Chương 4

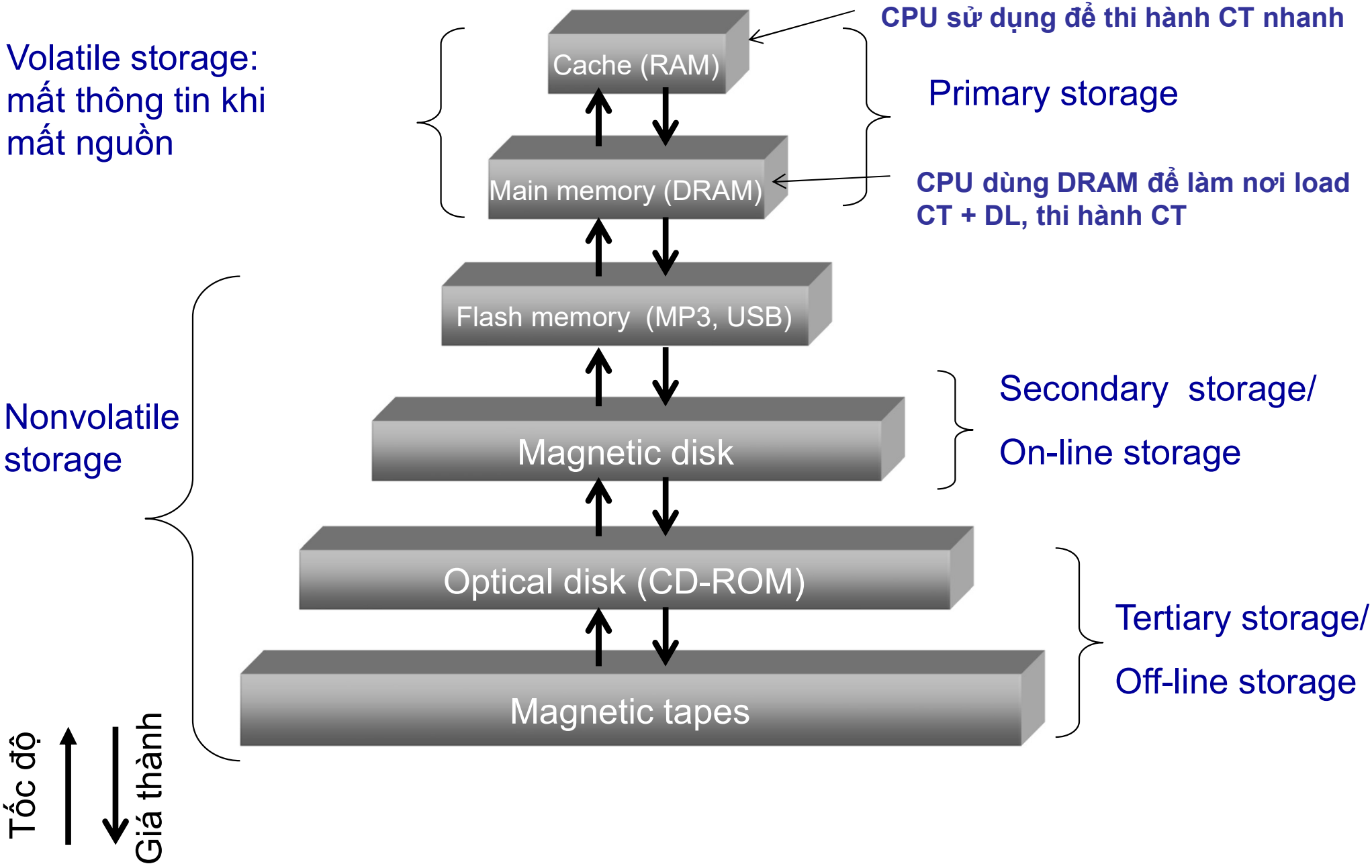
Lưu trữ dữ liệu vật lý &

Các phương pháp truy xuất

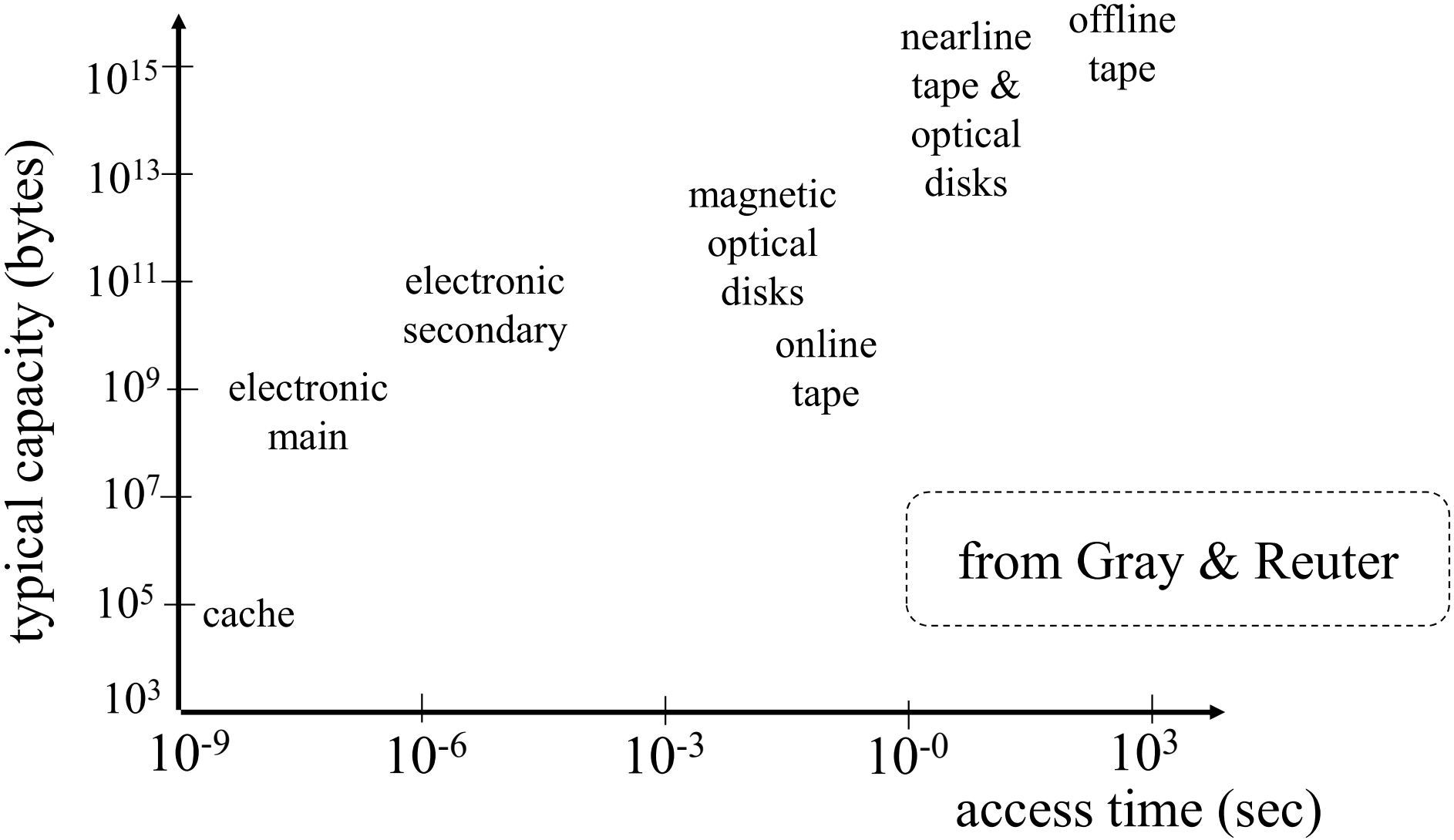
Nội dung

1. Một số khái niệm
2. Cách tổ chức file và phương pháp truy xuất
3. Index

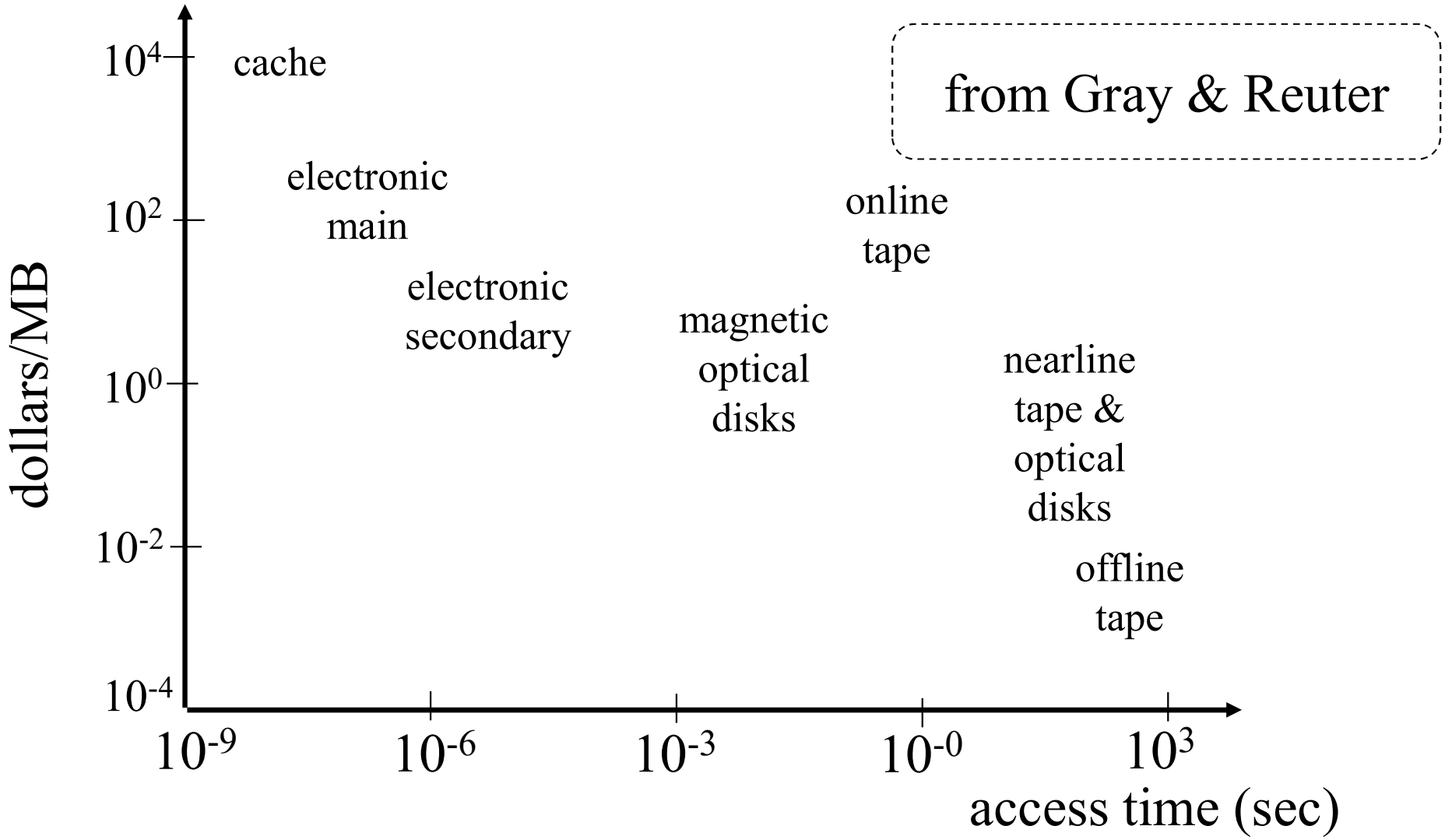
Các phương tiện lưu trữ DL



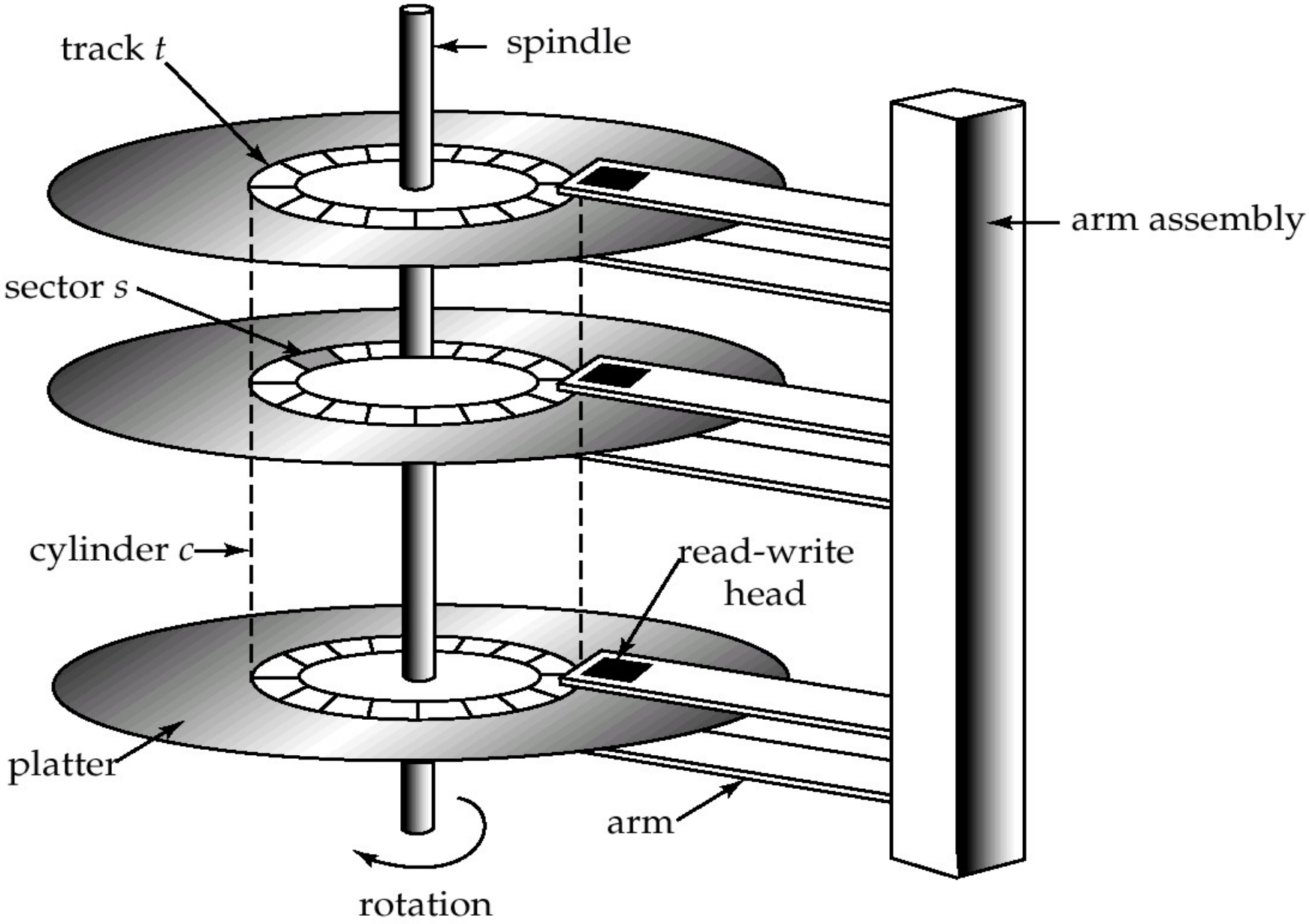
Storage Capacity



Storage Cost



Đĩa từ (Magnetic disk)



Về cách quản lý đĩa

- 1 mặt đĩa chia thành nhiều track, 1 track chia thành nhiều block (page). 1 cluster = n block.
- Dùng đĩa từ (magnetic disk) để lưu cơ sở dữ liệu vì:
  - Khối lượng lưu trữ lớn (không thể lưu ở bộ nhớ chính)
  - Lưu một cách bền bỉ, lâu dài, phục vụ cho truy cập và xử lý lặp lại (bộ nhớ chính không đáp ứng được)
  - Chi phí cho việc lưu trữ rẻ.
- Dữ liệu trên đĩa phải được chép vào bộ nhớ chính khi cần xử lý. Nếu dữ liệu này có thay đổi thì sẽ được ghi trở lại vào đĩa.
- Bộ điều khiển đĩa (disk controller - DC): giao tiếp giữa ổ đĩa và máy tính, nhận 1 lệnh I/O, định vị đầu đọc và làm cho hành động R/W diễn ra.
- Block cũng là đơn vị để lưu trữ và chuyển dữ liệu.

Chuyển dữ liệu

- Thời gian trung bình để tìm và chuyển 1 block = s + rd + btt
  - Seek time (s): để DC định vị đầu đọc/ ghi đúng track, trung bình khỏang 7-10 msec (desktop), 3-8 msec (server).
  - Rotational delay/latency (rd): để đầu đọc ở vị trí block cần đọc, phụ thuộc rpm, trung bình khỏang 2 msec.
  - Block transfer time (btt): để chuyển dữ liệu, phụ thuộc vào block size, track size, và rpm.
- Khi truy xuất đến các block liên tiếp thì tiết kiệm được thời gian.
- Một số kỹ thuật tìm kiếm khai thác điều này.

Một số nguyên tắc

- DBS giảm thiểu số lượng block được chuyển giữa đĩa và MM (main memory)→ giảm số lần truy xuất đĩa
  - Lưu lại càng nhiều càng tốt các block dữ liệu trên MM, tăng cơ hội tìm thấy block cần truy xuất trên MM.
- Buffer là thành phần của MM dùng để chứa các bản sao (version mới hơn) của các block được đọc lên/ lưu xuống đĩa, do Buffer manager quản lý.

Lưu tập tin trên đĩa

- CSDL được tổ chức trên đĩa thành một/nhiều tập tin, mỗi tập tin gồm nhiều mẩu tin, mỗi mẩu tin gồm nhiều trường.
- Mẩu tin phải được lưu trữ trên đĩa sao cho khi cần thì có thể truy cập được và truy cập một cách hiệu quả.
  - Cách tổ chức file chính (Primary file organization): cho biết các mẩu tin định vị một cách vật lý như thế nào trên đĩa, từ đó biết cách truy xuất chúng.
  - Cách tổ chức phụ (secondary organization/ auxiliary access structure): để truy cập các mẩu tin trên file hiệu quả.

Mục đích

- Kỹ thuật lưu trữ dữ liệu có ích cho người thiết kế CSDL, DBA, người cài đặt HQTCSDL.
  - Người thiết kế CSDL, DBA: biết ưu khuyết điểm của từng kỹ thuật lưu trữ để thiết kế, hiện thực và thao tác CSDL trên 1 HQTCSDL cụ thể.
    - Đặc điểm của đĩa từ + cách tổ chức file dữ liệu trên đĩa từ → Đưa ra cách thiết kế CSDL để có thể lưu trữ và khai thác hiệu quả.
    - HQTCSDL thường có nhiều chọn lựa để tổ chức DL, việc thiết kế vật lý cần chọn kỹ thuật tổ chức dữ liệu phù hợp cho yêu cầu ứng dụng.
  - Người cài đặt CSDL cần biết kỹ thuật tổ chức DL và cài đặt đúng, hiệu quả để cung cấp cho DBA và người dùng đầy đủ các chọn lựa.

Nội dung

1. Một số khái niệm
2. Cách tổ chức file và phương pháp truy xuất
   - Mẩu tin, kiểu mẩu tin, tập tin, mẩu tin có kích thước cố định, mẩu tin có kích thước thay đổi
   - Định vị file block trên đĩa
   - File header
   - Thao tác trên file
   - Heap file, Sorted file, Hashing technique
3. Index

Mẩu tin (Record)

- Data ≡ records ≡ data values/ items
  ≡ thực thể/mối quan hệ và các thuộc tính của chúng
- Kiểu mẩu tin: Record type/ Record format = tập các tên thuộc tính và kiểu dữ liệu của từng thuộc tính.
- VD: struct NHANVIEN {char TENNV[30];
  char MANV[9];
  int LUONG;
  char PHONG[20];}

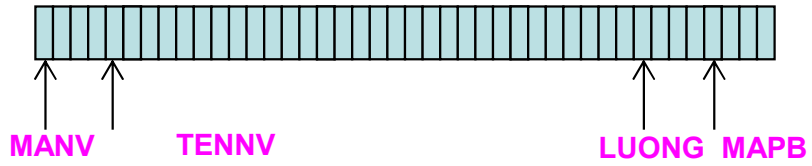
- Kiểu dữ liệu (Data type) của từng thuộc tính cho biết giá trị mà field có thể nhận lấy.
  - Số: integer(4 bytes), long integer (8 bytes), floating point (4 bytes)
  - Chuỗi (1 ký tự ≡ 1 byte): chiều dài cố định hoặc thay đổi
  - Boolean (1 byte)
  - Date/ time (YYYY-MM-DD: 10 byte)

Tập tin

- Tổ chức tập tin:
  - Lưu nhiều mẩu tin, có cùng kiểu hoặc khác kiểu mẩu tin.
  - Gồm các mẩu tin có cùng chiều dài (byte) → tập tin có kích thước các mẩu tin cố định (fixed-length record) hoặc
  - Gồm các mẩu tin có chiều dài khác nhau → tập tin có kích thước các mẩu tin thay đổi (variable-length record)
    - TH1: Cùng kiểu mẩu tin, có field có chiều dài thay đổi. VD: field kiểu chuỗi
    - TH2: Do khác kiểu mẩu tin, là trường hợp các mẩu tin có liên quan được gom nhóm lại (clustered) và lưu trên cùng block để truy xuất nhanh.
    - TH3: Cùng kiểu mẩu tin, có thuộc tính có thể có nhiều giá trị khác nhau.
    - TH4: Cùng kiểu mẩu tin, có thuộc tính có/ không có giá trị.
    - Ghi chú: TH3 và TH4 không xảy ra khi lưu trữ dữ liệu trên CSDL quan hệ.

Tập tin

- Mẩu tin có kích thước cố định: dễ truy xuất



- Mẩu tin có kích thước thay đổi
  - TH1:
    - Dùng ký tự đặc biệt (không lẫn lộn với giá trị thuộc tính) để kết thúc các trường có chiều dài thay đổi hoặc
    - Lưu kích thước thật tính bằng byte ngay trước giá trị thuộc tính.

  - TH2: Trước mỗi mẩu tin lưu kiểu mẩu tin.
  - TH3: Cần 1 ký tự đặc biệt ngăn cách giữa các giá trị lặp lại và 1 ký tự kết thúc.
  - TH4:
    - Nếu số thuộc tính nhiều nhưng số lượng các thuộc tính thực sự có giá trị là ít thì có thể lưu cặp <field-name, field-value> thay vì chỉ lưu field-value.
    - Dùng 3 ký tự đặc biệt để ngăn cách: giữa field-name và field-value, giữa các field với nhau, và kết thúc record. Có thể dùng cùng 1 ký tự đặc biệt cho 2 mục đích đầu.
    - Hoặc đánh mã cho kiểu dữ liệu của từng field là field-type, là số nguyên chẳng hạn, và lưu <field-type, field-value>

Tập tin

- Ta biết: đĩa được chia ra thành các ***block.***
- Thường:

  Kích thước block B > kích thước record R (cố định, tính bằng byte, B>=R)

- 1 block có chứa nhiều mẩu tin, số lượng:

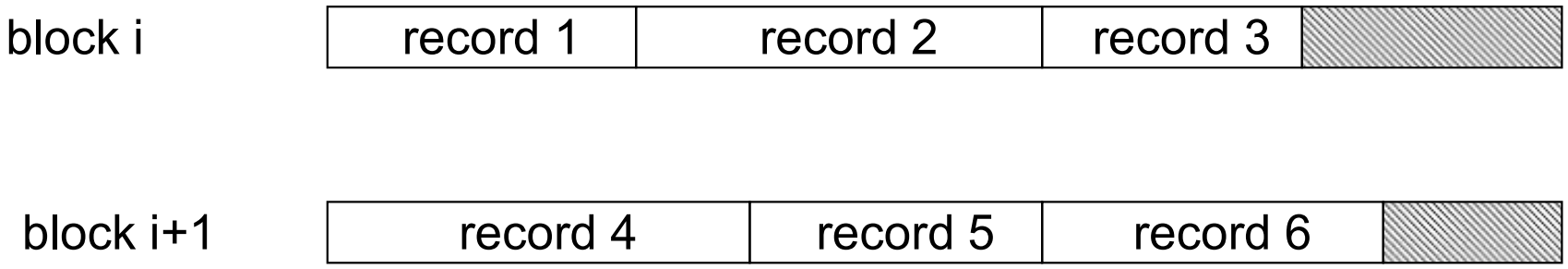
  bfr= floor(B/R) records/block

  bfr gọi là blocking factor của file

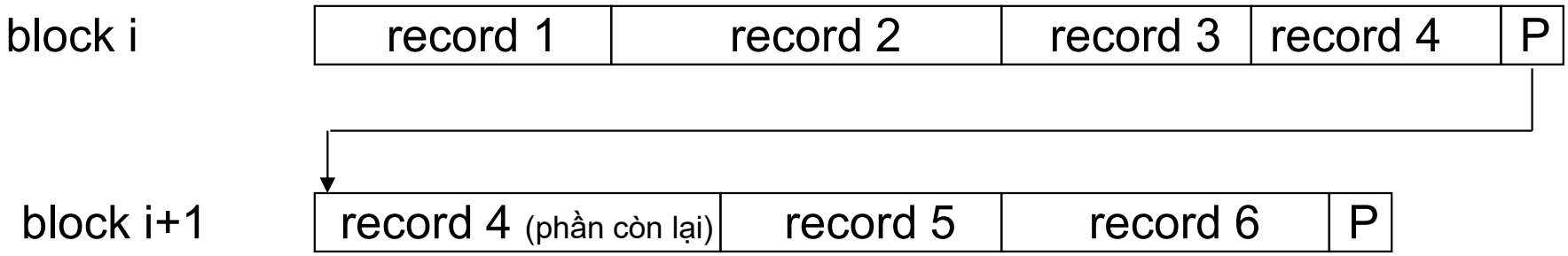
  B - bfr*R là số byte không dùng trong 1 block.

Lưu mẩu tin vào block

- Unspanned



- Spanned



Tổ chức file block trên đĩa

- Liên tục: Các block của file định vị liên tiếp trên các block của đĩa.
  - Đọc file nhanh dùng double buffering.
    - Trong khi CPU xử lý 1 block thì bộ xử lý I/O đọc và chuyển block kế tiếp vào buffer khác.
  - Mở rộng file khó khăn.
- Không liên tục: từng block của file chứa con trỏ trỏ tới block kế tiếp.
  - Dễ mở rộng file.
  - Đọc file chậm.
- Kết hợp hai cách trên.
  - Mỗi cluster là các disk block liên tiếp nhau, các cluster liên kết với nhau.
  - Cluster còn được gọi là file segment hoặc extent.

File Header

- Lưu lại thông tin cần thiết để có thể truy cập file:
  - Địa chỉ của các block của file.
  - Mô tả định dạng của record.
    - Tập tin gồm mẩu tin có chiều dài cố định: Field-length, thứ tự field đối với unspanned record.
    - Tập tin gồm mẩu tin có chiều dài thay đổi: mã kiểu cho field, ký tự ngăn cách, mã kiểu cho mẩu tin.
  - Tìm 1 mẩu tin trên đĩa:
    - 1 hoặc 1 số block được chép vào buffer.
    - CTrình tìm trên buffer dùng thông tin của file header.

Thao tác trên file

- Các thao tác trên tập tin
  - Tìm kiếm 1 mẩu tin theo điều kiện
  - Thêm 1 mẩu tin
  - Xóa 1 mẩu tin
  - Sửa 1 mẩu tin
- Tùy vào cách tổ chức tập tin mà có cách thao tác phù hợp.

Mẩu tin có chiều dài cố định

- Xét 1 tập tin gồm các mẩu tin *account*

```
type deposit = record
        account-number: char(10);
        branch-name: char(22);
        balance: real;
end
```

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
  - 1 mẩu tin *account* : 40 bytes
  - 40 bytes đầu tiên là mẩu tin thứ 1
  - 40 bytes tiếp theo là mẩu tin thứ 2…

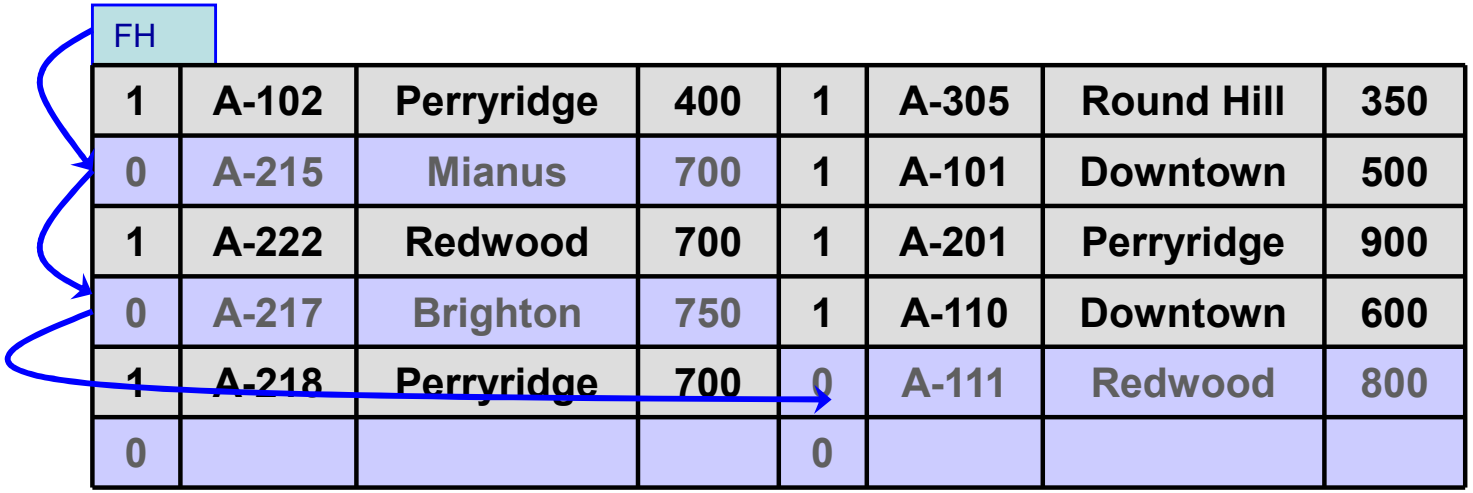
Mẩu tin có chiều dài cố định (tt)

- Mỗi mẩu tin có thêm 1 bit (tương tự .dbf)
  - =0: Xóa
  - =1: Đang dùng
- File header chứa địa chỉ của mẩu tin trống đầu tiên
  - Danh sách các mẩu tin trống (free list)
- Lưu trữ vật lý của các mẩu tin *account*

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			

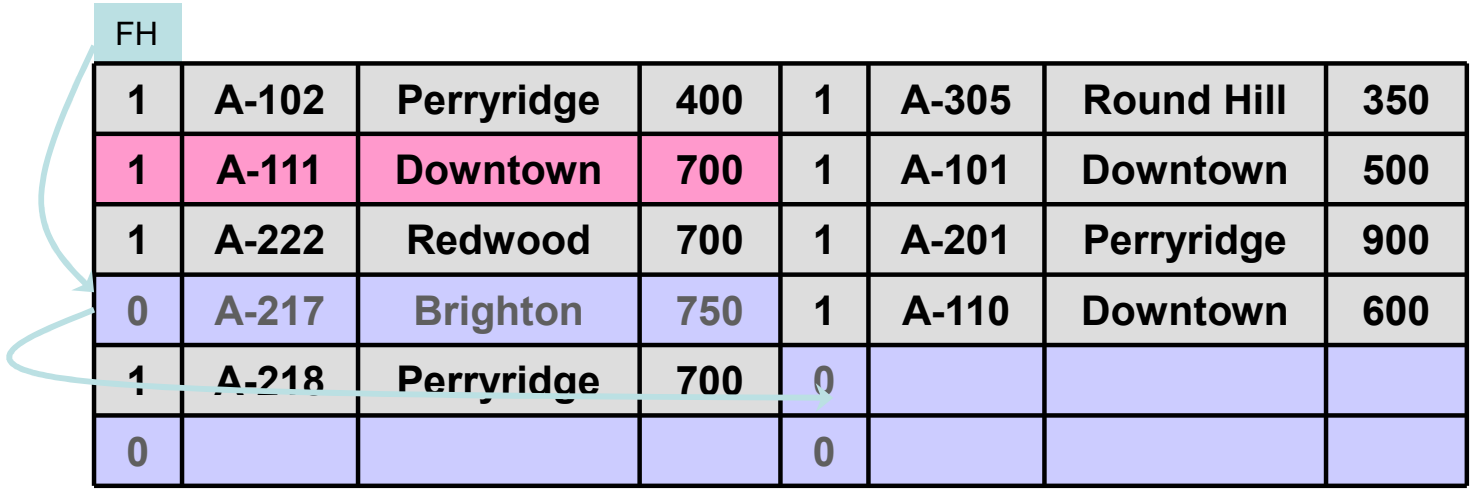
Mẩu tin có chiều dài cố định (tt)

- Hủy mẩu tin
  - Đánh dấu xóa vào bit thông tin
  - Đưa mẩu tin bị đánh dấu xóa vào free list



Mẩu tin có chiều dài cố định (tt)

- Thêm một mẩu tin
  - Hoặc thêm vào những mẩu tin bị đánh dấu xóa hoặc thêm vào cuối tập tin
  - Cập nhật lại free list



- Tìm kiếm
  - Quét tuần tự trên tập tin

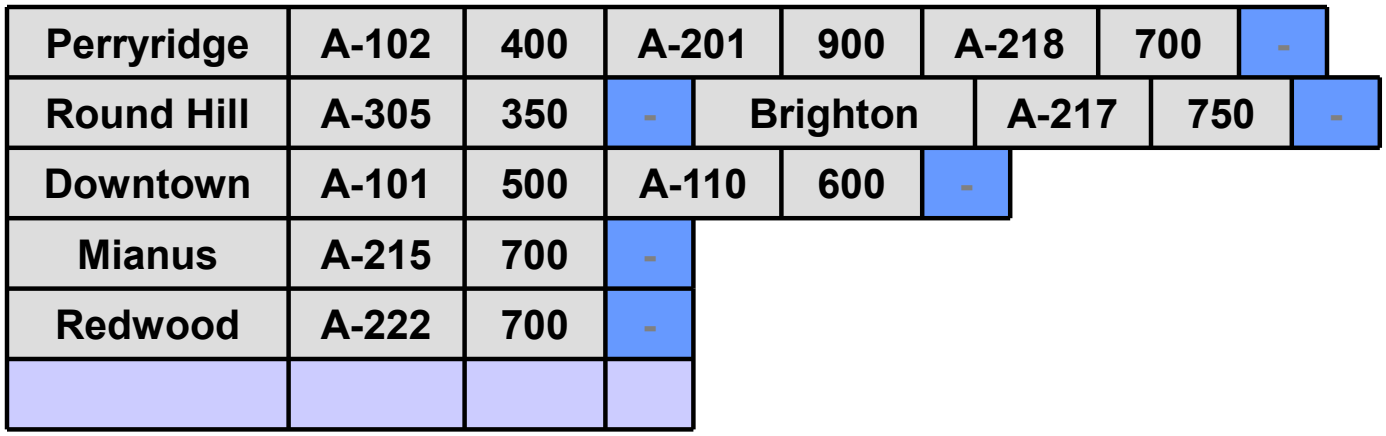
Mẩu tin có chiều dài động

- Trong DBMS, mẩu tin có chiều dài động dùng để
  - Lưu trữ nhiều loại mẩu tin trong 1 tập tin
  - Các loại mẩu tin chứa các trường có chiều dài động
- Xét tập tin gồm các mẩu tin account

```
type account-list = record
        branch-name: char(22);
        account-info: array [1..n] of
                record
                        account-number: char(10);
                        balance: real;
                end
end
```

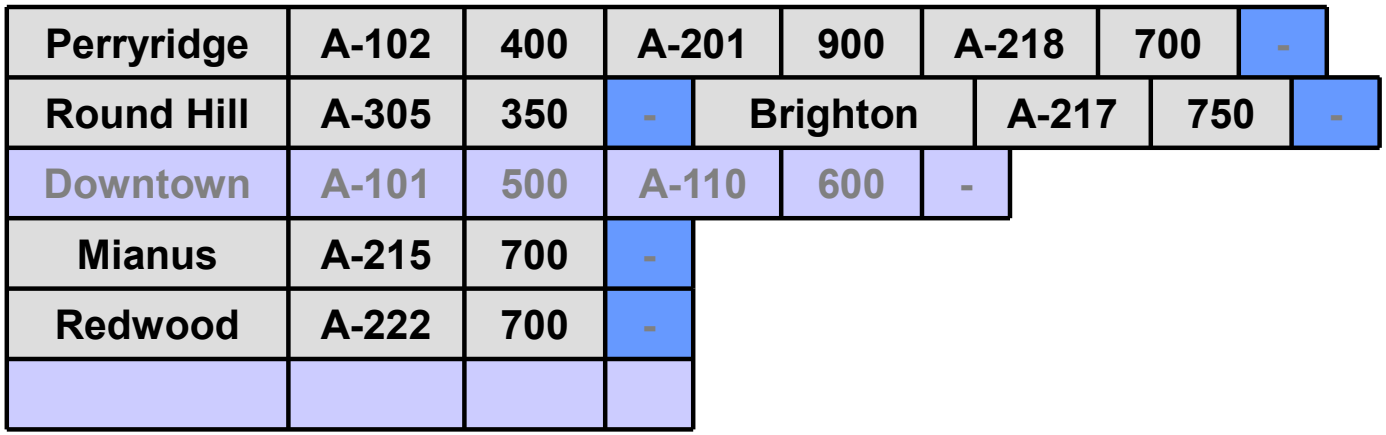
Mẩu tin có chiều dài động (tt)

- Byte-String Representation
  - Cuối mỗi mẩu tin có 1 byte ký tự đặc biệt cho biết kết thúc mẩu tin



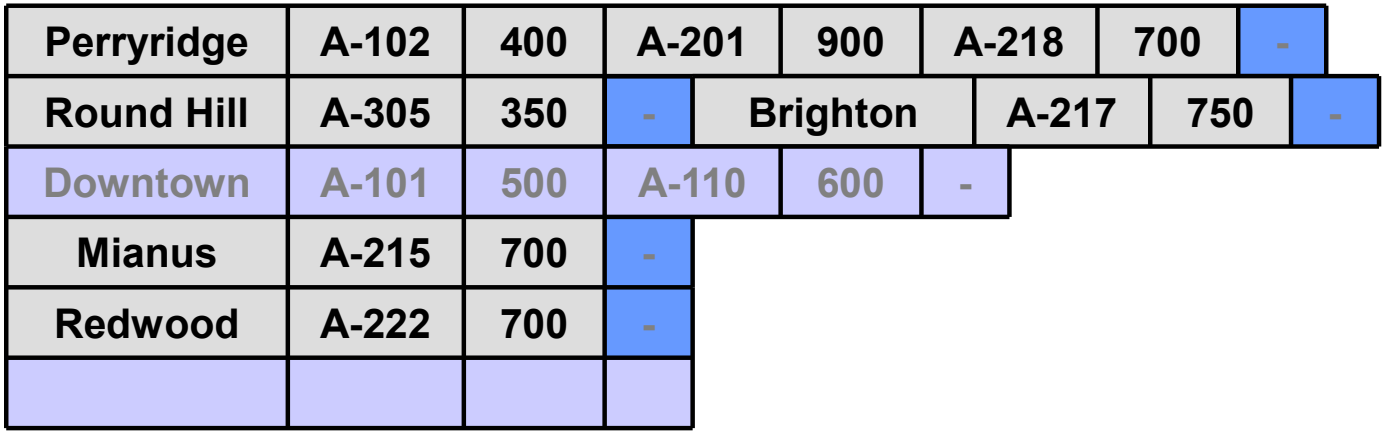
Mẩu tin có chiều dài động (tt)

- Byte-String Representation
  - Sử dụng lại không gian trống sau khi xóa 1 mẩu tin không hiệu quả
  - Dẫn đến tình trạng phân mảnh



Mẩu tin có chiều dài động (tt)

- Byte-String Representation
  - Tốn nhiều chi phí khi chiều dài mẩu tin thay đổi



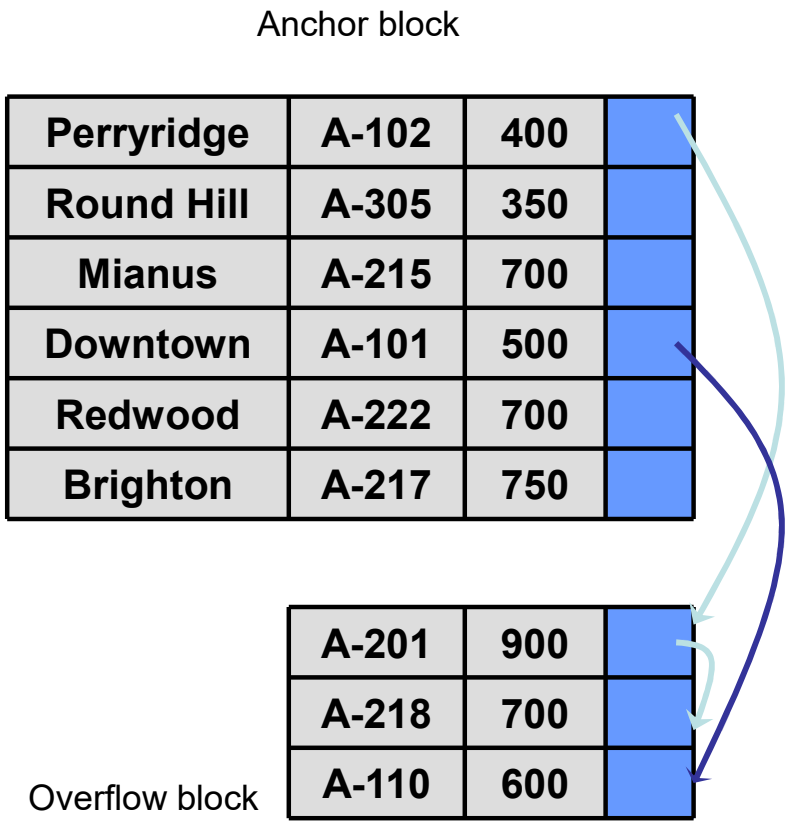
Mẩu tin có chiều dài động (tt)

- Fixed-Length Representation
  - Sử dụng 1 hay nhiều mẩu tin có chiều dài cố định biểu diễn cho những mẩu tin có chiều dài động
  - Có 2 kỹ thuật
    - Reserved space
      - Sử dùng độ dài lớn nhất của 1 mẩu tin nào đó cài đặt cho tất cả các mẩu tin còn lại
      - Độ dài này phải đảm bảo không bao giờ dài thêm được nữa

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				

Mẩu tin có chiều dài động (tt)

- Fixed-Length Representation
  - Pointer
    - Các mẩu tin có chiều dài động móc xích với nhau thông qua danh sách các mẩu tin có chiều dài cố định
    - Có 2 loại blocks trong tập tin
      - » Anchor block – Chứa mẩu tin đầu tiên của mảng *account-info*
      - » Overflow block – Chứa các mẩu tin tiếp theo của mảng *account-info*



Cách tổ chức mẩu tin trên file

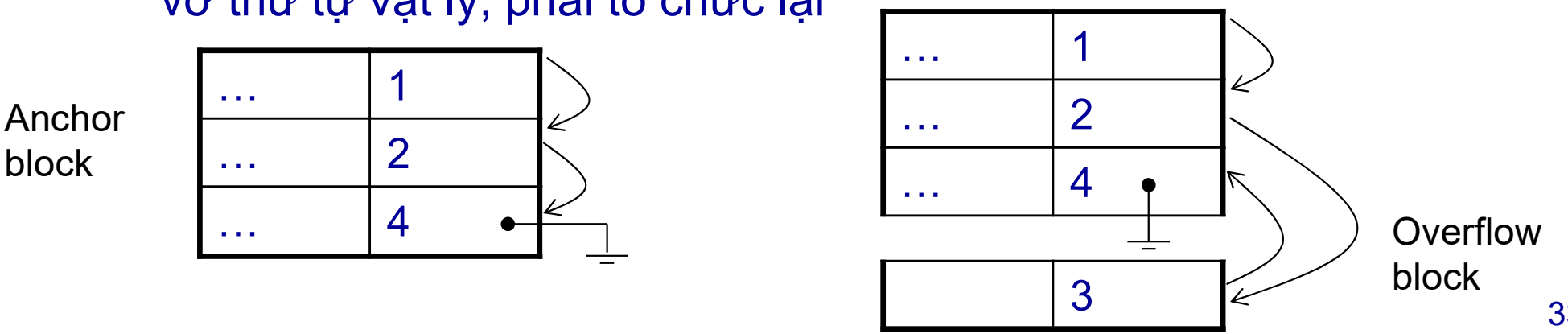
- Không được sắp: mẩu tin được chèn vào bất cứ chỗ nào trống trên file.
- Được sắp: mẩu tin lưu vào đúng vị trí để đảm bảo thứ tự theo một trường nào đó.
- Băm (Hashing): định vị mẩu tin trên thiết bị lưu trữ dùng một hàm băm.

Heap file

- Là hình thức lưu trữ dữ liệu trong đó các record được lưu *không* theo thứ tự logic nào cả (mà là thứ tự thêm dữ liệu).
- Thường thì dữ liệu của mỗi quan hệ được lưu trong 1 file.
  - Tìm kiếm: duyệt.
  - Thêm: nhanh.
- Cách thức lưu trữ và thao tác dữ liệu dễ, chỉ thích hợp cho tập tin có kích thước nhỏ, sẽ rất chậm khi tập tin có kích thước lớn.

Sequential file

- Là hình thức lưu trữ dữ liệu trong đó các mẩu tin được lưu trữ *theo thứ tự* của trường là *search key*.
- Link các mẩu tin quan hệ thứ tự bằng pointer.
- Thích hợp cho những ứng dụng đặc trưng làm việc trên dữ liệu được sắp xếp (theo search key).
  - Tìm kiếm: duyệt hoặc tìm tuần tự.
- Nên lưu trữ vật lý theo thứ tự của search key để giảm thiểu số block cần phải truy cập. Nhưng:
  - Khi dữ liệu lớn, thao tác Insert, Delete phức tạp
    - Insert: Định vị -> insert vào overflow block (≠ anchor block)-> phá vỡ thứ tự vật lý, phải tổ chức lại



Hashing file

- Một hàm băm (hash function) được thiết lập trên 1 thuộc tính là search key của quan hệ.
- Nguyên lý: “lưu ở đâu, tìm ở đó”.
- Chia tập tin thành các lô (bucket) tùy giá trị của search key. Mỗi lô có một số block, link nhau bởi pointer. Dữ liệu trong block được tổ chức như heap.
- B là số lượng các lô. Giá trị hàm băm tại giá trị tìm kiếm là số nguyên $\in$ [0,B-1] cho biết lô chứa mẩu tin.Nếu khóa là chuỗi ký tự, ta định ra nguyên tắc chuyển chuỗi ký tự thành số.

Hashing file

- Tìm kiếm mẩu tin khóa v
  - Tính h(v) để biết lô, và thực hiện tìm kiếm trong lô này.
- Chèn
  - Tính h(v) để biết lô. Tìm kiếm khối cuối cùng của lô, nếu còn chỗ thì chèn, còn không thì cấp phát 1 khối khác chèn vào cuối danh sách của lô h(v).
- Xóa/ Sửa
  - Tìm kiếm và sửa hoặc xóa (đánh dấu)
  - Sau khi xóa, có thể phải thực hiện bước hiệu chỉnh (dồn dữ liệu trong khối) để giảm số lượng khối trong lô này.

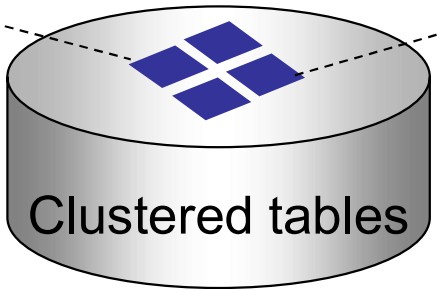
Clustering file

10	TENPB	ĐĐ
	KT	HCM

MANV	TENNV	…
N2	B	…
N5	E	…
N6	G	…

20	TENPB	ĐĐ
	KT	HCM

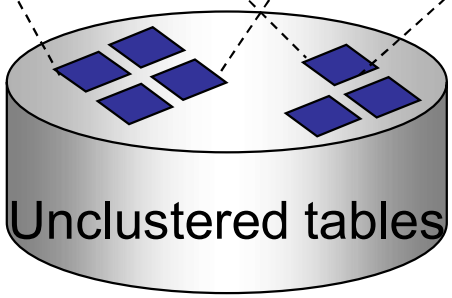
MANV	TENNV	…
N1	A	…
N3	C	…
N4	D	…



DL liên quan được lưu cùng nhau, tiết kiệm không gian

MANV	TENNV	MAPB	…
N1	A	20	…
N2	B	10	…
N3	C	20	…
N4	D	20	…
N5	E	10	…
N6	G	10	…

MAPB	TENPB	ĐĐ
10	KT	HCM
20	KD	HN



DL liên quan tách biệt, tốn không gian

Clustering file

- 1 cluster được hình thành từ việc lưu dữ liệu của một vài table chung trên một vài block.
- Cluster key là 1 hoặc nhiều field chung của các table tham gia việc gom nhóm. Cluster key được chỉ định khi người dùng tạo cluster.
- Các table này thường được dùng chung hoặc kết (join operator $\triangleright\triangleleft$) để phục vụ cho nhu cầu truy xuất dữ liệu.
- Việc lưu trữ này có ích:
  - Giảm thời gian truy xuất đĩa vì số block phải đọc giảm
  - Giá trị tại field là cluster key chỉ được lưu 1 lần, bất kể có bao nhiêu record ở table khác tham chiếu đến dòng này $\Rightarrow$ tiết kiệm không gian để lưu trữ (và tạo mối quan hệ trên dữ liệu)
- Tổ chức dl theo kiểu cluster không ảnh hưởng gì đến việc tạo index trên các table tham gia tạo cluster.

Sử dụng cluster file

- Chỉ định cluster ở giai đoạn thiết kế vật lý.
- Chọn các table để gom nhóm:
  - Các table chủ yếu phục vụ cho truy vấn (select), ít khi thêm mới (insert) hoặc cập nhật (update).
  - Chứa dữ liệu được truy vấn chung hoặc kết với tần suất cao.
- Chọn các field làm cluster key.
  - Cluster key phải có đủ các giá trị phân biệt để các record liên quan đến mỗi giá trị của cluster key lấp gần đầy 1 block dữ liệu.
    - Nếu có ít dòng quá sẽ làm tốn không gian lưu trữ mà hiệu quả không đáng kể.
    - Có thể định SIZE khi tạo cluster, là số byte trung bình ước tính để có thể lưu 1 cluster
    - Nếu có nhiều dòng quá cũng không hiệu quả.
    - Dùng cluster key có quá ít giá trị, vd: PHAI, sẽ phản tác dụng.

Index (1)

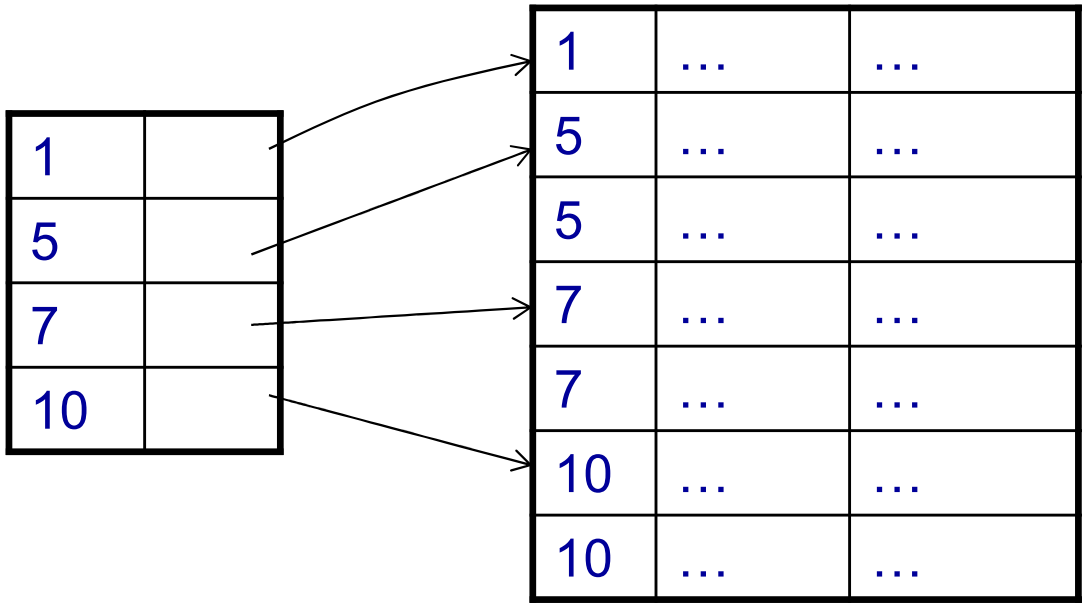
- Dùng chỉ mục cho file giống như việc dùng bảng liệt kê danh mục (catalog) trong một thư viện.
  - Thông tin trên catalog đã được sắp xếp $\Rightarrow$ tìm kiếm nhanh mà không phải duyệt tất cả.
- Về kỹ thuật, có 2 lọai index cơ bản:
  - Index sắp thứ tự (ordered indices) dựa trên các giá trị làm index.
    - Dùng PP tìm nhị phân trên file index.
    - Index là 1 file có thứ tự gồm các mẩu tin có chiều dài cố định gồm 2 field.
      - Field 1: khóa tìm kiếm.
      - Field 2: con trỏ trỏ đến các block.
  - Index dùng kỹ thuật băm (hash indices).
- Đánh giá các kỹ thuật dùng index.
  - Lọai truy xuất.
  - Thời gian truy xuất (access time).
  - Thời gian Insert / delete.
  - Không gian đĩa dùng cho index.

Index (2)

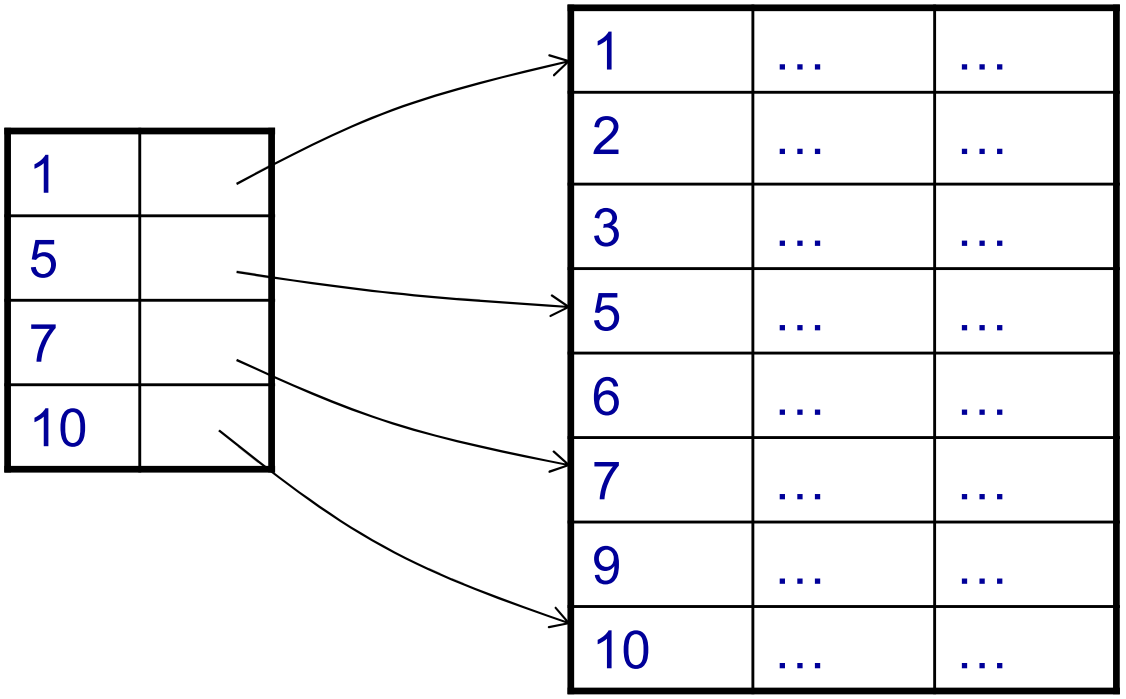
- Dense / Sparse index
  - Dense index
    - File dữ liệu có bao nhiêu giá trị trên search key thì trên file index có bấy nhiêu record
    - Mỗi record của file index chứa 1 giá trị là search key và 1 con trỏ trỏ đến record đầu tiên trên file dữ liệu có cùng giá trị trên trường search key.
  - Sparse index
    - Các record trên tập tin index chỉ ứng với một số giá trị trên file dữ liệu trên trường search key (chứ không phải tất cả các giá trị của search key như dense index)
    - Để tìm 1 giá trị, ta tìm trong tập tin index 1 mẩu tin sao cho giá trị search key lớn nhất <= giá trị cần tìm, và duyệt record xuất phát từ vị trí đầu tiên mà pointer chỉ đến.

Ví dụ

- Dense index



- Sparse index



Index (3)

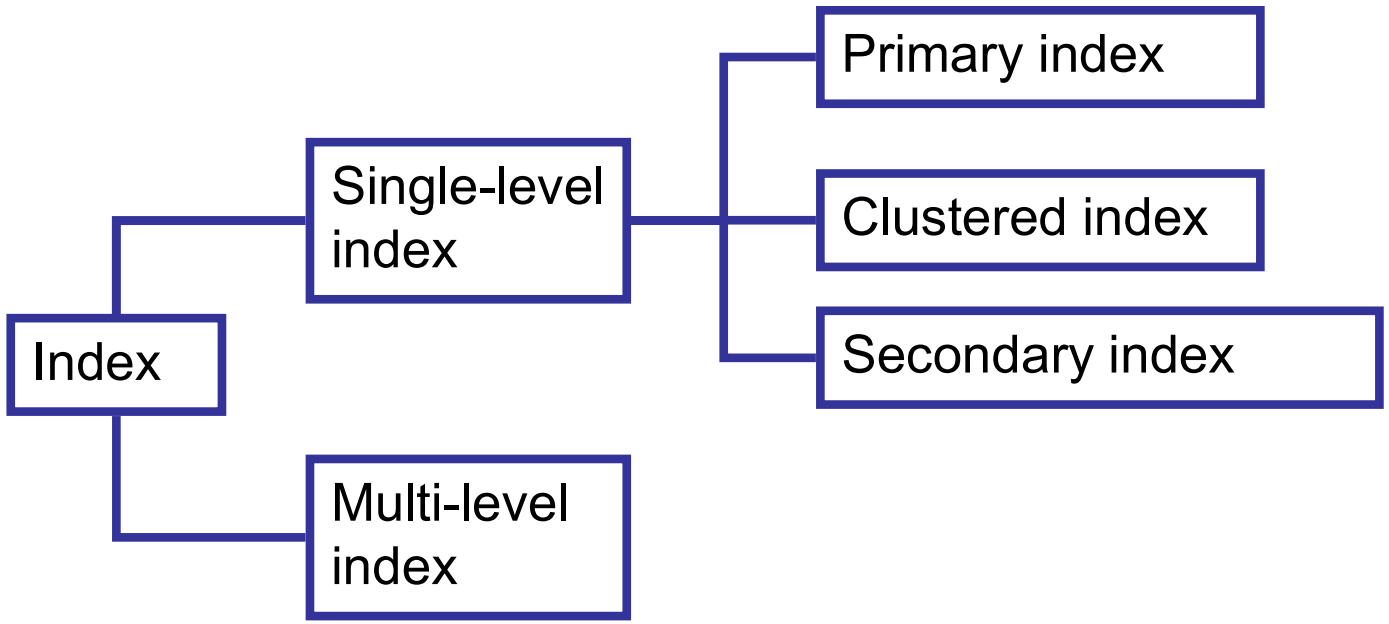
- File có cách tổ chức riêng và có cách thao tác trên file tương ứng.
- Bên cạnh đó, ta cần bổ sung thêm cấu trúc truy cập hỗ trợ truy cập nhanh trên file.
- Index là cơ chế giúp HQT CSDL truy xuất dữ liệu nhanh.
- Index key dùng để chỉ 1 hoặc nhiều trường (field) dùng làm index.
  - Simple Index: index key chỉ có 1 field duy nhất.
  - Composite index: index key có nhiều 1 field, nhưng <=16
  - Ví dụ:
- NOIGD (…,TENNGD, SONHA, DUONG, QH,TP)
  - Nhu cầu tìm nhanh 1 địa chỉ giao dịch.
  - Nếu index key là DUONG (simple index) thì không hiệu quả
  - Mà phải dùng SONHA, DUONG, QH, TP (composite index), và trên các field này, giá trị là duy nhất

Index(4)

- Mỗi cấu trúc index kết hợp với 1 index key cụ thể.
- Truy cập đến CSDL dùng index, ta phải sử dụng 1 hoặc một số field là index key trong mệnh đề WHERE của câu SQL.
  - Nếu là composite index thì nên dùng nhiều hơn 1 field trong mệnh đề WHERE, khi đó truy xuất sẽ hiệu quả hơn.
- Cơ bản, bất cứ field nào cũng có thể là index và có thể có nhiều index trên cùng 1 file. Vấn đề là index có mang lại hiệu quả hay không.
- Index có hiệu quả hay không căn cứ vào:
  - Lọai dữ liệu mà trên đó thiết lập index.
  - Giá trị trên index key có phân biệt hay không.
  - Lọai câu SQL được dùng.
    - Khi thi hành 1 câu SQL nếu nhiều hơn 20% các dòng dữ liệu trong 1 bảng được truy cập đến thì việc dùng index không có lợi bằng việc không dùng index (table scan).
  - Các truy cập khác trên bảng, nếu cập nhật nhiều trên field làm index sẽ làm chậm hệ thống.
  - Có quá nhiều index sẽ làm chậm hệ thống.

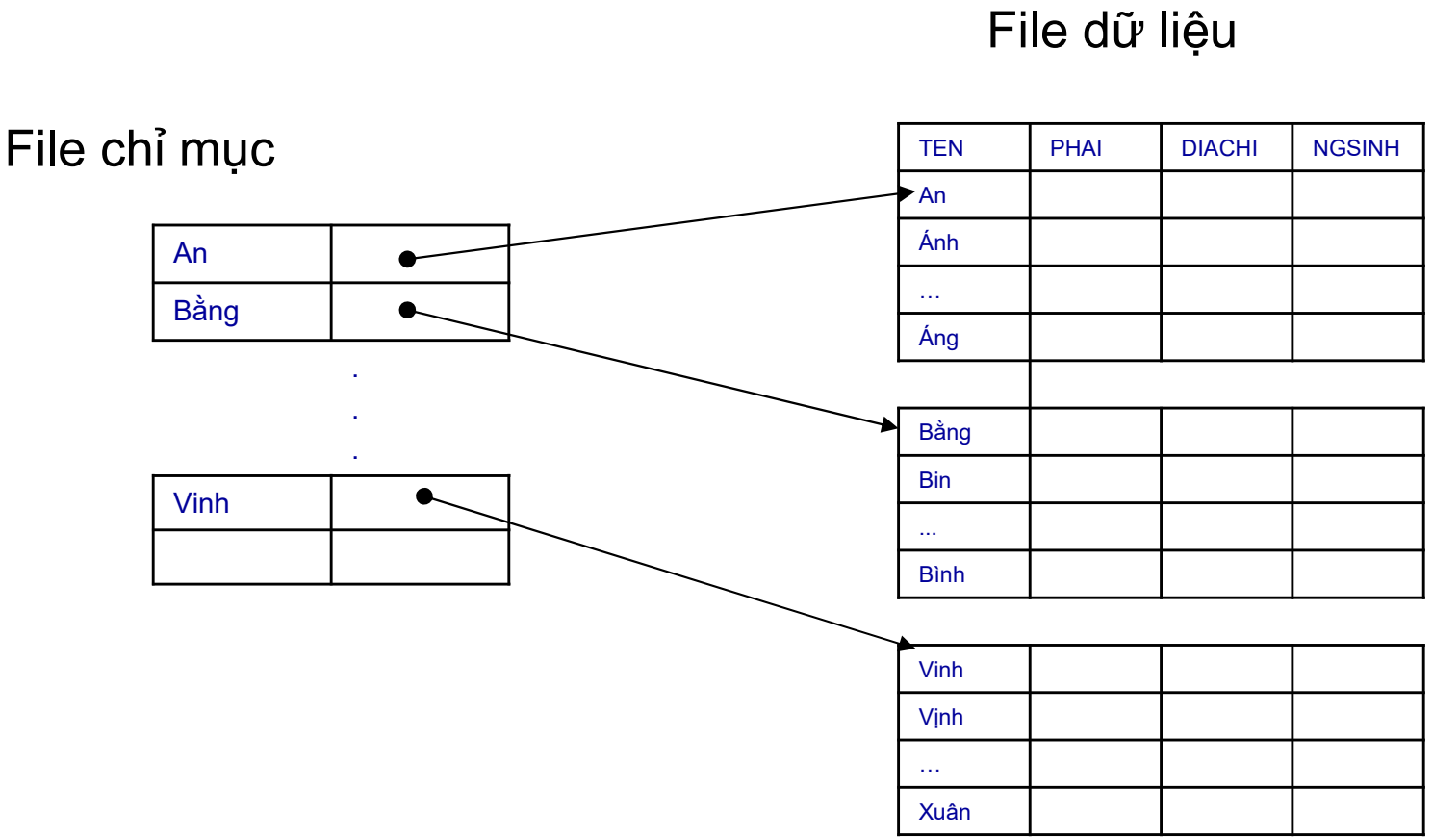
Index (5)

Các thuật ngữ:



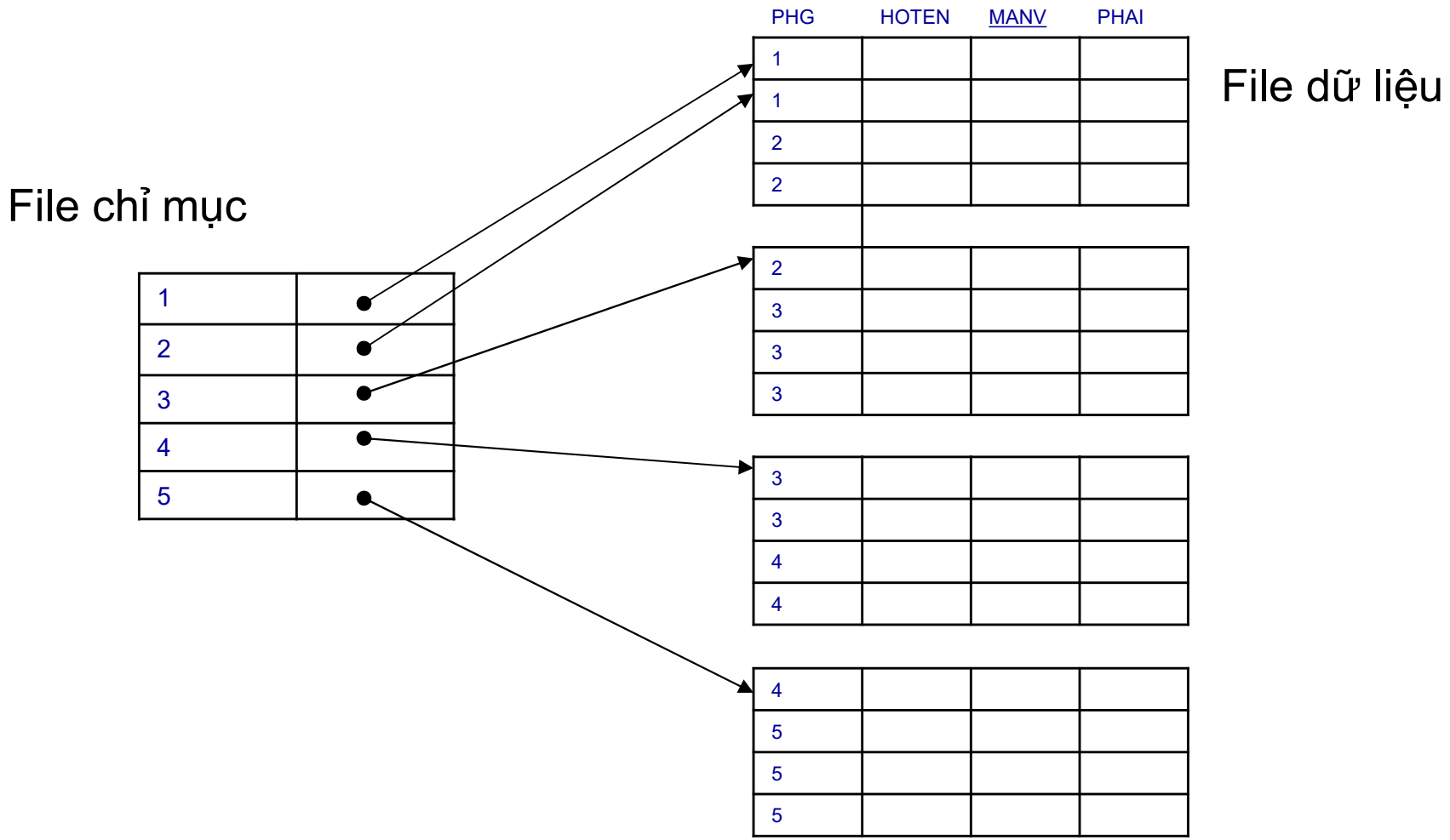
Primary index

- Được tạo trên field làm khóa sắp xếp cho file dữ liệu. Thứ tự vật lý của các record trên đĩa cũng dựa trên field này, và trên đó các record có giá trị duy nhất.
  - Nếu có nhiều record có cùng giá trị trên field dùng để sắp xếp, ta sẽ tạo clustering index trên field này.
- Có 1 mẩu tin index trong file index ứng với 1 block trong file dữ liệu.
- File primary index có kích thước nhỏ hơn rất nhiều so với file dữ liệu.
  - Record đầu tiên trong mỗi block của file dữ liệu gọi là **anchor record** hay **block anchor**.
- Chỉ có thể có 1 primary index, hoặc 1 clustering index trên 1 file dữ liệu, không thể có cả hai loại index này trên cùng 1 file.



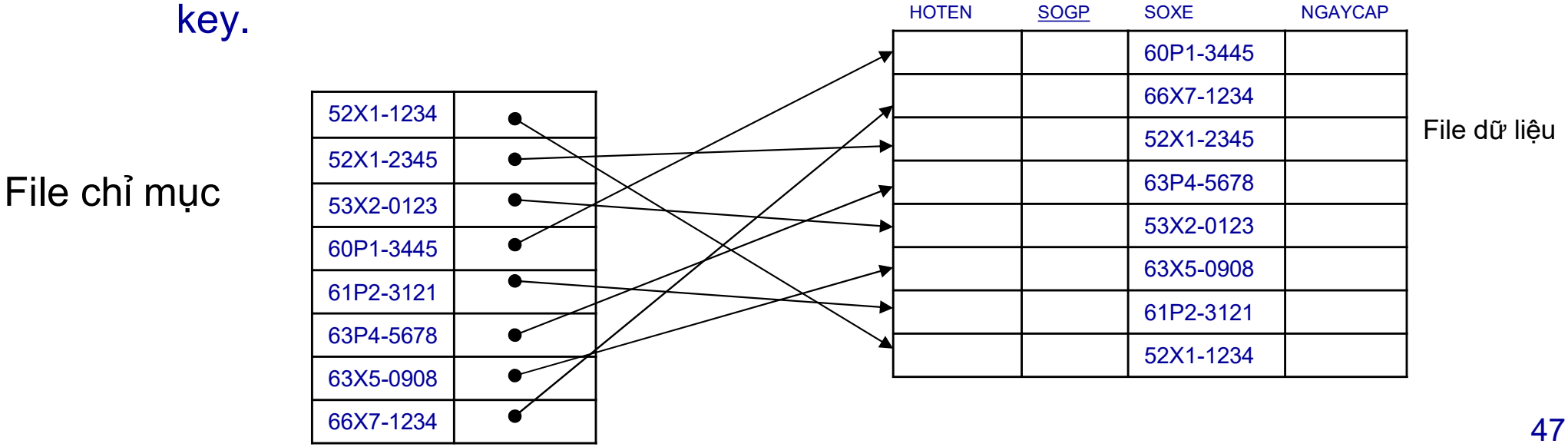
Clustering index

- Nếu dữ liệu trên data file được sắp vật lý theo field không phải là khóa (không duy nhất đối với từng mẩu tin) thì field đó là clustering field.
- Clustering index được tạo trên clustering field để tăng tốc độ tìm kiếm các mẩu tin có cùng giá trị trên clustering field.
- Có 1 record trên file index chứa 1 giá trị của clustering field, con trỏ trỏ đến block đầu tiên chứa giá trị phân biệt.

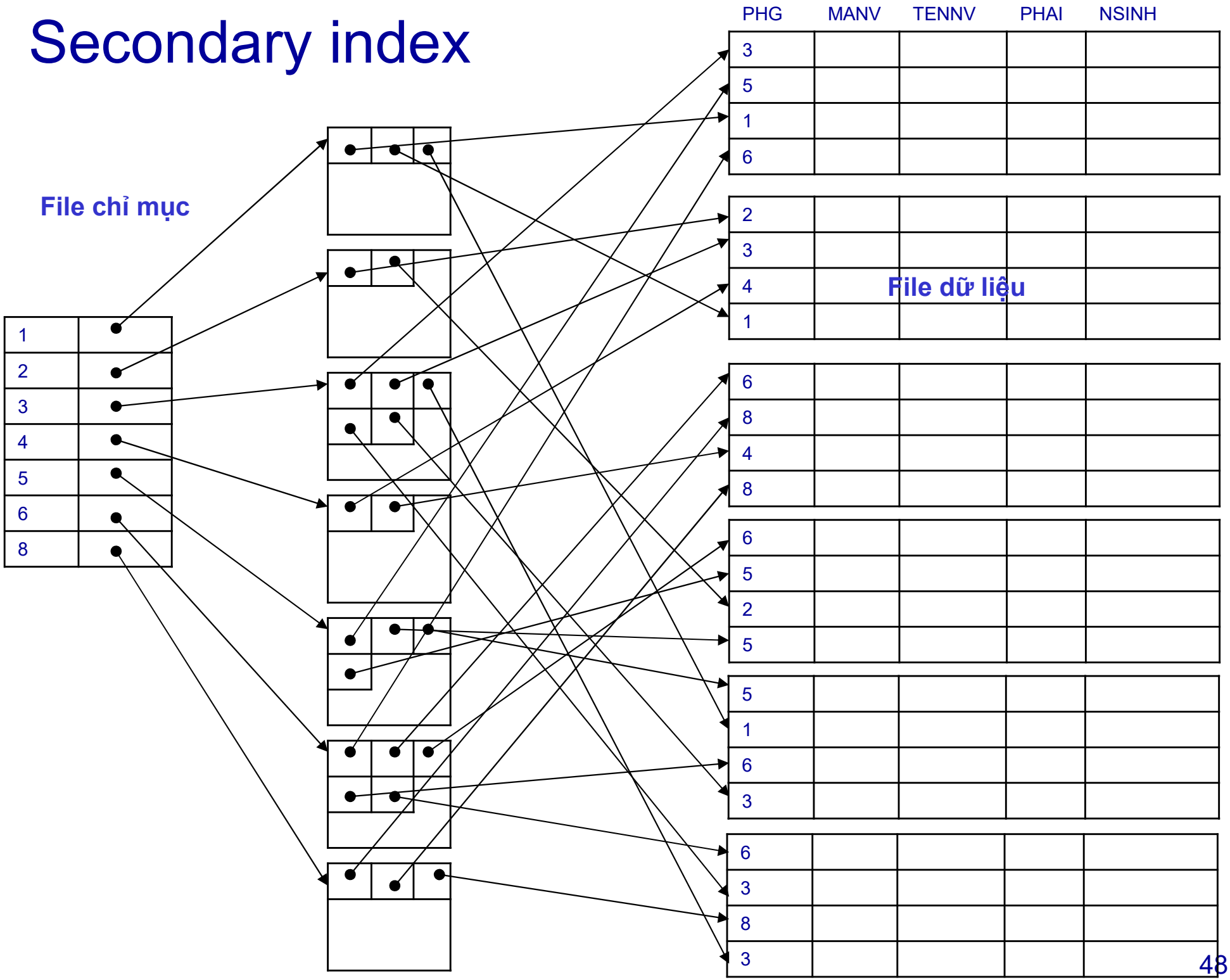


Secondary Index

- Một secondary index cung cấp thêm phương tiện để truy cập file, ngoài primary index ra.
  - Được tạo trên field là candidate key và có giá trị duy nhất trên mỗi record, dữ liệu của data file không được sắp thứ tự trên field này.
  - Cũng có thể tạo trên field không phải là khóa và có giá trị trùng nhau.
  - Field 1 là field dữ liệu không được sắp thứ tự của data file, và cần tìm kiếm trên đó.
  - Field 2 là con trỏ trỏ đến block đầu tiên chứa giá trị, hoặc trỏ đến record chứa giá trị.
- Có thể tạo nhiều secondary index cho 1 file dữ liệu.
  - Ví dụ: tạo trên field có giá trị duy nhất, field này còn được gọi là secondary key.



Secondary index

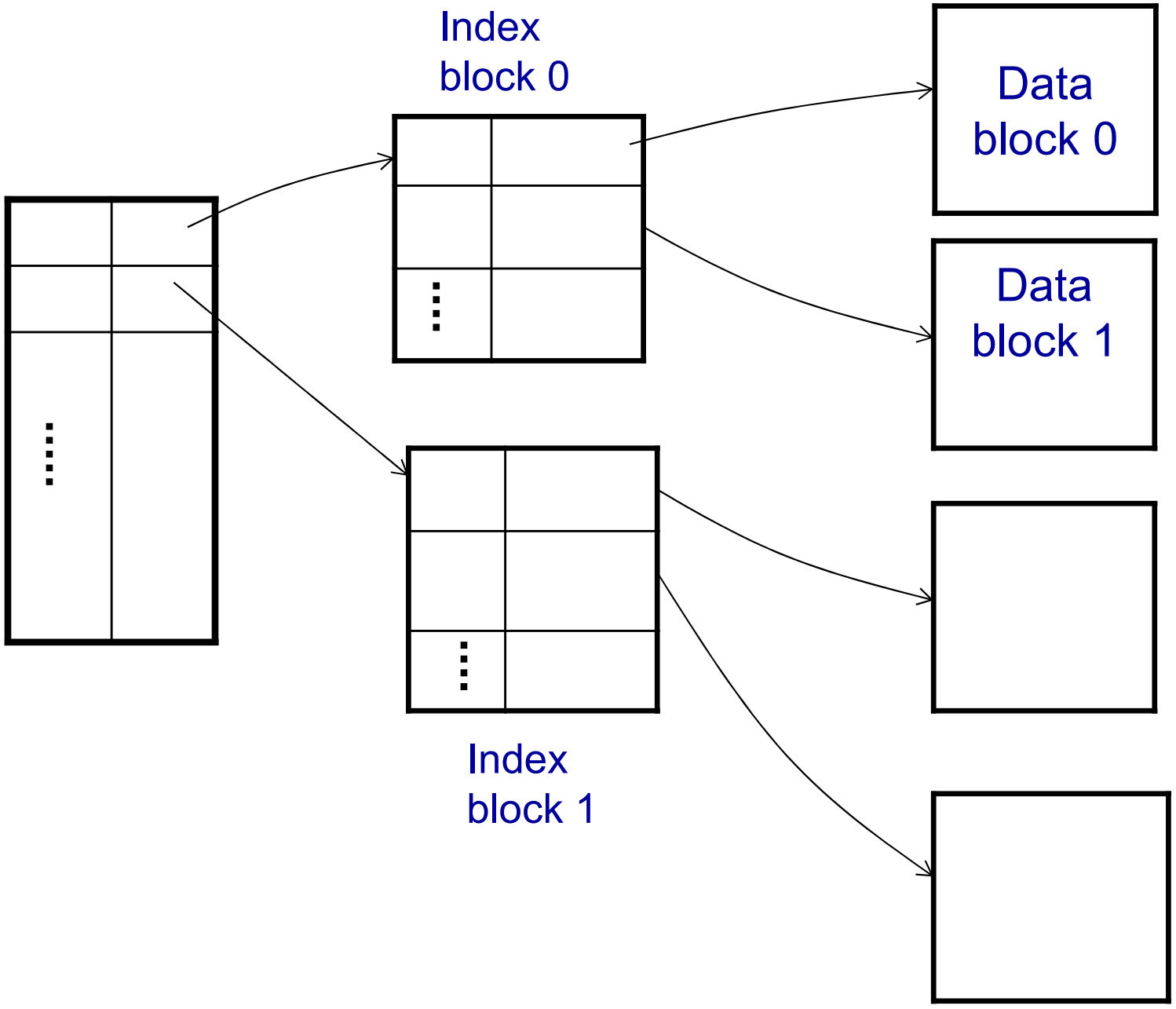


Nhận xét

	File dữ liệu sắp theo Index field	File dữ liệu không sắp theo index field
Index field làm khóa	Primary index	Secondary index (Key)
Index field không là khóa	Clustering index	Secondary index (non key)

- 1 file có thể có nhiều index vì có thể có nhiều nhu cầu tìm kiếm trên file.
- Trong SQL Server, có thể tạo tối đa là 1 primary index và 249 secondary index.

Multi – level Index



Dùng index

- Nên tạo index trên các field có giá trị phân biệt, được truy xuất đến với tần suất cao, và kết quả câu truy vấn là nhiều dòng dữ liệu.
- Truy vấn trên một miền giá trị dùng các toán tử BETWEEN, >, >=, <, <=.
- Index key là các trường sẽ dùng cho phép kết, hoặc trong mệnh đề GROUP BY, ORDER BY.
- Không nên tạo clustered index trên các trường sẽ thường xuyên cập nhật.

Hết chương 4.