

Lưu trữ dữ liệu

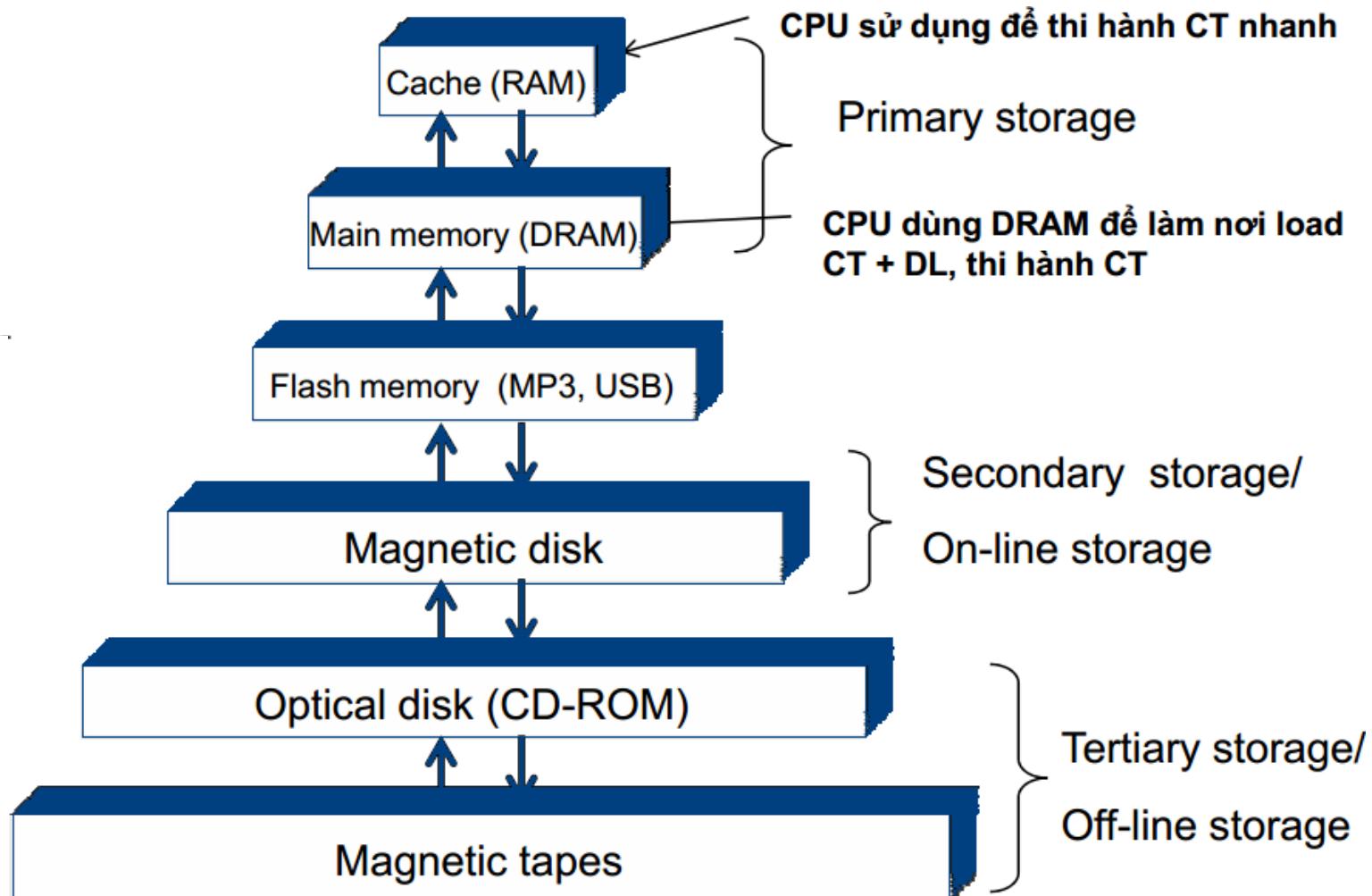
Mục đích

- 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.

Lưu trữ dữ liệu

Phương tiện lưu trữ

Các phương tiện lưu trữ dữ liệu



Phân cấp lưu trữ

- ❑ Primary Storage - Bộ nhớ chính
 - Dữ liệu hiện hành
- ❑ Secondary Storage - Đĩa
 - CSDL chính thức

Primary storage

- ❑ Là dạng lưu trữ mà CPU có thể thao tác trực tiếp được
 - VD: bộ nhớ chính của máy tính, bộ nhớ sử dụng cho cache.
- ❑ Có tốc độ truy cập nhanh, nhưng có giới hạn về khả năng lưu trữ và giá thành cao

Primary storage

Các dạng Primary storage:

- **Static RAM (Random Access Memory)**: cho phép đọc ghi (các dữ liệu bị thay đổi hay đang sử dụng)
 - Dữ liệu trên RAM sẽ mất khi mất điện
- **Cache memory**: chính là RAM nhưng lưu trữ dữ liệu của những lần đọc trước đó
 - Khi chương trình cần đọc dữ liệu thì có thể đọc trong cache, làm cho việc thực thi chương trình sẽ nhanh
- **Dynamic RAM**: là vùng làm việc chính cho CPU (main memory), lưu trữ các chương trình và dữ liệu

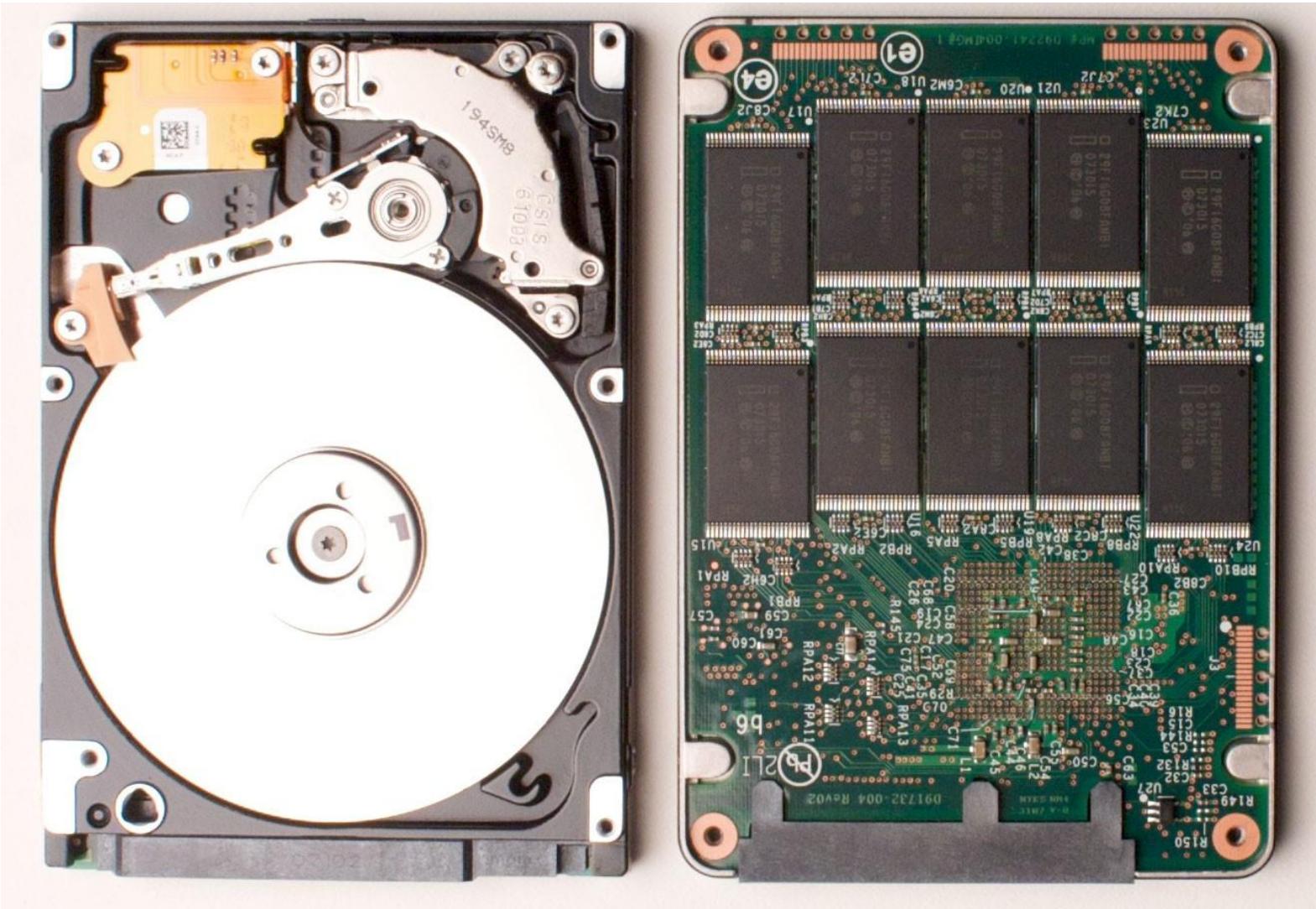
Secondary storage

- ❑ Là dạng lưu trữ mà CPU không thể thao tác trực tiếp được, dữ liệu phải được chuyển vào primary storage trước khi thao tác
- ❑ Secondary storage có tốc độ truy cập chậm hơn so với primary storage, nhưng khả năng lưu trữ cao hơn và giá thành thấp hơn

Secondary storage

- ❑ Các dạng secondary storage (lưu CSDL):
 - **SSD** (Solid-State Drive)
 - **HDD** (Hard Disk Drive)
- ❑ Cơ sở dữ liệu được lưu trữ trên đĩa, khi cần truy xuất dữ liệu phải chuyển từ đĩa vào bộ nhớ chính
- ❑ Các dạng storage khác (backup dữ liệu):
 - Đĩa quang (Optical Disk)
 - Băng từ (Magnetic Tape)

Đĩa cứng



HDD

SSD

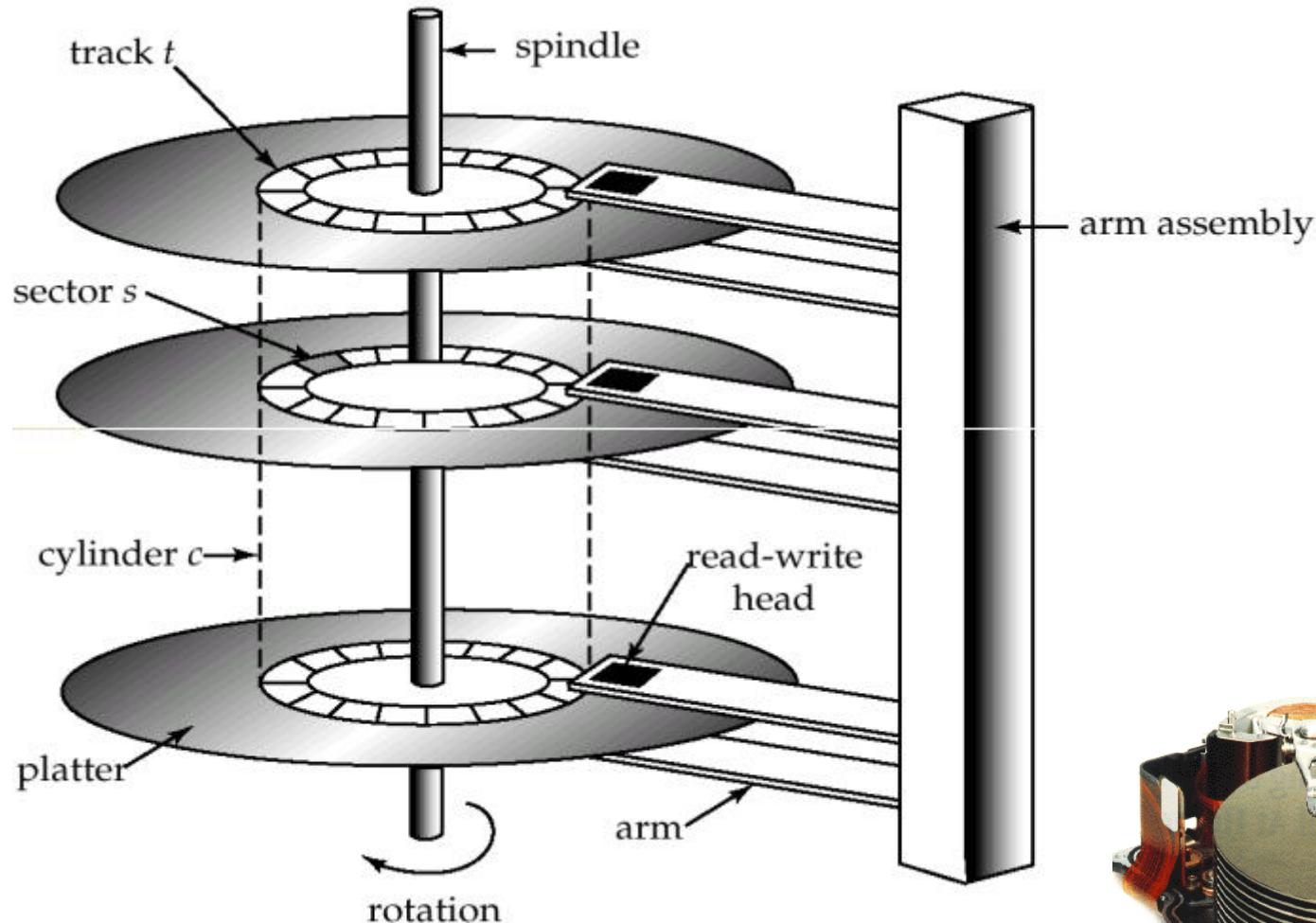
Đĩa từ (Magnetic disk/HDD)

Dùng đĩa từ để lưu CSDL vì

- Chi phí thấp
- Khối lượng lưu trữ lớn
- Lưu trữ lâu dài, phục vụ cho truy cập và xử lý
lặp lại



Đĩa từ (Magnetic disk)

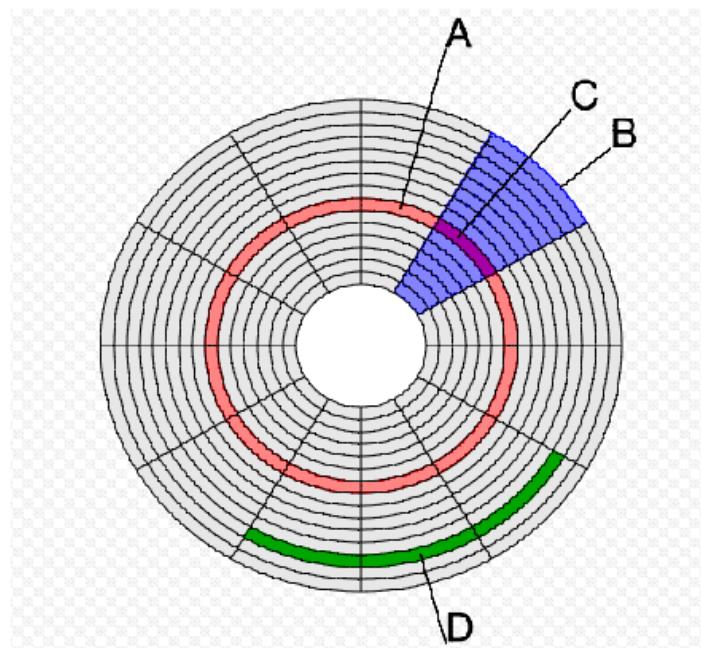


Đĩa từ (Magnetic disk)

❑ Định dạng mặt đĩa

- 1 mặt đĩa chia nhiều track
- 1 track chia thành nhiều block (page)
- 1 cluster gồm nhiều block

A : Track
B : Geometrical Sector
C: Track Sector
D: Cluster



Đĩa cứng

- Dữ liệu trên đĩa phải được chép vào bộ nhớ chính khi cần xử lý. Nếu dữ liệu có thay đổi thì sẽ được ghi trở lại vào đĩa.
- Bộ điều khiển đĩa (disk controller): giao tiếp giữa ổ đĩa và máy tính
 - nhận lệnh I/O → định vị đầu đọc → thực hiện R/W
- Block là đơn vị để lưu trữ và chuyển dữ liệu.
- Khi truy xuất các block liên tiếp thì tiết kiệm được thời gian → một số kỹ thuật tìm kiếm khai thác điều này

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.

Mẫu tin

- ❑ Mẫu tin (Record) là tập hợp dữ liệu có liên quan với nhau
 - Mỗi mẫu tin gồm nhiều trường
 - Mỗi trường có kiểu dữ liệu riêng
- ❑ Có 2 loại mẫu tin
 - Mẫu tin có chiều dài cố định
 - Mẫu tin có chiều dài thay đổi

Mẫu tin có chiều dài cố định

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

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

- Giả sử

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

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

Mẫu tin có chiều dài cố định

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

Mẫu tin có chiều dài cố định

❑ Hủy mẫu tin

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

FH							
1	A-102	Perryridge	400	1	A-305	Round Hill	350
0	A-215	Mianus	700	1	A-101	Downtown	500
1	A-222	Redwood	700	1	A-201	Perryridge	900
0	A-217	Brighton	750	1	A-110	Downtown	600
1	A-218	Perryridge	700	0	A-111	Redwood	800
0				0			

Mẫu tin có chiều dài cố định

- 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



FH							
1	A-102	Perryridge	400	1	A-305	Round Hill	350
1	A-111	Downtown	700	1	A-101	Downtown	500
1	A-222	Redwood	700	1	A-201	Perryridge	900
0	A-217	Brighton	750	1	A-110	Downtown	600
1	A-218	Perryridge	700	0			
0				0			

- Tìm kiếm
 - Quét tuần tự trên tập tin

Mẫu tin có chiều dài động

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

Mẫu tin có chiều dài động

❑ Byte-string Representation

- Cuối mỗi mẫu tin có 1 byte ký tự đặc biệt cho biết kết thúc mẫu tin

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

- Sử dụng lại không gian trống sau khi xóa mẫu tin không hiệu quả, dẫn đến tình trạng phân mảnh
- Tốn nhiều chi phí khi chiều dài mẫu tin thay đổi

Mẫu tin có chiều dài động

❑ Fixed-Length Representation

- 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
 - Reserved space
 - Pointer

Mẫu tin có chiều dài động

❑ Reserved space:

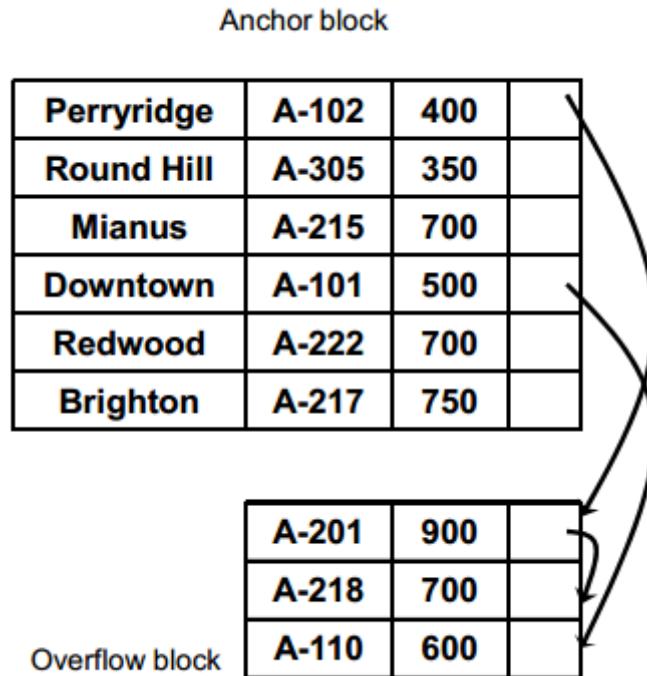
- Sử dụng độ dài lớn nhất của 1 mẫu tin nào đó cài đặt cho tất cả các mẫu tin còn lại.
- Độ dài này phải đảm bảo không bao giờ dài thêm được nữa.

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				

Mẫu tin có chiều dài động

❑ Pointer:

- Các mẫu tin có chiều dài động móc xích với nhau thông qua danh sách các mẫu tin có chiều dài cố định



Lưu tập tin trên đĩa

- CSDL được tổ chức trên đĩa thành một/nhiều tập tin, mỗi tập tin gồm nhiều mẫu tin
- Mẫu 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 tt chính (primary file organization) cho biết các mẫu tin định vị vật lý thế nào trên đĩa → cách truy cập
 - Cách tổ chức phụ (secondary organization / auxiliary access structure) → để truy cập các mẫu tin trên tt hiệu quả

Lưu tập tin trên đĩa

□ Tổ chức tập tin:

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

TH3 và TH4 không xảy ra khi lưu trữ dữ liệu trên CSDL quan hệ.

Lưu tập tin trên đĩa

- ❑ Mẫu tin có kích thước cố định: dễ truy xuất



- ❑ Mẫu 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.

001	Nguyen Van A	1000	LAB	
-----	--------------	------	-----	--

- TH2: Trước mỗi mẫu tin lưu kiểu mẫu 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 chẵn hạn, và lưu <field-type, field-value>

Lưu mẫu tin vào block

- Ta biết: đĩa được chia ra thành các **block**.
- Thường:
 - Kích thước block B > kích thước record R (cố định, tính bằng byte, $B \geq R$)
- 1 block có chứa nhiều mẫu tin, số lượng:
 - $bfr = \text{floor}(B/R)$ records/block
 - bfr gọi là blocking factor của file
 - $B - bfr * R$ là số byte không dùng trong 1 block.

Lưu mẫu tin vào block

Unspanned

block i	record 1	record 2	record 3	
---------	----------	----------	----------	--

block i+1	record 4	record 5	record 6	
-----------	----------	----------	----------	--

Spanned

block i	record 1	record 2	record 3	record 4	P
---------	----------	----------	----------	----------	---

block i+1	record 4 (phần còn lại)	record 5	record 6	P
-----------	-------------------------	----------	----------	---

Tổ chức 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 trả 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 mẫu tin có chiều dài cố định: Field-length, thứ tự field đổi với unspanned record.
 - Tập tin gồm mẫu tin có chiều dài thay đổi: mã kiểu cho field, ký tự ngăn cách, mã kiểu cho mẫu tin.
 - Tìm 1 mẫu tin trên đĩa:
 - 1 hoặc 1 số block được chép vào buffer.
 - CT trình tìm trên buffer dùng thông tin của file header.

Cách tổ chức mẫu tin trên tập tin

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

Cách tổ chức mẫu tin trên tập tin

- Heap file
- Sequential file
- Hashing file
- Clustering file

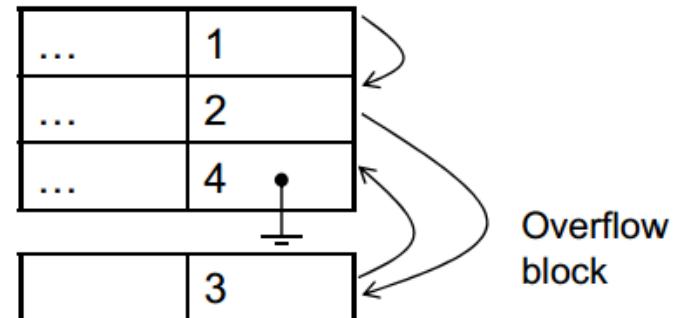
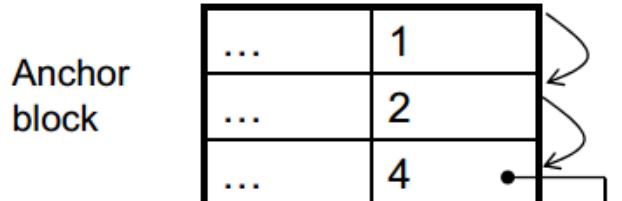
Heap file

- ❑ Là hình thức lưu trữ dữ liệu trong đó các mảnh tin 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: 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 theo thứ tự của trường là search key
- ❑ Liên kết các mẩu tin quan hệ thứ tự bằng con trỏ
- ❑ Thích hợp cho những ứng dụng đặc trưng làm việc trên dữ liệu được sắp xếp (theo search key)
 - tìm: duyệt hoặc tìm tuần tự
- ❑ Nên lưu trữ vật lý theo thứ tự của search key để giảm thiểu số block cần truy cập
- ❑ Khi dữ liệu lớn thì thao tác thêm, xóa phức tạp

Insert: Định vị -> insert vào overflow block (\neq anchor block) -> phá vỡ thứ tự vật lý, phải tổ chức lại



Tập tin tuần tự có độ dài mẫu tin cố định

- ❑ Các mẫu tin có độ dài cố định: l
- ❑ Xác định mẫu tin thứ n: $l * n$
- ❑ Ưu: định vị nhanh vị trí một mẫu tin
- ❑ Khuyết: khoảng trống trong lưu trữ

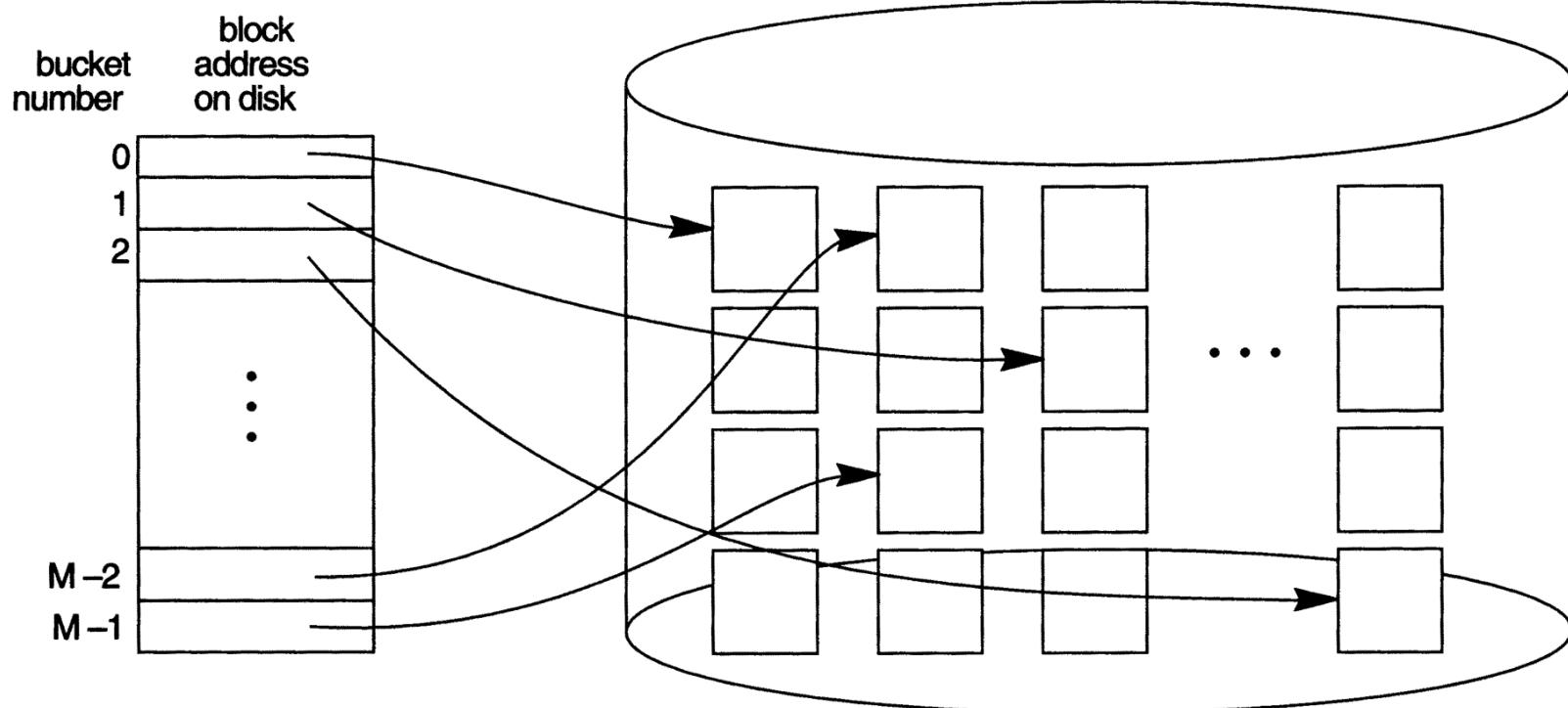
Tập tin tuần tự có độ dài mẫu tin thay đổi

- Các mẫu tin có độ dài khác nhau
- Đầu mỗi mẫu tin sẽ có một vùng đặc biệt lưu trữ độ dài của mẫu tin
- Ưu: tối ưu không gian lưu trữ
- Khuyết: muốn truy xuất mẫu tin thứ n phải duyệt qua $n-1$ mẫu tin trước đó

Hashing file

- ❑ Một hàm băm được thiết lập trên 1 thuộc tính là search key của quan hệ
- ❑ Chia tập tin thành các lô (bucket) tùy giá trị của search key. Mỗi lô có một số block, liên kết nhau bởi con trỏ. Dữ liệu trong block được tổ chức như heap.
- ❑ Nếu số lượng các lô là b , 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: định quy tắc chuyển chuỗi ký tự thành số

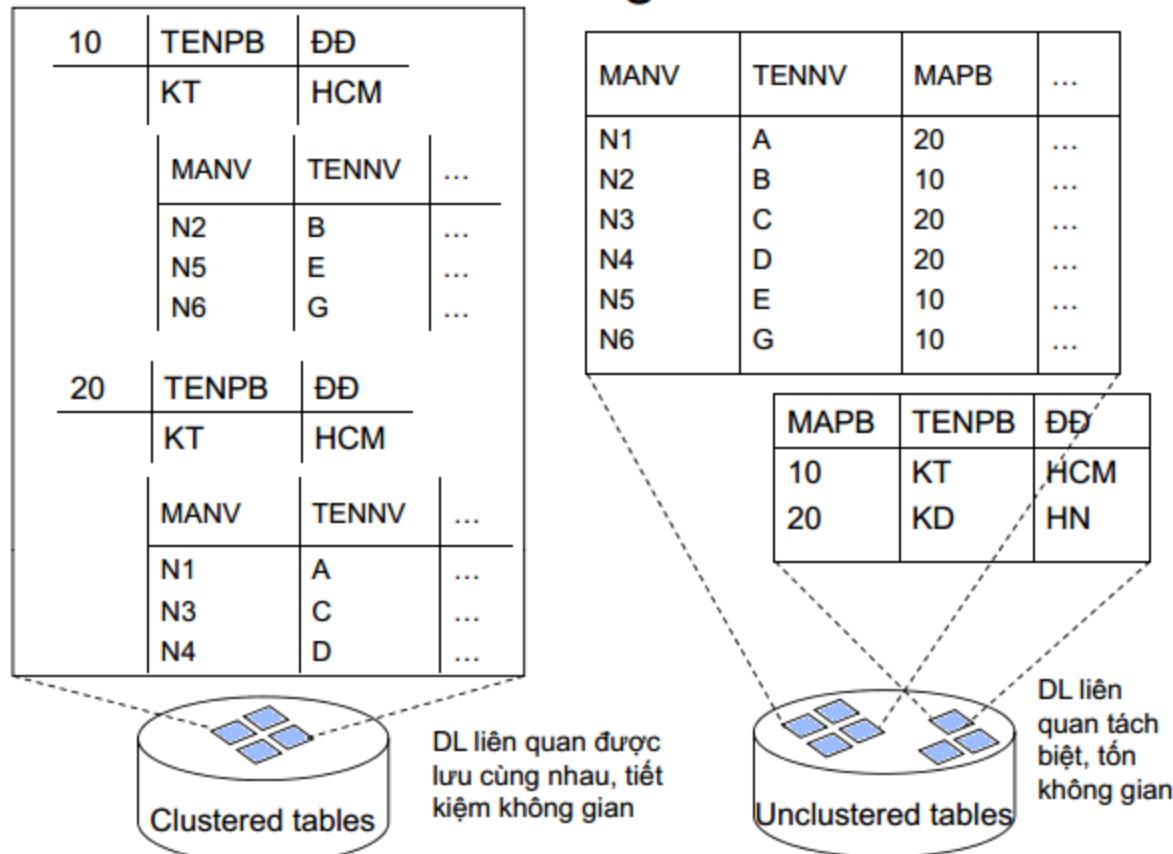
Hashing file



Hashing file

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

Clustering file



Clustering file

- ❑ Một cluster được hình thành từ việc lưu dữ liệu của một/nhiều bảng chung trên một vài block
- ❑ **Intra-clustering:** gom nhóm dữ liệu của cùng một bảng
 - Hình thành từ 1 phép chọn các bộ thỏa điều kiện nào đó để gom nhóm
- ❑ **Inter-clustering:** gom nhóm dữ liệu của nhiều bảng
 - Cluster key là 1 hoặc nhiều trường chung của các bảng tham gia việc gom nhóm
 - Các bảng này thường được dùng chung hoặc kết để phục vụ nhu cầu truy xuất dữ liệu

Clustering file

- ❑ Cách lưu trữ này có ích:
 - Giảm thời gian truy xuất đĩa vì số block phải đọc giảm
 - Tiết kiệm không gian lưu trữ: giá trị tại trường là cluster key chỉ được lưu 1 lần, bất kể có bao nhiêu mẫu tin ở bảng khác tham chiếu đến dòng này
- ❑ Tổ chức dữ liệu kiểu cluster không ảnh hưởng đến việc tạo chỉ mục (index) trên các bảng tham gia tạo cluster

Clustering file

Sử dụng cluster file

- Chỉ định cluster ở giai đoạn thiết kế vật lý
- Chọn các bảng 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 truy vấn chung hoặc kết với tần suất cao

Clustering file

Sử dụng cluster file

□ Chọn các trường làm cluster key:

- Phải có đủ giá trị phân biệt để các mẫu tin liên quan đến mỗi giá trị của cluster key lắp gần đầy 1 block dữ liệu
- Nếu có ít dòng: 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 quá nhiều dòng cũng không hiệu quả
- Dùng cluster key có quá ít giá trị (VD: Phai), sẽ phản tác dụng

Lưu trữ dữ liệu

Các kỹ thuật tổ chức lưu trữ

Storage Area Network

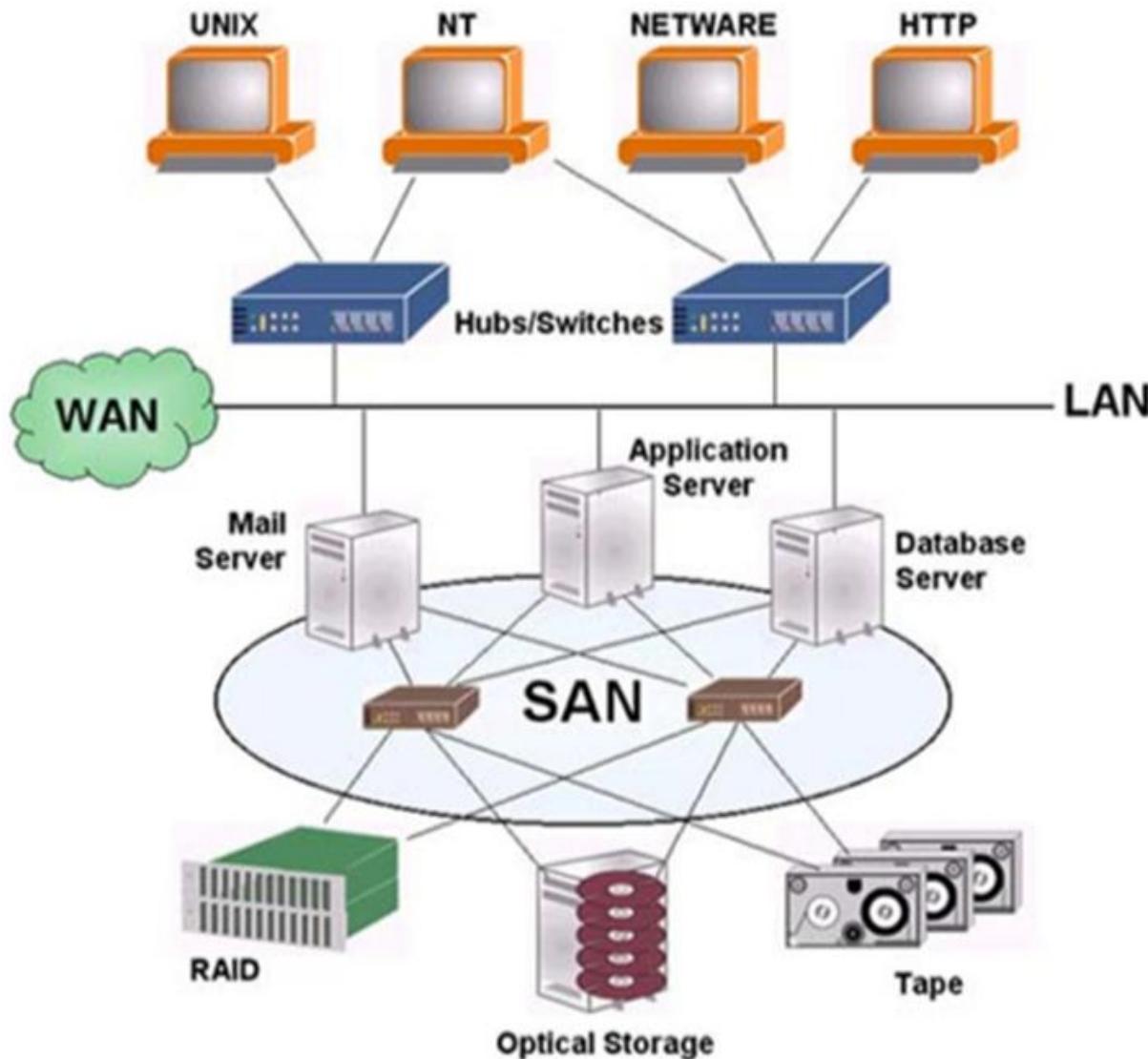
- SAN là hệ thống mạng lưu trữ chuyên dụng, thường dùng ở những nơi
 - lưu trữ nhiều dữ liệu
 - cần độ an toàn, dự phòng cao
 - có thể truy xuất nhanh

Storage Area Network

SAN có 3 thành phần chính:

- **Thiết bị lưu trữ:** là các tủ đĩa chứa dữ liệu chung cho toàn bộ hệ thống có dung lượng lớn, khả năng truy xuất nhanh, có hỗ trợ các chức năng RAID, local Replica, ...
- **Thiết bị chuyển mạch SAN:** các SAN switch thực hiện việc kết nối các máy chủ đến tủ đĩa
- **Các máy chủ hoặc máy trạm cần lưu trữ,** được kết nối đến SAN switch bằng cáp quang thông qua HBA card.

Storage Area Network



Storage Area Network

Lợi ích khi sử dụng SANs:

- ❑ Dễ quản lý, chia sẻ, mở rộng khả năng lưu trữ
- ❑ cho phép nhiều máy chủ cùng chia sẻ một thiết bị lưu trữ.
- ❑ cho phép thay các máy chủ khi đang sử dụng mà không ảnh hưởng
- ❑ cung cấp giải pháp khôi phục dữ liệu nhanh chóng

Redundant Arrays of Independent Disks

- ❑ RAID là hệ thống lưu trữ gồm nhiều đĩa cứng nhằm cải thiện tốc độ đọc ghi dữ liệu và tăng độ tin cậy bằng cách lưu trữ dư thừa thông tin
- ❑ Ban đầu, RAID được sử dụng như một giải pháp phòng hộ
 - ghi dữ liệu lên nhiều đĩa cứng cùng lúc
 - nếu một ổ bị trục trặc thì có thể đổi sang ổ khác
- ❑ Về sau, RAID đã có nhiều biến thể để đảm bảo an toàn dữ liệu và giúp tăng đáng kể tốc độ truy xuất dữ liệu từ đĩa cứng

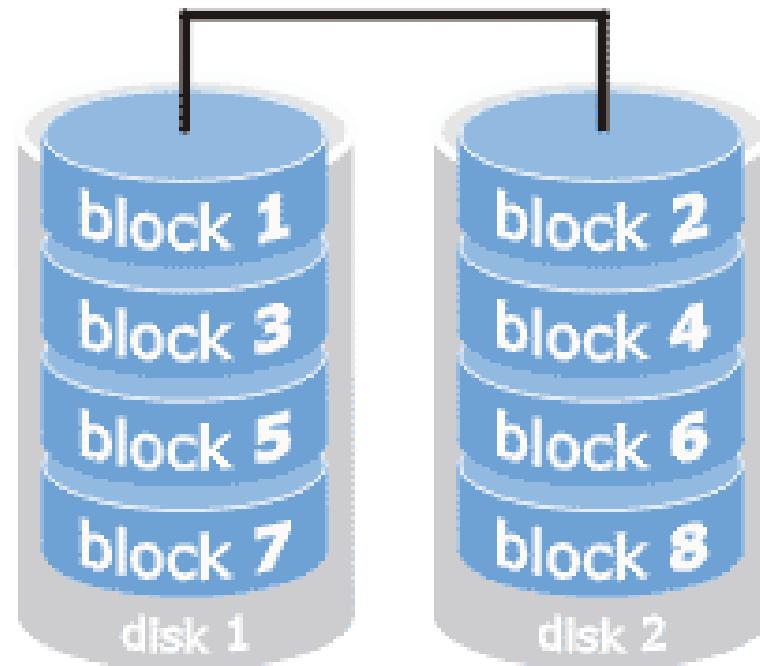
RAID 0

- RAID 0 cần ít nhất hai ổ cứng, sử dụng kĩ thuật gọi là “striping”
 - dữ liệu được ghi thành nhiều phần trên nhiều ổ đĩa
 - tăng hiệu quả thực thi, có thể ghi hai khối dữ liệu cùng lúc tới hai ổ cứng
- RAID 0 thực ra không phải là phiên bản RAID hợp lệ, do không cung cấp bản dự phòng cho dữ liệu lưu trữ
 - kém an toàn
 - một ổ cứng bị lỗi thì không phục hồi lại được dữ liệu

RAID 0

RAID 0

striping



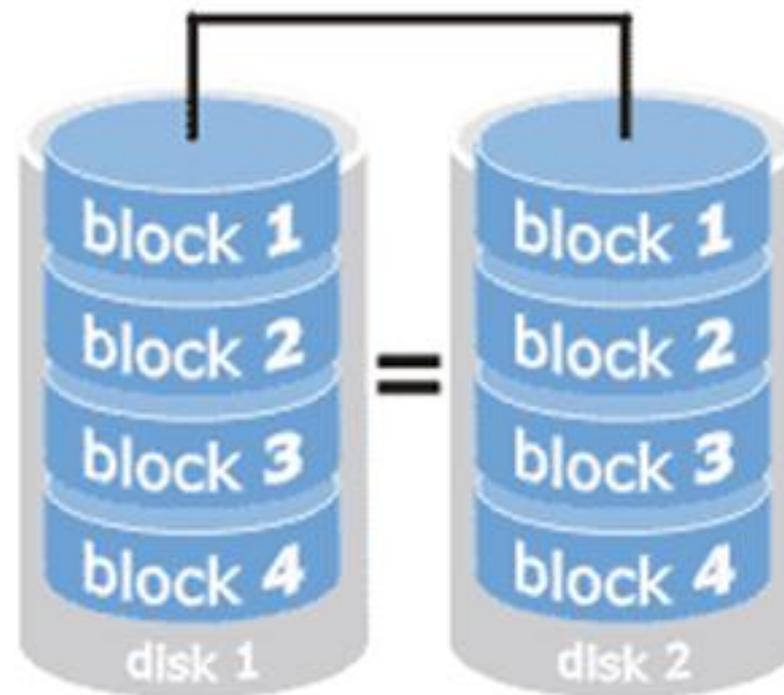
RAID 1

- ❑ RAID 1 cần ít nhất 2 đĩa cứng, cung cấp một phiên bản dự phòng dữ liệu đầy đủ cho hệ thống.
 - Dữ liệu được ghi giống như nhau trên cả 2 đĩa (mirroring)
 - nếu một ổ gặp sự cố, ổ còn lại vẫn hoạt động
- ❑ Ưu điểm là độ an toàn cao. Tuy nhiên, hiệu suất thực thi không phải là yếu tố hàng đầu

RAID 1

RAID 1

mirroring

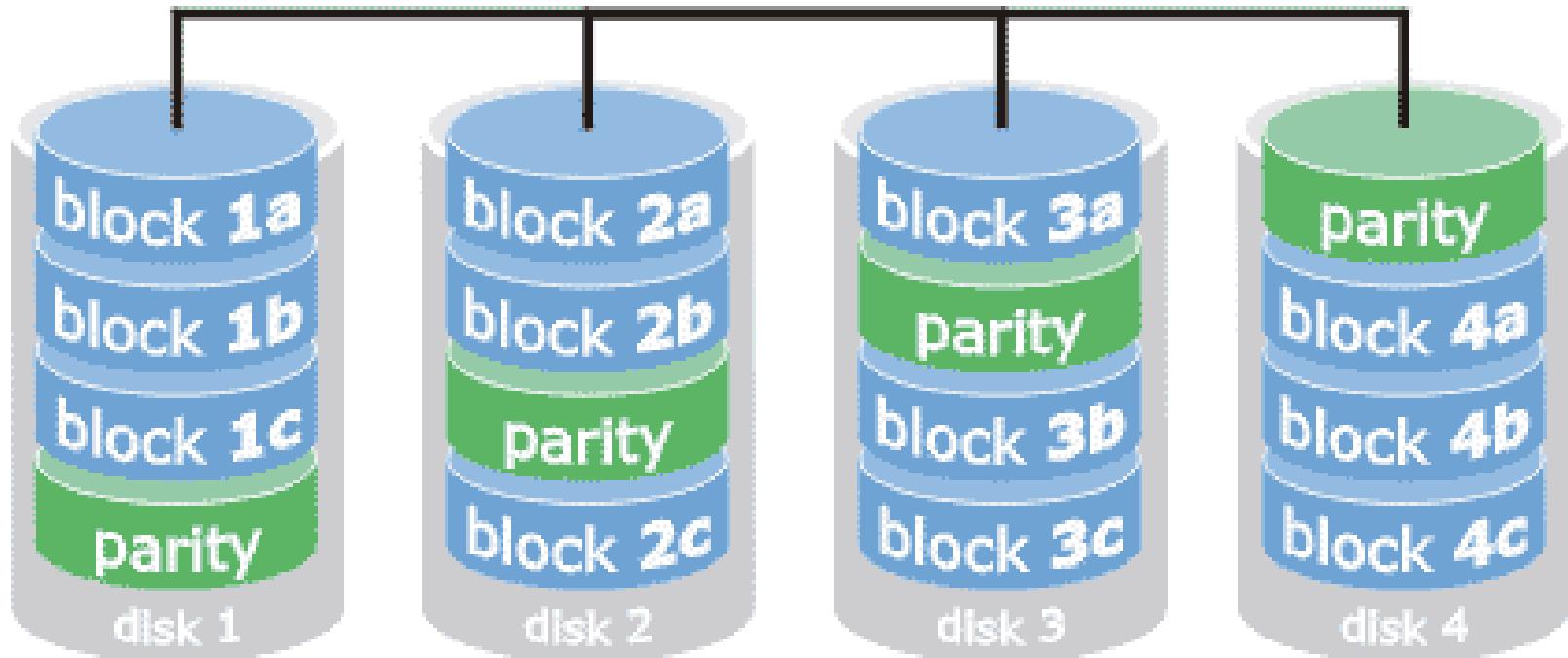


RAID 5

- ❑ RAID 5 cần ít nhất 3 ổ đĩa cứng có năng suất cao như nhau, sử dụng phương pháp phân chia “parity”
 - là phép toán nhị phân so sánh 2 khối dữ liệu với một khối dữ liệu thứ 3
 - một ổ trong dãy bị trực trặc thì sẽ cho phép các bit “parity” khôi phục lại dữ liệu khi ổ đó được thay thế
- ❑ Dữ liệu và bản sao lưu được chia lên tất cả các ổ cứng
- ❑ RAID 5 vừa đảm bảo cải thiện tốc độ, vừa giữ được độ an toàn

RAID 5

RAID 5 parity across disks



Lưu trữ dữ liệu

Chỉ mục

Chỉ mục (Index)

- ❑ Dùng chỉ mục cho tập tin cũng giống như dùng bảng liệt kê danh mục trong thư viện

Về kỹ thuật có 2 loại chỉ mục cơ bản:

- ❑ Chỉ mục sắp thứ tự (ordered indices) dựa trên các giá trị làm index
 - Dùng pp tìm nhị phân trên tập tin chỉ mục
 - Chỉ mục là một tập tin có thứ tự chứa các mẫu tin có chiều dài cố định gồm 2 trường
 - Trường 1: khóa tìm kiếm
 - Trường 2: con trỏ trỏ đến các block
- ❑ Chỉ mục dùng kỹ thuật băm (hash indices)

Chỉ mục

Đánh giá các kỹ thuật chỉ mục

- Loại truy xuất
- Thời gian truy xuất
- Thời gian thêm / xóa
- Không gian đĩa dùng cho chỉ mục

Phân loại chỉ mục

□ Dense index

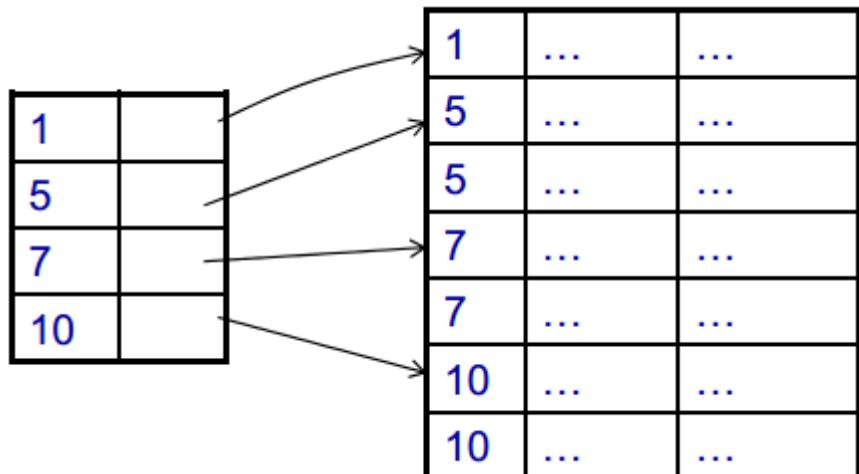
- Tt dữ liệu có bao nhiêu giá trị trên search key thì trên tt chỉ mục có bấy nhiêu mẫu tin
- Mỗi mẫu tin của tt chỉ mục chứa
 - search key
 - con trỏ trỏ đến mẫu tin đầu tiên trên tt dữ liệu có cùng giá trị với trường search key

□ Sparse index

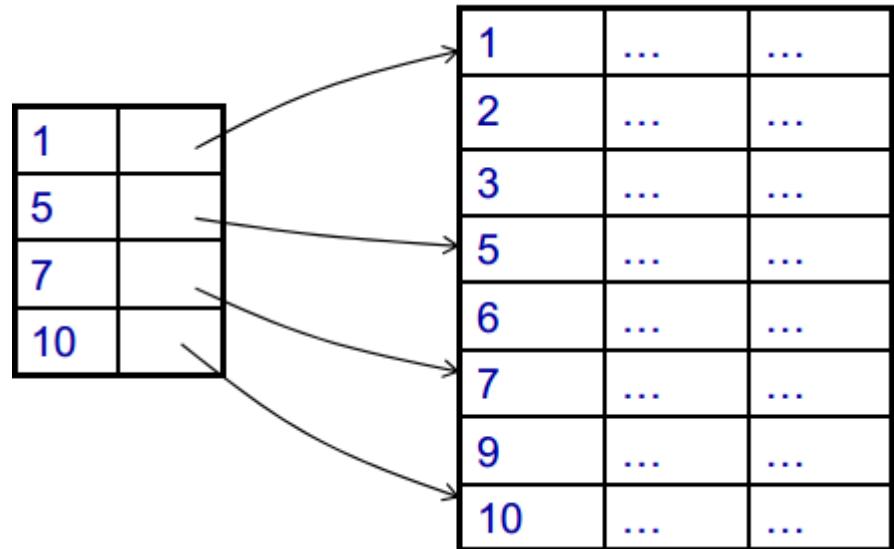
- Các mẫu tin trên tt chỉ mục chỉ ứng với một số giá trị trên tt dữ liệu trên trường search key
- Để tìm 1 giá trị, ta tìm trong tt chỉ mục một mẫu tin sao cho giá trị search key lớn nhất \leq giá trị cần tìm, và duyệt mẫu tin xuất phát từ vị trí đầu tiên mà con trỏ chỉ đến

Phân loại chỉ mục

❑ Dense Index



❑ Sparse Index



Chỉ mục

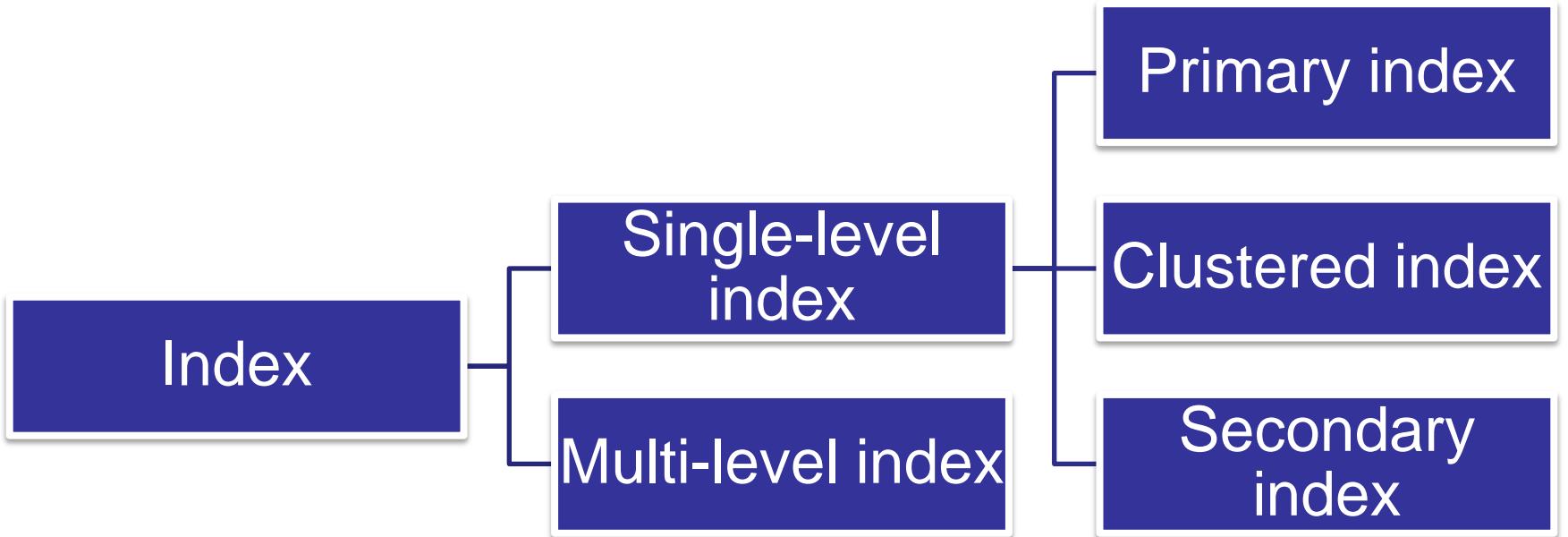
- ❑ Chỉ mục là cơ chế giúp HQT CSDL truy xuất dữ liệu nhanh
- ❑ Khóa chỉ mục (index key): một hoặc nhiều trường dùng làm chỉ mục
 - simple index: index key chỉ có 1 trường
 - composite index: index key có nhiều trường
- ❑ VD:
NOIGD(..., TENNGD, SONHA, DUONG, QH, TP)

Nhu cầu tìm nhanh một địa chỉ giao dịch: index key là DUONG (simple index) thì không hiệu quả, mà phải dùng SONHA, DUONG, QH, TP (composite index)

Chỉ mục

- ❑ Mỗi cấu trúc chỉ mục kết hợp với một index key cụ thể
- ❑ Bất cứ trường nào cũng có thể là chỉ mục, và có thể có nhiều chỉ mục trên cùng một tập tin
- ❑ Chỉ mục hiệu quả hay không căn cứ vào
 - loại dữ liệu mà trên đó thiết lập chỉ mục
 - giá trị trên index key có phân biệt hay không
 - loại câu SQL được dùng
 - các truy cập khác trên bảng, nếu cập nhật nhiều trên trường chỉ mục sẽ làm chậm hệ thống
 - có quá nhiều chỉ mục sẽ làm chậm hệ thống

Chỉ mục

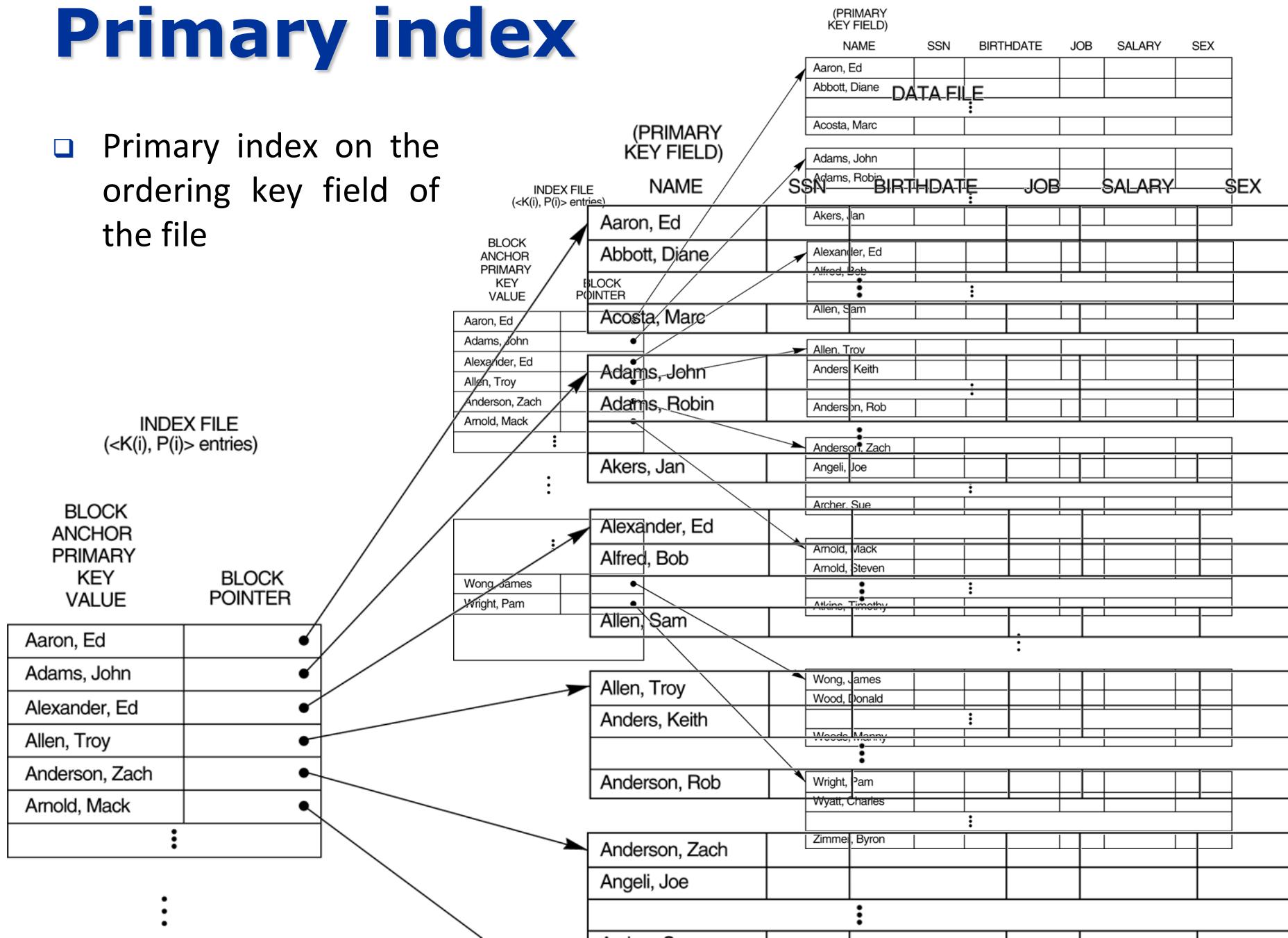


Primary index

- ❑ Được tạo trên trường làm khóa sắp xếp cho tt dữ liệu. Thứ tự vật lý của các mẫu tin trên đĩa cũng dựa trên trường này, và trên đó các mẫu tin có giá trị duy nhất
 - nếu có nhiều mẫu tin cùng giá trị trên trường dùng để sắp xếp, ta sẽ tạo clustering index trên trường này
- ❑ Có 1 mẫu tin chỉ mục trong tập tin chỉ mục ứng với một block trong tt dữ liệu
- ❑ Tt primary index có kích thước nhỏ hơn rất nhiều so với tt dữ liệu
 - mẫu tin đầu tiên trong mỗi block của tt dữ liệu gọi là anchor record hay block anchor
- ❑ Chỉ có thể có 1 primary index, hoặc 1 clustering index trên 1 tt dữ liệu, không thể có cả 2 loại index này trên cùng 1 tt

Primary index

- ❑ Primary index on the ordering key field of the file

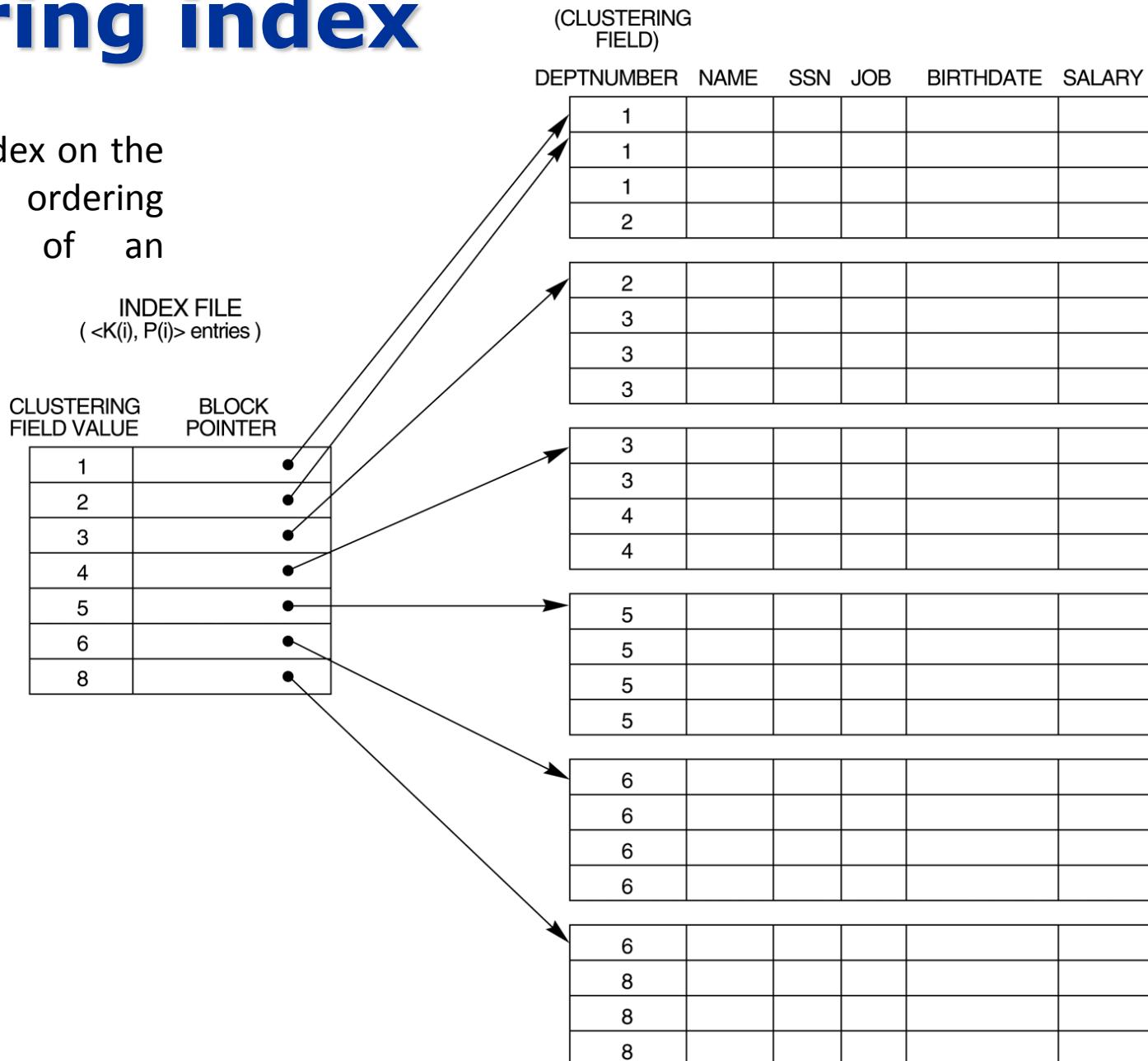


Clustering index

- ❑ Nếu tt dữ liệu được sắp vật lý theo trường không phải là khóa thì trường đó chính là clustering field
- ❑ Có 1 mẫu tin trên tt chỉ mục chứa 1 giá trị của trường clustering, con trỏ trỏ đến block đầu tiên chứa giá trị phân biệt

Clustering index

- A clustering index on the DEPTNUMBER ordering nonkey field of an EMPLOYEE file

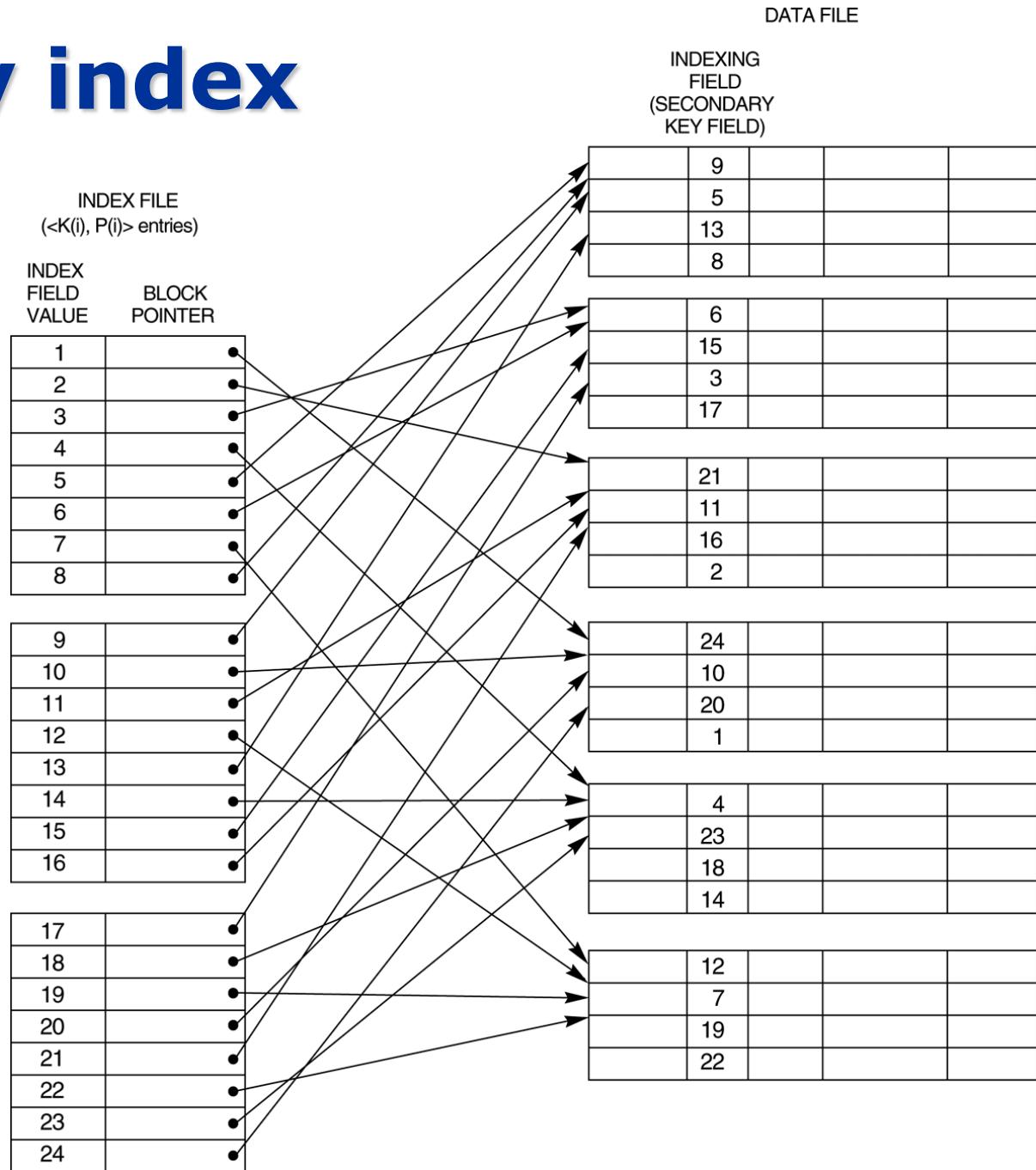


Secondary index

- ❑ Một secondary index cung cấp thêm phương tiện để truy cập tt, ngoài primary index ra
 - Được tạo trên trường là khóa ứng viên và có giá trị duy nhất trên mỗi mẩu tin, dữ liệu của tt dữ liệu không được sắp thứ tự trên trường này
 - Cũng có thể tạo trên trường không phải là khóa và có giá trị trùng nhau
 - Trường thứ nhất là trường dữ liệu không được sắp thứ tự của tt dữ liệu, và cần tìm kiếm trên đó
 - Trường thứ hai là con trỏ trỏ đến block đầu tiên chứa giá trị, hoặc trỏ đến mẩu tin chứa giá trị
- ❑ Có thể tạo nhiều secondary index cho 1 tt dữ liệu

Secondary index

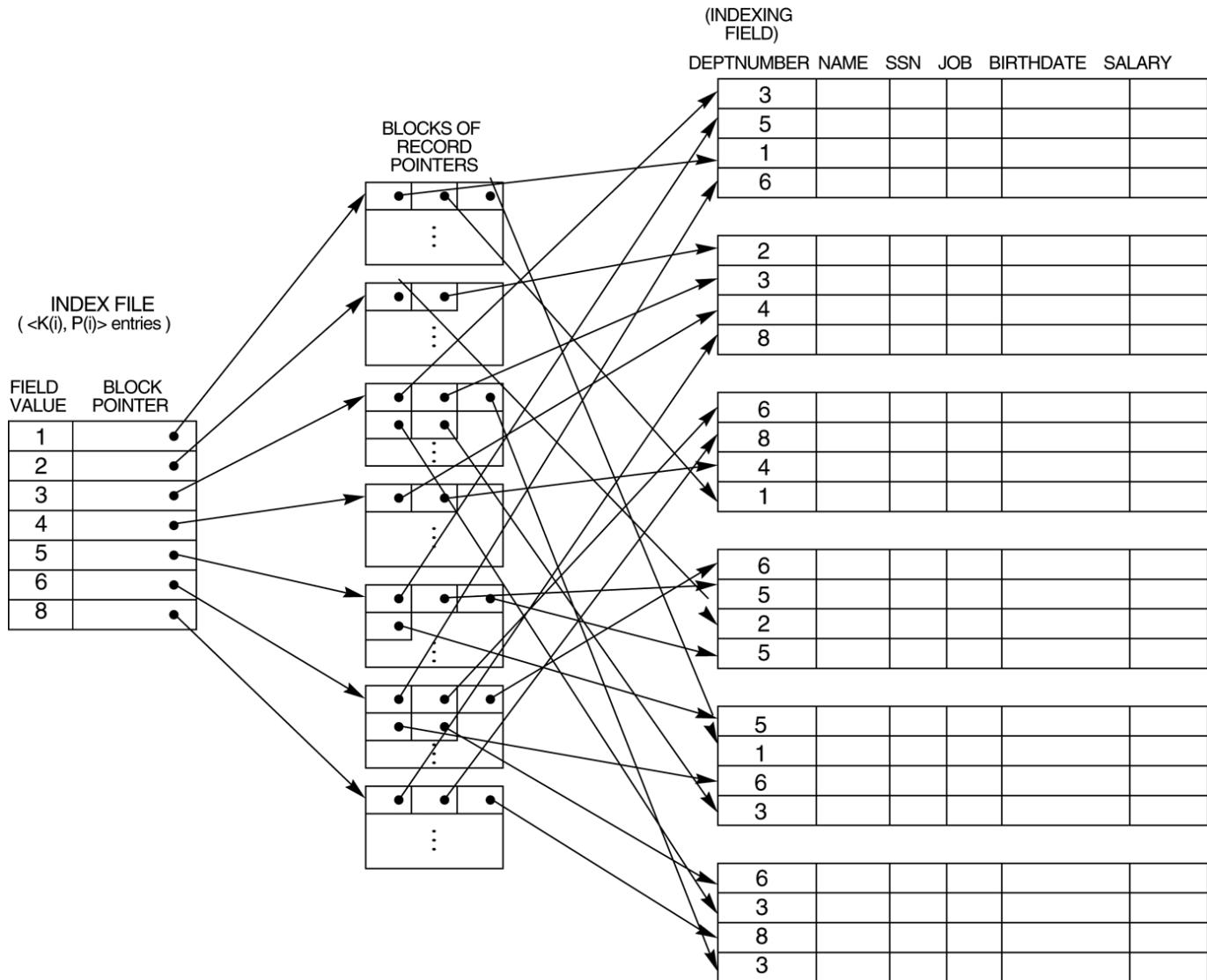
- A dense secondary index (with block pointers) on a nonordering key field of a file



Secondary index

DATA FILE

- A secondary index (with record pointers) on a nonkey field implemented using one level of indirection so that index entries are of fixed length and have unique field values



Nhận xét

	Tt dữ liệu xếp theo index field	Tt dữ liệu không xếp theo index field
Index field làm khóa	Primary index	Secondary index (key)
Index field không làm khóa	Clustering index	Secondary index (nonkey)

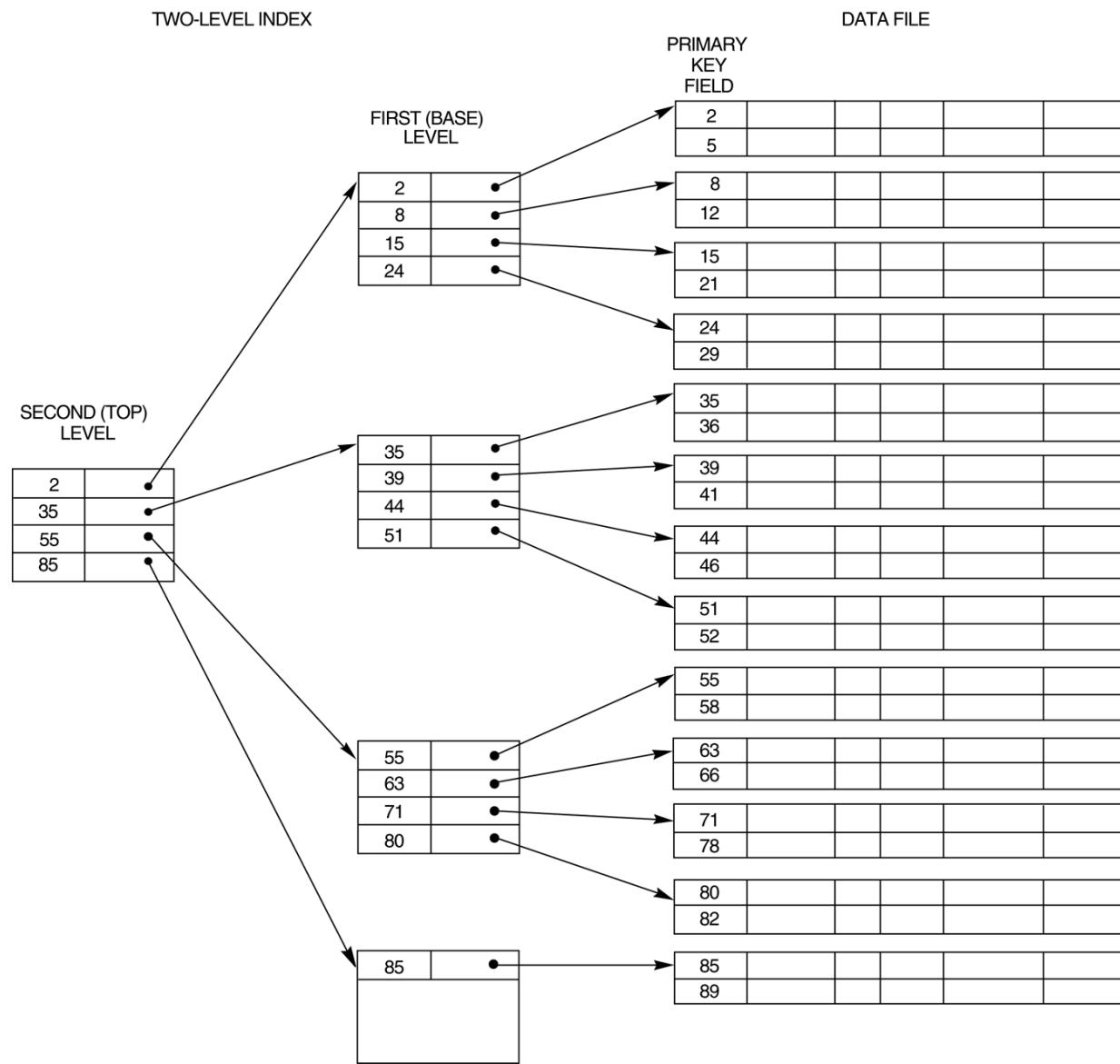
- 1 tập tin có thể có nhiều chỉ mục vì có thể có nhiều nhu cầu tìm kiếm trên tập tin

Multi-level index

- Chỉ mục có thể rất lớn, khi truy cập phải đọc nhiều block → dùng chỉ mục nhiều mức
- Xem tt chỉ mục như một tt tuần tự và xây dựng chỉ mục thừa cho nó
 - Tìm mẫu tin có khóa tìm kiếm lớn nhất trong các mẫu tin có search key \leq khóa muốn tìm trên tt chỉ mục ngoài, con trỏ tương ứng trỏ tới block của chỉ mục trong.
 - Trong block này, tìm mẫu tin có khóa tìm kiếm lớn nhất trong các mẫu tin có search key \leq khóa muốn tìm trên tt chỉ mục trong, trường con trỏ trỏ tới block chứa mẫu tin muốn tìm.

Multi-level index

- A two-level primary index resembling ISAM (Indexed Sequential Access Method) organization



Cây cân bằng (B-Tree)

- ❑ B-Tree là một cây có gốc thỏa điều kiện:
 - Tất cả đường đi từ nút gốc 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

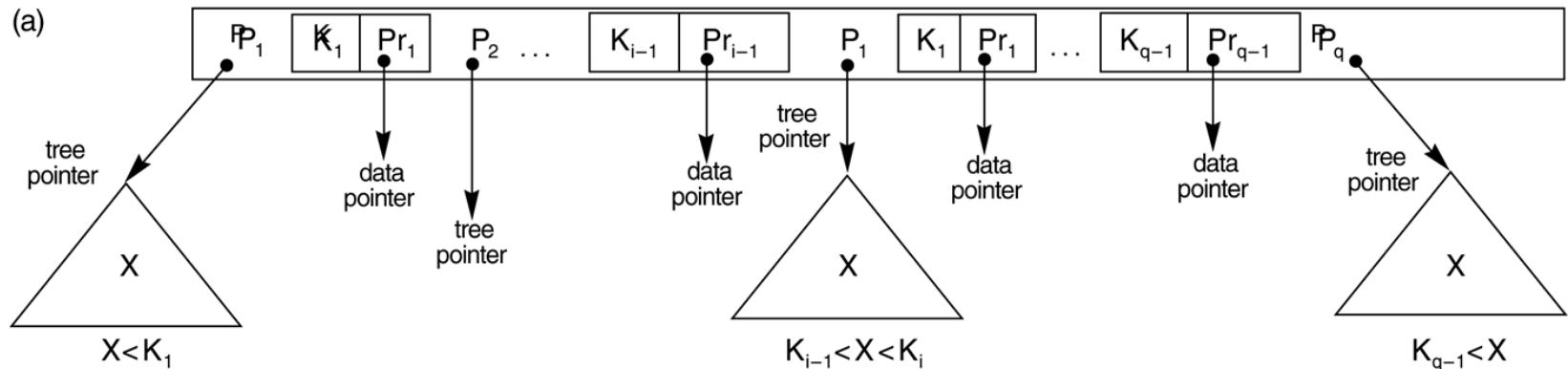
❑ Cấu trúc 1 nút

P_1	K_1	P_2	...	P_{n-1}	K_{n-1}	P_n
-------	-------	-------	-----	-----------	-----------	-------

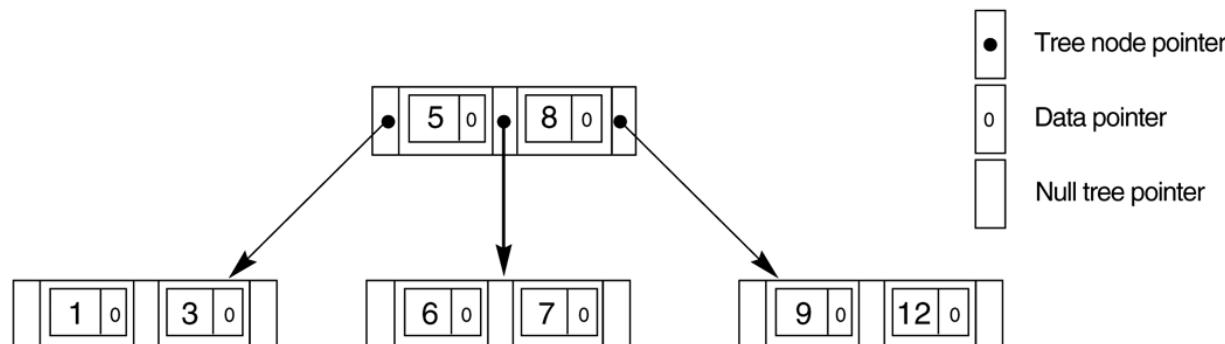
- ❑ Khóa tìm kiếm trên 1 nút được sắp thứ tự tăng dần $K_1 < K_2 < \dots < K_{n-1}$

B-Tree

(a) A node in a B-tree with $q - 1$ search values.



(b)



(b) A B-tree of order $p = 3$. The values were inserted in the order 8, 5, 1, 7, 3, 12, 9, 6.

Dùng chỉ mục

- ❑ Nên tạo chỉ mục trên các trường có giá trị phân biệt, được truy xuất với tần suất cao, và kết quả 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