin cuối vào vị trí mẫu tin bị xoá

15

## Mẫu tin có độ dài cố định (Fixed-Length Records)

0	Perryridge	A-102	400		0	Perryridge	A-102	400
1	Round Hill	A-305	350		1	Round Hill	A-305	350
2	Mianus	A-215	700		3	Downtown	A-101	500
3	Downtown	A-101	500		4	Redwood	A-222	700
4	Redwood	A-222	700		5	Perryridge	A-201	900
5	Perryridge	A-201	900		6	Brighton	A-217	750
6	Brighton	A-217	750		7	Downtown	A-110	600
7	Downtown	A-110	600		8	Perryridge	A-218	700
8	Perryridge	A-218	700					
1. File F chứa các mẫu tin account					2. File F sau khi xoá mẫu tin 2 và di chuyển các mẫu tin sau nó			

16

## Mẫu tin có độ dài cố định (Fixed-Length Records)

### ❑ Các vấn đề nảy sinh tiếp:

- Phải truy xuất khối bổ sung. Vì hoạt động xen xảy ra thường xuyên hơn hoạt động xóa, ta có thể chấp nhận việc để "ngỏ" không gian bị chiếm bởi mẫu tin bị xóa, và chờ một hoạt động xen đến sau để tái sử dụng không gian đó.
- Đánh dấu mẫu tin bị xóa là không khả thi vì sẽ gây khó khăn cho việc tìm kiếm không gian "tự do" đó khi xen

17

## Mẫu tin có độ dài cố định (Fixed-Length Records)

### ❑ Cách giải quyết:

- Tạo con trỏ header của file để quản lý các mẫu tin bị xóa.

18

## Mẫu tin có độ dài cố định (Fixed-Length Records)

0	Perryridge	A-102	400		header			
1	Round Hill	A-305	350		0	Perryridge	A-002	400
8	Perryridge	A-218	700		1			
3	Downtown	A-101	500		2	Mianus	A-215	700
4	Redwood	A-222	700		3	Downtown	A-101	500
5	Perryridge	A-201	900		4			
6	Brighton	A-217	750		5	Perryridge	A-201	900
7	Downtown	A-110	600		6			
3. File F sau khi xóa mẫu tin 2 và di chuyển mẫu tin cuối vào vị trí của header					7	Downtown	A-110	600
					8	Perryridge	A-218	700

19

## Mẫu tin có độ dài thay đổi (Variable-Length Records)

### ❑ type

```

depositor = record
    branch_name : char(20) ;
    account_info : array[ 1.. ∞ ] of record
        account_number: char(10) ;
        balance : real;
    end
end
end

```

20

## Mẫu tin có độ dài thay đổi (Variable-Length Records)

### ☐ Biểu diễn bằng chuỗi byte

- Gắn một ký hiệu đặc biệt  $\perp$  (End-of-record) vào cuối mỗi mẫu tin.
- Lưu mỗi mẫu tin như một chuỗi byte liên tiếp

0	Perryridge	A-102	400	A-201	900	A-210	700	$\perp$
1	Round Hill	A-301	350	$\perp$				
2	Mianus	A-101	800	$\perp$				
3	Downtown	A-211	500	A-222	600	$\perp$		
4	Redwood	A-300	650	A-220	1200	A-255	950	$\perp$
5	Brighton	A-111	750	$\perp$				

21

## Mẫu tin có độ dài thay đổi (Variable-Length Records)

### ☐ Biểu diễn bằng mẫu tin độ dài cố định

- Sử dụng một hoặc một vài mẫu tin độ dài cố định để biểu diễn một mẫu tin độ dài thay đổi
- Hai kỹ thuật là *Không gian dự trữ* (reserved space) và *Con trỏ* (Pointers).

22

## Mẫu tin có độ dài thay đổi (Variable-Length Records)

### ☐ Biểu diễn bằng mẫu tin độ dài cố định

- Kỹ thuật Không gian dự trữ:

0	Perryridge	A-102	400	A-201	900	A-210	700	$\perp$
1	Round Hill	A-301	350	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
2	Mianus	A-101	800	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
3	Downtown	A-211	500	A-222	600	$\perp$	$\perp$	$\perp$
4	Redwood	A-300	650	A-200	1200	A-255	950	$\perp$
5	Brighton	A-111	750	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

Sử dụng phương pháp không gian dự trữ

23

## Mẫu tin có độ dài thay đổi (Variable-Length Records)

### ☐ Biểu diễn bằng mẫu tin độ dài cố định

- Kỹ thuật con trỏ:

- ☐ Nêu các bất lợi?
- ☐ Nêu hướng giải quyết?

0	Perryridge	A-102	400	•	
1		A-201	900	•	←
2		A-210	700	•	←
3	Round Hill	A-301	350	•	
4	Mianus	A-101	800	•	
5	Downtown	A-211	500	•	
6		A-222	600	•	←
7	Redwood	A-300	650	•	
8		A-200	1200	•	←
9		A-255	950	•	←
10	Brighton	A-111	750	•	

24

### Tổ chức các mẫu tin trong file

- ❑ **Tổ chức file đồng (Heap File Organization):** Một mẫu tin bất kỳ có thể được lưu trữ ở bất kỳ nơi nào trong file, ở đó có không gian cho nó. Không có thứ tự nào giữa các mẫu tin. Mỗi file cho mỗi quan hệ.
- ❑ **Tổ chức file tuần tự (Sequential File Organization):** Các mẫu tin được lưu trữ tuần tự liên tiếp dựa trên thứ tự "giá trị khoá tìm kiếm" của mỗi mẫu tin.

25

### Tổ chức các mẫu tin trong file

- ❑ **Tổ chức file băm (Hashed File Organization):** Có một hàm băm được áp dụng trên thuộc tính nào đó của mẫu tin. Kết quả của hàm băm xác định mẫu tin được bố trí vào khối nào trong file. Tổ chức này liên hệ chặt chẽ với "cấu trúc chỉ mục".
- ❑ **Tổ chức file cụm (Clustering File Organization):** Các mẫu tin của một vài quan hệ khác nhau được lưu trữ trong cùng một file. Các mẫu tin có liên hệ của các quan hệ khác nhau được lưu trữ trên cùng một khối sao cho một hoạt động I/O đem lại các mẫu tin kết nối có liên hệ từ tất cả các quan hệ.

26

### Tập tin tuần tự

- ❑ **Tổ chức:** một danh sách liên kết của các khối, các mẫu tin được lưu trữ trong các khối theo một thứ tự bất kỳ.
- ❑ **Tìm mẫu tin:**
  - Đọc từng khối, với mỗi khối ta tìm mẫu tin cần tìm trong khối, nếu không tìm thấy ta lại đọc tiếp một khối khác.
  - Quá trình cứ tiếp tục cho đến khi tìm thấy mẫu tin hoặc duyệt qua toàn bộ các khối của tập tin và trong trường hợp đó thì mẫu tin không tồn tại trong tập tin.

27

### Tập tin tuần tự (tt)

- ❑ **Thêm mẫu tin mới:**
  - Đưa mẫu tin này vào khối cuối cùng của tập tin nếu như khối đó còn chỗ trống.
  - Ngược lại nếu khối cuối cùng đã hết chỗ thì xin cấp thêm một khối mới, thêm mẫu tin vào khối mới và nối khối mới vào cuối danh sách.
- ❑ **Sửa đổi mẫu tin:** Tìm mẫu tin cần sửa, thực hiện các sửa đổi cần thiết sau đó ghi lại mẫu tin vào vị trí cũ trong tập tin.

28

## Tập tin tuần tự (tt)

- **Xoá mẫu tin:**
  - Tìm mẫu tin đó, nếu tìm thấy ta có thể thực hiện một trong các cách xoá sau đây:
  - Một là xoá mẫu tin cần xoá trong khối lưu trữ nó, nếu sau khi xoá, khối trở nên rỗng thì xoá khỏi danh sách.
  - Hai là đánh dấu xoá mẫu tin bằng một trong hai cách:
    - Thay thế mẫu tin bằng một giá trị nào đó mà giá trị này không bao giờ là giá trị thật của bất kỳ một mẫu tin nào.
    - Sử dụng bit xoá

29

## Đánh giá tập tin tuần tự

- Đây là một phương pháp tổ chức tập tin đơn giản nhất nhưng kém hiệu quả nhất.
- Giả sử tập tin có  $n$  mẫu tin và mỗi khối lưu trữ được  $k$  mẫu tin thì toàn bộ tập tin được lưu trữ trong  $n/k$  khối.
- Do đó mỗi lần tìm (hoặc thêm hoặc sửa hoặc xoá) một mẫu tin thì phải truy xuất  $n/k$  khối.

30

## Tăng tốc độ cho các thao tác tập tin

- Để cải thiện tốc độ thao tác trên tập tin, chúng ta phải tìm cách giảm số lần truy xuất khối.
- Muốn vậy phải tìm các cấu trúc sao cho khi tìm một mẫu tin chỉ cần truy xuất một số nhỏ các khối.
- Để tạo ra các tổ chức tập tin như vậy chúng ta phải giả sử rằng mỗi mẫu tin có một khoá (key).
- Hai mẫu tin khác nhau thì khoá của chúng phải khác nhau.

31

## Lưu trữ tự điển dữ liệu (Data Dictionary)

- Một hệ CSDL cần thiết duy trì dữ liệu thông tin
  - Các quan hệ
  - Các sơ đồ quan hệ và một số cái cần thiết.

Các thông tin trên được gọi là "tự điển dữ liệu" hay "mục lục hệ thống" (System Catalog)

32



## Lưu trữ tự điển dữ liệu (Data Dictionary)

### □ Thông tin hệ thống gồm

- Các tên của các quan hệ
- Các tên của các thuộc tính của mỗi quan hệ
- Các miền giá trị và các độ dài của các thuộc tính
- Các tên của các View được định nghĩa trên CSDL và định nghĩa của các view này
- Các ràng buộc toàn vẹn

33

## Chỉ mục (Indices / Indexes)

- “Chỉ mục” của một file được hiểu như là một “cấu trúc dữ liệu” giúp cho việc tìm kiếm trên file đó được nhanh chóng
- Có hai kiểu chỉ mục :
  - *Chỉ mục được sắp (Ordered Indices)*: Là chỉ mục dựa trên một thứ tự sắp xếp theo các giá trị được chọn.
  - *Chỉ mục băm (Hash indices)*: Là chỉ mục dựa trên các giá trị được phân phối đều qua các bucket (cái thùng). Bucket mà một giá trị được gán được xác định bởi một hàm, gọi là hàm băm (Hash Function)

34

## Chỉ mục (Indices / Indexes)

- Mỗi kỹ thuật phải được đánh giá trên cơ sở của các nhân tố sau:
  - *Kiểu truy xuất*: Các kiểu truy xuất được hỗ trợ hiệu quả. Các kiểu này bao hàm cả tìm kiếm mẫu tin với một giá trị thuộc tính cụ thể hoặc tìm các mẫu tin với giá trị thuộc tính nằm trong một khoảng xác định.
  - *Thời gian truy xuất*: Thời gian để tìm kiếm một hạng mục dữ liệu hay một tập các hạng mục.
  - *Thời gian xen*: Thời gian để xen một hạng mục dữ liệu mới. Giá trị này bao hàm thời gian để tìm vị trí xen thích hợp và thời gian cập nhật cấu trúc chỉ mục.
  - *Thời gian xoá*: Thời gian để xoá một hạng mục dữ liệu. Giá trị này bao hàm thời gian tìm kiếm hạng mục cần xoá, thời gian cập nhật cấu trúc chỉ mục.
  - *Tổng phí tổn không gian*: Không gian phụ bị chiếm bởi một cấu trúc chỉ mục.

35

## Chỉ mục được sắp

- Mỗi chỉ mục được sắp của một file lưu trữ các giá trị khoá tìm kiếm theo thứ tự được sắp và kết hợp với mỗi khoá tìm kiếm các mẫu tin chứa khoá tìm kiếm này.
- Các mẫu tin trong file có chỉ mục được sắp có thể chính bản thân nó cũng được sắp.
- Một file có thể có một vài chỉ mục được sắp trên những khoá tìm kiếm khác nhau

36

### Chỉ mục được sắp

- ❑ Nếu file chứa các mẫu tin được sắp thứ tự tuần tự thì chỉ mục trên khoá tìm kiếm xác định thứ tự này của file được gọi "**chỉ mục sơ cấp**" (**Primary index**).
- ❑ Các chỉ mục sơ cấp cũng được gọi là "**chỉ mục cụm**" (**Clustering Index**).
- ❑ Khoá tìm kiếm của chỉ mục sơ cấp thường là khoá sơ cấp (hay khoá chính).

37

### Chỉ mục được sắp

- ❑ Các chỉ mục được sắp mà khoá tìm kiếm của nó xác định một thứ tự khác với thứ tự của các mẫu tin trong file được gọi là các "**chỉ mục thứ cấp**" (**Secondary Indices**) hay các "**chỉ mục không cụm**" (**Nonclustering Indices**).

38

### Chỉ mục được sắp

Brighton	A-217	750	•	←
Downtown	A-101	500	•	←
Downtown	A-110	600	•	←
Mianus	A-215	700	•	←
Perryridge	A-102	400	•	←
Perryridge	A-201	900	•	←
Perryridge	A-218	700	•	←
Redwood	A-222	850	•	←
Round Hill	A-301	550	•	←

file tuần tự các mẫu tin account

39

### Chỉ mục sơ cấp (Primary Index)

- ❑ Có hai loại chỉ mục sơ cấp là
  - **Chỉ mục đặc**
  - **Chỉ mục thừa**

40

## Chi mục đặc

- ❑ Là chi mục mà một "mẫu tin chi mục" (index entry) xuất hiện trong file chi mục đối với mỗi giá trị khoá tìm kiếm trong file dữ liệu.
- ❑ Mỗi mẫu tin chi mục chứa hai trường: một là giá trị khoá tìm kiếm và hai là một con trỏ tới mẫu tin dữ liệu đầu tiên với giá trị khoá tìm kiếm đó

41

## Chi mục đặc

Chi mục đặc					
Brighton	•	Brighton	A-217	750	•
Downtown	•	Downtown	A-101	500	•
Downtown	•	Downtown	A-110	600	•
Mianus	•	Mianus	A-215	700	•
Perryridge	•	Perryridge	A-102	400	•
Perryridge	•	Perryridge	A-201	900	•
Redwood	•	Perryridge	A-218	700	•
Round Hill	•	Redwood	A-222	850	•
		Round Hill	A-301	550	•

42

## Chi mục thưa

- ❑ Là chi mục mà một mẫu tin chi mục được tạo ra **chỉ với một số giá trị khóa tìm kiếm** trong file dữ liệu.
- ❑ Mỗi mẫu tin chi mục chứa một giá trị khoá tìm kiếm và một con trỏ tới mẫu tin dữ liệu đầu tiên với giá trị khoá tìm kiếm này.

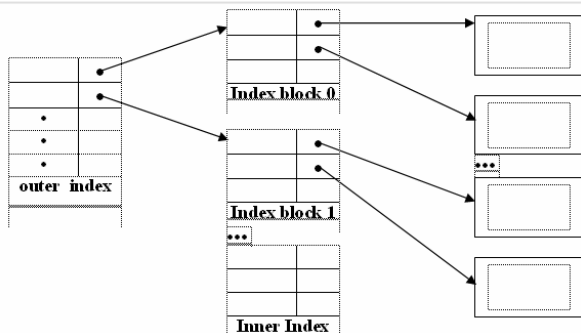
43

## Chi mục thưa

Chi mục thưa					
Brighton	•	Brighton	A-217	750	•
Downtown	•	Downtown	A-101	500	•
Downtown	•	Downtown	A-110	600	•
Mianus	•	Mianus	A-215	700	•
Redwood	•	Perryridge	A-102	400	•
		Perryridge	A-201	900	•
		Perryridge	A-218	700	•
		Redwood	A-222	850	•
		Round Hill	A-301	550	•

44

## Chỉ mục nhiều mức



45

## Chỉ mục thứ cấp

### Chỉ mục thứ cấp trên một khoá dự tuyển

- Giống như chỉ mục sơ cấp đặc (ngoại trừ các mẫu tin được trỏ đến bởi các giá trị liên tiếp trong chỉ mục không được lưu trữ tuần tự).

### Nếu khoá tìm kiếm của chỉ mục sơ cấp không là khoá dự tuyển

- Chỉ mục chỉ cần trỏ đến mẫu tin đầu tiên với một giá trị khoá tìm kiếm riêng là đủ (các mẫu tin khác cùng giá trị khoá này có thể tìm lại được nhờ quét tuần tự file).

46

## Chỉ mục thứ cấp

### Nếu khoá tìm kiếm của một chỉ mục thứ cấp không là khoá dự tuyển

- Việc trỏ tới mẫu tin đầu tiên với giá trị khoá tìm kiếm riêng không đủ, do các mẫu tin trong file không còn được sắp tuần tự theo khoá tìm kiếm của chỉ mục thứ cấp, chúng có thể nằm ở bất kỳ vị trí nào trong file.
- Chỉ mục thứ cấp phải chứa tất cả các con trỏ tới mỗi mẫu tin.
- Ta có thể sử dụng mức phụ gián tiếp để thực hiện chỉ mục thứ cấp trên các khoá tìm kiếm không là khoá dự tuyển.
- Các con trỏ trong chỉ mục thứ cấp như vậy không trực tiếp trỏ tới mẫu tin mà trỏ tới một bucket chứa các con trỏ tới file

47

## Chỉ mục thứ cấp



48

## Cây tìm kiếm m-phân

- Cây tìm kiếm m-phân (m-ary tree) là sự tổng quát hoá của cây tìm kiếm nhị phân trong đó mỗi nút có thể có m nút con.
- Giả sử  $n_1$  và  $n_2$  là hai con của một nút nào đó,  $n_1$  bên trái  $n_2$  thì tất cả các con của  $n_1$  có giá trị nhỏ hơn giá trị của các nút con của  $n_2$ .

49

## B-cây

- B-cây bậc m là cây tìm kiếm m-phân cân bằng có các tính chất sau:
  - Nút gốc hoặc là lá hoặc có ít nhất hai nút con,
  - Mỗi nút, trừ nút gốc và nút lá, có từ  $\lceil m/2 \rceil$  đến m nút con và
  - Các đường đi từ gốc tới lá có cùng độ dài.

50

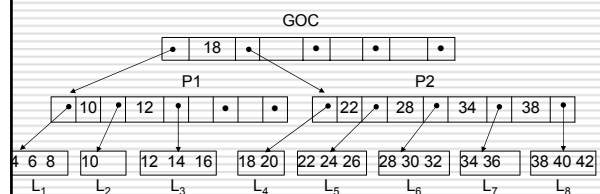
## Tập tin B-cây: Tổ chức

- Mỗi nút trên cây là một khối trên đĩa, các mẫu tin của tập tin được lưu trữ trong các nút lá trên B-cây và lưu theo thứ tự của khóa.
- Giả sử mỗi nút lá lưu trữ được nhiều nhất b mẫu tin.
- Mỗi nút không phải là nút lá có dạng  $(p_0, k_1, p_1, k_2, p_2, \dots, k_n, p_n)$ , với  $p_i$  ( $0 \leq i \leq n$ ) là con trỏ, trỏ tới nút con thứ i của nút đó và  $k_i$  là các giá trị khóa. Các khóa trong một nút được sắp thứ tự, tức là  $k_1 < k_2 < \dots < k_n$ .
- Tất cả các khóa trong cây con được trỏ bởi  $p_0$  đều nhỏ hơn  $k_1$ .
- Tất cả các khóa nằm trong cây con được trỏ bởi  $p_i$  ( $0 < i < n$ ) đều lớn hơn hoặc bằng  $k_i$  và nhỏ hơn  $k_{i+1}$ .
- Tất cả các khóa nằm trong cây con được trỏ bởi  $p_n$  đều lớn hơn hoặc bằng  $k_n$ .

51

## Tập tin B-cây: Ví dụ

Cho tập tin bao gồm 20 mẫu tin với giá trị khóa là các số nguyên được tổ chức thành B-cây bậc 5 với các nút lá chứa được nhiều nhất 3 mẫu tin.



52

## Tập tin B-cây: Tìm mẫu tin

- Bắt đầu gốc đến nút lá chứa  $r$  (nếu  $r$  tồn tại trong tập tin).
- Tại mỗi bước, đưa nút trong ( $p_0, k_1, p_1, \dots, k_n, p_n$ ) vào bộ nhớ trong và xác định mối quan hệ giữa  $x$  với các giá trị khóa  $k_i$ .
  - Nếu  $k_i \leq x < k_{i+1}$  ( $0 < i < n$ ) chúng ta sẽ xét tiếp nút được trỏ bởi  $p_i$ .
  - Nếu  $x < k_1$  ta sẽ xét tiếp nút được trỏ bởi  $p_0$ .
  - Nếu  $x \geq k_n$  ta sẽ xét tiếp nút được trỏ bởi  $p_n$ .
- Quá trình trên sẽ dẫn đến việc xét một nút lá.
- Trong nút lá này ta sẽ tìm mẫu tin  $r$  với khóa  $x$  bằng tìm kiếm tuần tự hoặc tìm kiếm nhị phân.

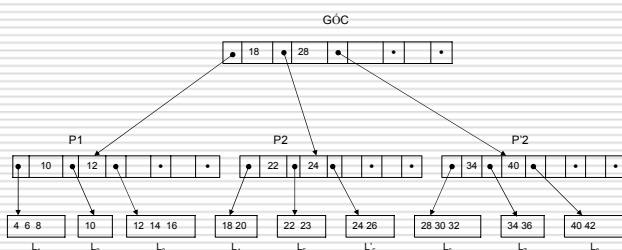
53

## Tập tin B-cây: Thêm mẫu tin mới

- Tìm  $r$ . Việc tìm kiếm này sẽ dẫn đến nút lá  $L$ .
- Nếu tìm thấy thì thông báo "Mẫu tin đã tồn tại", ngược lại thì  $L$  là nút lá mà ta có thể xen  $r$  vào trong đó.
- Nếu khối  $L$  này còn chỗ cho  $r$  thì ta thêm  $r$  vào sao cho đúng thứ tự của nó trong khối  $L$  và giải thuật kết thúc.
- Nếu  $L$  không còn chỗ thì cấp phát một khối mới  $L'$ .
- Dời  $\lceil b/2 \rceil$  mẫu tin nằm ở cuối khối  $L$  sang  $L'$ , rồi xen  $r$  vào  $L$  hoặc  $L'$  sao cho việc xen đảm bảo thứ tự các khóa trong khối.
- Giả sử nút  $P$  là cha của  $L$ .
- Xen đệ quy để xen vào  $P$  một khóa  $k'$  và con trỏ  $p'$  tương ứng của  $L'$ .
- Trong trường hợp trước khi xen  $k'$  và  $p'$ ,  $P$  đã có đủ  $m$  con thì ta phải cấp thêm một khối mới  $P'$  và chuyển một số con của  $P$  sang  $P'$  và xen con mới vào  $P$  hoặc  $P'$  sao cho cả  $P$  và  $P'$  đều có ít nhất  $\lceil m/2 \rceil$  con.
- Việc chia cắt  $P \Rightarrow$  phải xen một khóa và một con trỏ vào nút cha của  $P$ ...
- Quá trình này có thể sẽ dẫn tới nút gốc và cũng có thể phải chia cắt nút gốc, trong trường hợp này phải tạo ra một nút gốc mới mà hai con của nó là hai nửa của nút gốc cũ.
- Khi đó chiều cao của B-cây sẽ tăng lên 1.

54

## Kết quả xen 23



55

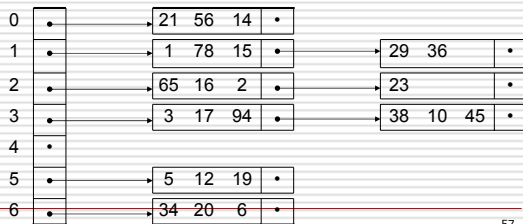
## Tập tin băm (hash files): Tổ chức

- Bảng băm là mảng một chiều  $B$  gồm  $m$  phần tử  $B[0], B[1], \dots, B[m-1]$ .
- Mỗi phần tử là một con trỏ, trỏ tới phần tử đầu tiên của danh sách liên kết các khối.
- Để phân phối mẫu tin có khóa  $x$  vào trong các danh sách liên kết, ta dùng hàm băm.
- $h(x) = x \text{ MOD } m$ .

56

## Ví dụ về tập tin bảng băm

- Một tập tin có 24 mẫu tin với giá trị khóa là các số nguyên: 3, 5, 12, 65, 34, 20, 21, 17, 56, 1, 16, 2, 78, 94, 38, 15, 23, 14, 10, 29, 19, 6, 45, 36
- Giả sử chúng ta có thể tổ chức tập tin này vào trong bảng băm gồm 7 phần tử và giả sử mỗi khối có thể chứa được tối đa 3 mẫu tin. Với mỗi mẫu tin  $r$  có khóa là  $x$  ta xác định  $h(x) = x \text{ MOD } 7$  và đưa mẫu tin  $r$  vào trong một khối của danh sách liên kết được trỏ bởi  $B[h(x)]$ .



57

## Tập tin băm: Tìm mẫu tin

- Để tìm một mẫu tin  $r$  có khóa là  $x$ , chúng ta xác định  $h(x)$  chẳng hạn  $h(x) = i$ , khi đó ta chỉ cần tìm  $r$  trong danh sách liên kết được trỏ bởi  $B[i]$ .
- Chẳng hạn để tìm mẫu tin  $r$  có khóa là 36, ta tính  $h(36) = 36 \text{ MOD } 7 = 1$ .
- Như vậy nếu mẫu tin  $r$  tồn tại trong tập tin thì nó phải thuộc một khối nào đó được trỏ bởi  $B[1]$ .

58

## Tập tin băm: Thêm mẫu tin mới

- Để thêm mẫu tin  $r$  có khóa  $x$ , trước hết ta phải tìm mẫu tin  $r$ .
- Nếu tìm thấy thì thông báo "mẫu tin đã tồn tại"
- Ngược lại ta sẽ tìm một khối (trong danh sách các khối của lô được trỏ bởi  $B[h(x)]$ ) còn chỗ trống và thêm  $r$  vào khối này.
- Nếu không còn khối nào đủ chỗ cho mẫu tin mới ta yêu cầu hệ thống cấp phát một khối mới và đặt mẫu tin  $r$  vào khối này rồi nối khối mới này vào cuối danh sách liên kết của lô.

59

## Tập tin băm: Xóa mẫu tin

- Để xóa mẫu tin  $r$  có khóa  $x$ , trước hết ta phải tìm mẫu tin này.
- Nếu không tìm thấy thì thông báo "Mẫu tin không tồn tại".
- Nếu tìm thấy thì đặt bit xóa cho nó.
- Ta cũng có thể xóa hẳn mẫu tin  $r$  và nếu việc xóa này làm khối trở nên rỗng thì ta giải phóng khối này (xóa khối khỏi danh sách liên kết các khối).

60

## Tập tin băm: Đánh giá

---

- ❑ Giả sử tập tin có  $n$  mẫu tin và mỗi khối lưu trữ được  $k$  mẫu tin thì tập tin cần  $n/k$  khối.
- ❑ Trung bình mỗi danh sách liên kết (mỗi lô) của bảng băm có  $n/mk$  khối (do bảng băm có  $m$  lô), mà chúng ta chỉ tìm trong một danh sách liên kết nên ta chỉ phải truy xuất  $n/mk$  khối.
- ❑ Số này nhỏ hơn  $m$  lần so với cách tổ chức tập tin tuần tự (trong tập tin tuần tự ta cần truy xuất tất cả các khối, tức là  $n/k$  khối).