COMP 3311
DATABASE MANAGEMENT SYSTEMS

TUTORIAL 5
STORAGE AND FILE STRUCTURE

DATA FILE ORGANIZATION

Database ⇔ a collection of files

File ⇔ a sequence of records

Record ⇔ a sequence of fields

Record Organization

- Fixed-Length Records
  - Relative location of records
  - Free lists for deleted records
- Variable-Length Records
  - Byte-string representation
  - Embedded identification
  - Reserved space
  - Pointer method
  - Slotted-page Structure

FIXED-LENGTH RECORDS

record 1	**…**	~~**record i**~~	**…**

- Record access is simple: n*(i-1)
  - where n is the (fixed) record length in bytes
  - i is the record number
  - addressing starts at byte 0

Problem: what if there is a deletion?

- use a free list

account#, branchName, balance

header			
record 1	A-102	Perryridge	400
record 2			
record 3	A-215	Mianus	700
record 4	A-101	Downtown	500
record 5			
record 6	A-201	Perryridge	900
record 7			
record 8	A-110	Downtown	600
record 9	A-218	Perryridge	700

VARIABLE-LENGTH RECORDS

Byte-string Representation

– Attach an *end-of-record* (⊥) control character to the end of each record.

  Problem: deletion and record growth.

Reserved space

– Use fixed-length records of a known maximum length.

  Problem: wastes storage space.

Pointer Method

– Anchor page + Overflow page
  - ➢ For Perryridge insert: A-210, 900
  - ➢ For Perryridge insert: A218, 700
  - ➢ For Downtown insert: A110, 600

branchName, account#, balance

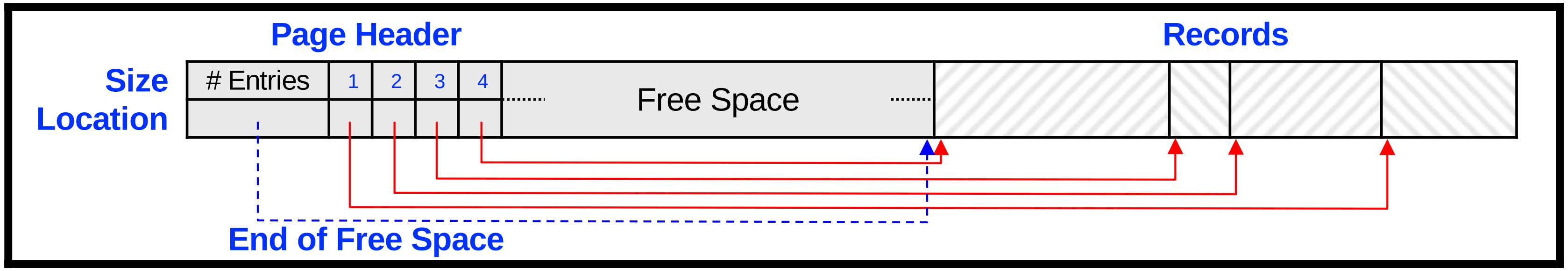
Anchor page

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

Overflow page

A-210	900	⊥
A-218	700	⊥
A-110	600	⊥

VARIABLE-LENGTH RECORDS (CONT'D)



Slotted Page Structure

- The page header contains:
  - the number of record entries.
  - the location of the end of the free space in the page.
  - the location and size of each record.
- The cost for moving data within a page is low.

FILE ORGANIZATION

Database ⇔ a collection of files

File ⇔ a sequence of records

Record ⇔ a sequence of fields

Heap	A record can be placed anywhere in the file where there is space. There is no relationship between the search key value and a record's location.
Sequential	Records are stored in sequential order, based on a search key usually the primary key).
Hash	A hash function is applied to the search key value of a record; the result specifies in which bucket (page) of the file the record should be placed.

PAGE I/O COST OF OPERATIONS

Operation	Heap File	Sequential File	Hash File
Scan all records	B	B	1.25^1 B
Equality search[2]	0.5 B	$\log_2 B^3$	1
Range search	B	$\log_2 B$ + # of pages with matches	1.25^1 B
Insert	2^4	Equality search + B^5	2
Delete	Equality search + 1	Equality search + B^5	2

B is the number of pages in a file.

1 Assumes 80% occupancy of pages to allow for future additions. Thus 1.25B pages are needed to store all records.

2 Assumes the search is on the key value.

3 Assumes binary search is used.

4 Assumes the record is inserted at the end of the file – read last page and write it back.

5 Assumes insert/delete is in the middle of the file and need to read and write all pages in second half of the file.

EXERCISE 1

Which file organization, heap, sequential or hash, would you choose for a file where the most frequent operations are:

a) **search for records based on a range of field values?**

   sequential file

b) **perform inserts and scans, where the order of records does not matter?**

   heap file

c) **search for a record based on a particular field value?**

   hash file

EXERCISE 2

A file has 10,000 student records of fixed-length. Each record has the following fields: studentId (9 bytes), name (30 bytes), address (40 bytes), phone (8 bytes), birthdate (8 bytes), gender (1 byte) and degreeProgram (3 bytes). An additional byte is used as a deletion marker.

a) **What is the size of a record in bytes?**

Record size: 9+30+40+8+8+1+3+1 = 100 bytes

b) **What is the blocking factor bf_r if the page size is 4096 bytes?**

bf_r: $\lfloor$# bytes per page / # bytes per record$\rfloor$

$= \lfloor 4096 / 100 \rfloor = 40$ records/page

records: 10000
bf_r: 40

EXERCISE 2 (CONT'D)

c) **How many pages are required to store the file:**

i. **if a sequential file organization is used?**

 # pages: $\lceil \#\text{ records} / bf_r \rceil = \lceil 10000 / 40 \rceil =$ 250 pages

ii. **if a heap file organization is used?**

 # pages: $\lceil \#\text{ records} / bf_r \rceil = \lceil 10000 / 40 \rceil =$ 250 pages

iii. **if a hash file organization is used (assuming 100% page occupancy)?**

 # pages: $\lceil \#\text{ records} / bf_r \rceil = \lceil 10000 / 40 \rceil =$ 250 pages

iv. **if a hash file organization is used (assuming 80% page occupancy)?**

 # pages: $\lceil \#\text{ records} / bf_r * 1.25 \rceil = \lceil 10000 / 40 * 1.25 \rceil =$ 313 pages

records: 10000
bf_r: 40
#pages: 250

EXERCISE 2 (CONT'D)

d) **Consider the query "Find a student record given a particular student id". Assuming that a record with the student id exists in the file, what is the cost, in page I/Os, to answer this query:**

i. **if a sequential file organization is used?**

A binary search can be used.

Search cost: $\lceil \log_2 (\text{\# of pages}) \rceil = \lceil \log_2 250 \rceil = \underline{8}$ page I/Os

ii. **if a heap file organization is used?**

A scan of the pages is required, but *on average only half the pages need to be scanned*.

Search cost: $\lceil (\text{\# of pages}) / 2 \rceil = \lceil 250 / 2 \rceil = \underline{125}$ page I/Os

iii. **if a hash file organization is used?**

Search cost: $\underline{1}$ page I/O

You must upload your completed exercise sheet to Canvas by **11 p.m. today.**

EXERCISE 3

A school keeps the following file with the records of its students :

Student(studentId: 4 bytes, name: 10 bytes, deptId: 4 bytes)

where deptId is the department id to which a student belongs.

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

a) What is the size of a record in bytes?

b) How many records can fit on each page?

c) How many pages are needed to store these student records?

d) Given this data file, what is the cost, in page I/Os, to find a particular student given a studentId?

EXERCISE 3 (CONT'D)

Student(studentId: 4 bytes, name: 10 bytes, deptId: 4 bytes)

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

a) **What is the size of a record in bytes?**

Record size: 4 bytes + 10 bytes + 4 bytes = 18 bytes

b) **How many records can fit on each page?**

$bf_{\text{Student}} = \lfloor 128 / 18 \rfloor = 7$ records/page

These records will occupy 18*7 = 126 bytes on a page.

EXERCISE 3 (CONT'D)

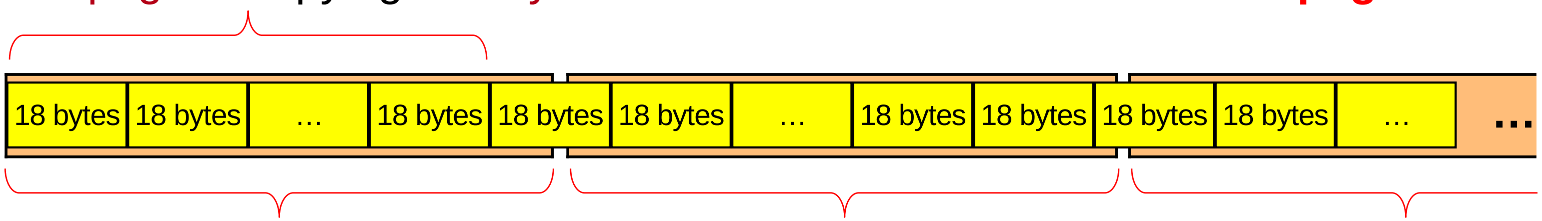
Student(studentId: 4 bytes, name: 10 bytes, deptId: 4 bytes)

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

c) How many pages are needed to store these student records?

 # pages: ~~(#records * size of record) / size of page = 10,000 * 18 / 128 = 1406.25 = 1407 pages~~

7 records/page occupying 126 bytes **Do not allow records to cross page boundaries!**



Page size: 128 bytes

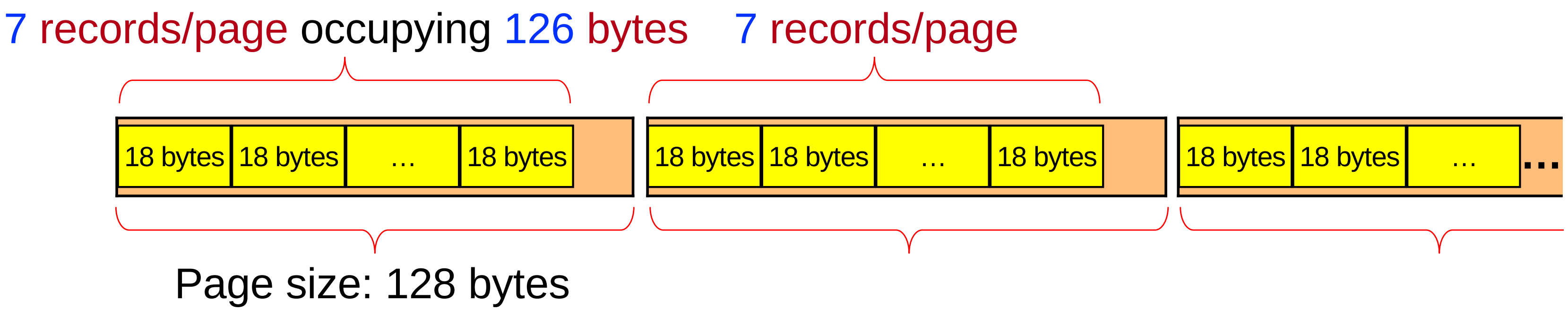
EXERCISE 3 (CONT'D)

Student(studentId: 4 bytes, name: 10 bytes, deptId: 4 bytes)

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

c) How many pages are needed to store these student records?

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



EXERCISE 3 (CONT'D)

Student(studentId: 4 bytes, name: 10 bytes, deptId: 4 bytes)

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

d) Given this data file, what is the cost, in page I/Os, to find a particular student given a studentId?

 # pages: 1429

Binary search on studentId: $\lceil \log_2(\text{\# of pages}) \rceil$
$= \lceil \log_2 1429 \rceil = \underline{11}$ page I/Os