BCNF,
Memory Hierarchy & File Structures

Exercise: BCNF

- BCNF: Boyce-Codd Normal Form： Boyce-Codd范式

- R is in BCNF if and only if
  for each FD: $X \rightarrow \{A\}$ in F+,
  $A \in X$ (trivial FD), or
  X is a superkey for R

- In other words: 对于每个不包含无关（无用）属性的 FD，每个 FD 的 左侧 都是候选键。

Exercise: BCNF (cont)

R =(A, B, C, D).

F = {C → D, C → A, B → C}.

Decompose R into a set of BCNF relations.

- S={R}. When we find a schema R with BCNF violation X → Y we:
    1. 把R从S中删除
    2. 新建一个schema加入S，其属性集为 R – Y
    3. 新建一个schema加入S，其属性集为 X和Y的并集

Exercise: BCNF (cont)

R =(A, B, C, D).

F = {C $\rightarrow$ D, C $\rightarrow$ A, B $\rightarrow$ C}.

Decompose R into a set of BCNF relations.

- List all candidate keys of R.

  {B}

- C $\rightarrow$ D, C $\rightarrow$ A violates BCNF. Because C is not a super key for R.

  Take C $\rightarrow$ D: decompose R to

  R_1= {A, B, C} , R_2={C, D}.

- C $\rightarrow$ A in R_1 violates BCNF. C is not a super key for R_1.

  Decompose R_1 to

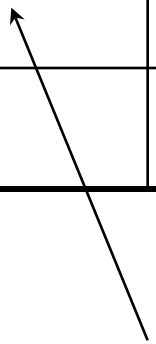
  R_{11} = {B, C} R_{12} = {C, A}.

  Final decomposition:

  R_2 = {C, D}, R_{11} = {B, C}, R_{12} = {C, A}.

Memory Hierarchy

	Cost (money)	Access (time)	volatile	Capacity
Cache	Very expensive	Very Fast	Yes	KB -> MB
Main memory	Expensive	Fast	Yes	MB -> GB
Disk	Inexpensive	Slow	No	GB -> TB
Tape	Very inexpensive	Very slow	No	TB -> PB



Database

ref: numbers

Accessing Disk

- Access unit: disk blocks or pages

(typically a block is 1K-16Kbytes)

- Physical movement, random access

- Time to access (read/write) a disk block

seek time + rotational delay + transfer time

Data File Organization

Database<->a collection of files

File<->a sequence of records

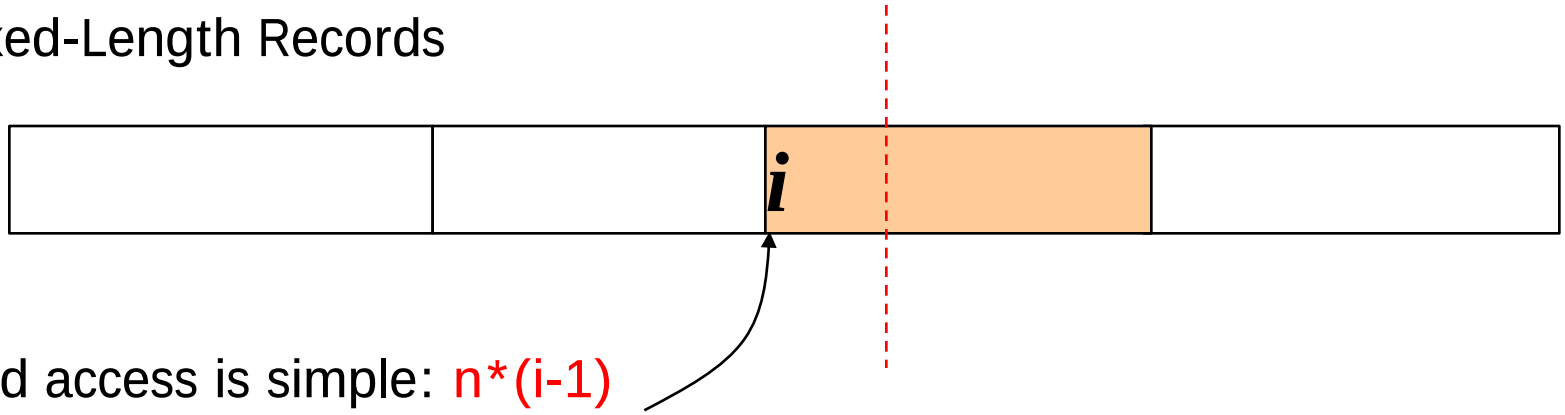
Record<->a sequence of fields(属性、域值)

Storing records in files

- Fixed-Length Records
  - Free Lists
- Variable-Length Records
  - Byte String Representation
  - Reserved space
  - Pointer Method
  - Slotted Page Structure

Fixed-Length Records

- Fixed-Length Records



Record access is simple: n*(i-1)

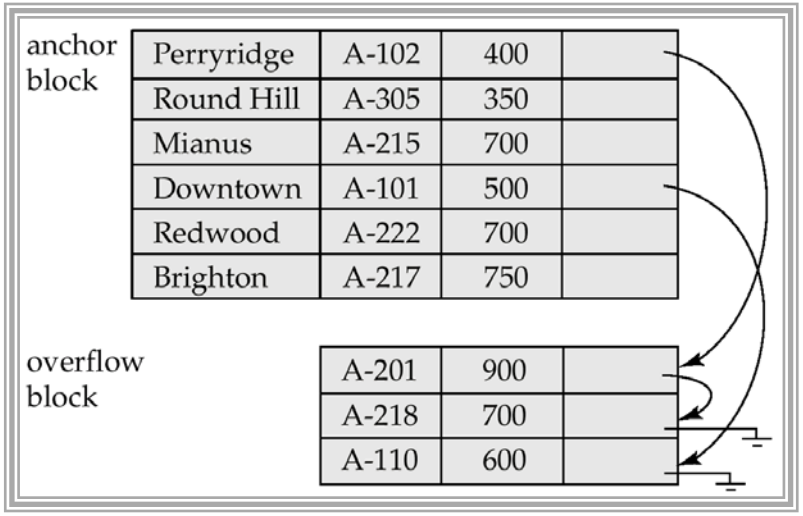
Problem: what if deletion?

- Free list

header				
record 0	A-102	Perryridge	400	
record 1				
record 2	A-215	Mianus	700	
record 3	A-101	Downtown	500	
record 4				
record 5	A-201	Perryridge	900	
record 6				
record 7	A-110	Downtown	600	
record 8	A-218	Perryridge	700	

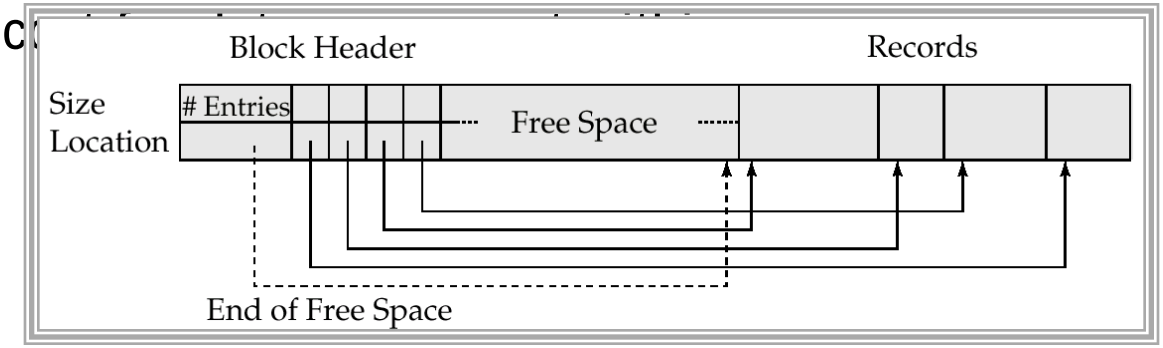
Data File Organization

- Variable-Length Records
  - Byte String Representation : Attach an *end-of-record* (⊥) control character to the end of each record
    - Problem: deletion & record growth
  - Reserved space : use fixed-length records of a known maximum length
    - Problem: waste storage space
  - Pointer Method : Anchor block + Overflow block



Data File Organization

- Variable-Length Records
  - Byte String Representation : Attach an *end-of-record* (⊥) control character to the end of each record
    - Problem: deletion & record growth
  - Reserved space : use fixed-length records of a known maximum length
    - Problem: waste storage space
  - Pointer Method : Anchor block + Overflow block
  - Slotted Page Structure :
    - Low c



Data File Organization

Database<->a collection of files

File<->a sequence of records

Record<->a sequence of fields

- Heap
  - record can be placed anywhere in the file where there is space
- Sequential
  - store records in sequential order
- Hashing
  - a hash function computed on some attribute of each record

Exercise 1

- Assume that a school keeps a file with the records of its students: Student (sid:4 bytes, sname: 10 bytes, dept-id: 4 bytes), dept-id is the department(院系) id where a student belongs to.

- There exist 10,000 student records and 50 departments. A page is 128 bytes. The data file is sorted sequentially on sid.

- Q1: What's the size of a record?
- Q2: How many pages do we need to store these student records?
- Q3: Given this data file, what's the cost to find a particular student by sid?

Exercise 1

- Student (sid:4 bytes, sname: 10 bytes, dept-id: 4 bytes)

- There exist 10,000 student records and 50 departments. A page is 128 bytes and a pointer is 4 bytes. The data file is sorted sequentially on sid.

- What's the size of a record?

  4 + 10 + 4 = 18

Exercise 1

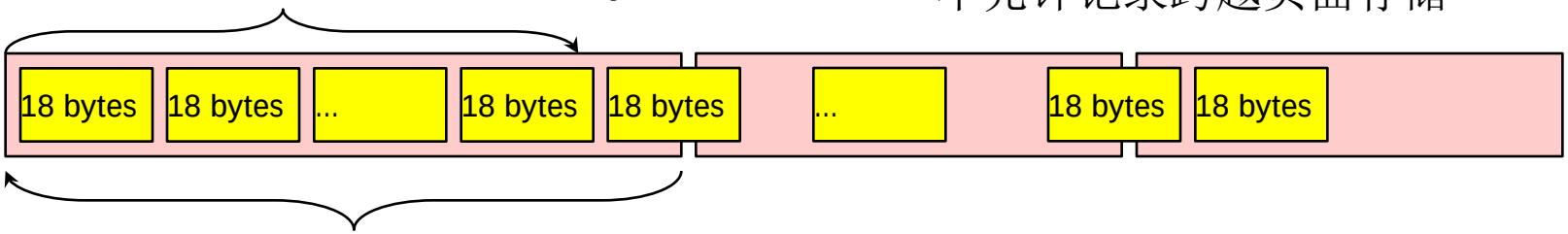
- Student (sid:4 bytes, sname: 10 bytes, dept-id: 4 bytes)
- There exist 10,000 student records and 50 departments. A page is 128 bytes and a pointer is 4 bytes. The data file is sorted sequentially on sid.
- How many pages do we need to store these student records?

(#records * size of record) / size of page =

10,000 * 18 / 128 = 1406.25 ≈ 1407

7 records, of size 18*7=126 bytes

不允许记录跨越页面存储

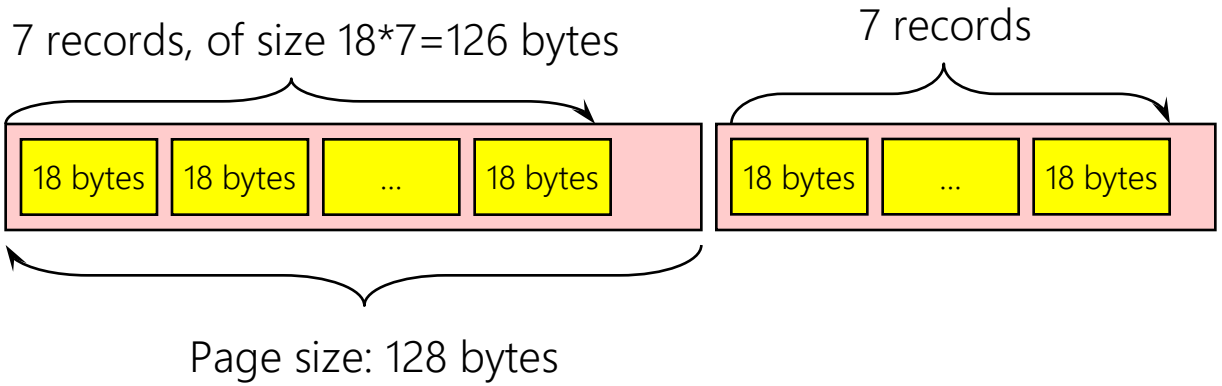


Page size: 128 bytes

Exercise 1

- Student (sid:4 bytes, sname: 10 bytes, dept-id: 4 bytes)
- There exist 10,000 student records and 50 departments. A page is 128 bytes and a pointer is 4 bytes. The data file is sorted sequentially on sid.
- How many pages do we need to store these student records?

$$\lceil (\#\ \text{of records} / \lfloor \text{size of page} / \text{size of record} \rfloor) \rceil =$$
$$\lceil (10{,}000 / \lfloor 128 / 18 \rfloor) \rceil = 1429$$



Exercise 1

- Student (sid:4 bytes, sname: 10 bytes, dept-id: 4 bytes)

- There exist 10,000 student records and 50 departments. A page is 128 bytes and a pointer is 4 bytes. The data file is sorted sequentially on sid.

- Q3: Given this data file, what's the cost to find a particular student by sid?

  # of pages required: 1429

Binary search on sid:

$\lceil \log_2(\#\text{ of pages}) \rceil = \lceil \log_2 1429 \rceil = 11$