COMP 3311
DATABASE MANAGEMENT SYSTEMS

LECTURE 11 EXERCISES
INDEXING: INTRODUCTION

$bf = \lfloor \# \text{ bytes per page} / \# \text{ bytes per record} \rfloor$
$\# \text{ pages} = \lceil \# \text{ records} / bf_r \rceil$

Film records: 30,000
Actor records: 100,000
Page size: 512 bytes
Pointer size: 6 bytes

EXERCISE 1

A movie database has the following files and sizes of each field:

84 bytes/record Film(title: 40 bytes, director: 20 bytes, releaseYear: 4 bytes, company: 20 bytes)

28 bytes/record Actor(id: 4 bytes, name: 20 bytes, dateOfBirth: 4 bytes)

There are 30,000 film and 100,000 actor records.
Each page is 512 bytes. Each pointer is 6 bytes.

a) What is the blocking factor bf_F for the Film file and bf_A for the Actor file?

bf_F: $\lfloor 512 \text{ bytes per page} / 84 \text{ bytes per Film record} \rfloor$
$= \underline{6}$ records/page

bf_A: $\lfloor 512 \text{ bytes per page} / 28 \text{ bytes per Actor record} \rfloor$
$= \underline{18}$ records/page

$bf = \lfloor$# bytes per page / # bytes per record$\rfloor$
pages = $\lceil$# records / $bf_r\rceil$

Film records: 30,000
Actor records: 100,000
Page size: 512 bytes
Pointer size: 6 bytes
Film record size: 84 bytes; $bf_F = 6$
Actor record size: 28 bytes; $bf_A = 18$

EXERCISE 1 (CONT'D)

b) Assuming the Film file is ordered on title and there is no index, what is the page I/O cost for:

i. Finding the film with title "Titanic"?

Pages needed: $\lceil$30,000 Film records / 6 Film records per page$\rceil$ = 5000 pages

Page I/O cost: $\lceil \log_2 5000 \rceil$ = 13 page I/Os (binary search)

ii. Finding all the films directed by "John Woo"?

Page I/O cost: 5000 page I/Os **Why?**

Explanation: A sequential scan is needed since the file is not ordered based on director.

$bf = \lfloor \# \text{ bytes per page} / \# \text{ bytes per record} \rfloor$
$\# \text{ pages} = \lceil \# \text{ records} / bf_r \rceil$

Film records: 30,000
Actor records: 100,000
Page size: 512 bytes
Pointer size: 6 bytes
Film record size: 84 bytes; $bf_F = 6$
Actor record size: 28 bytes; $bf_A = 18$

EXERCISE 2

Assume the Actor file is ordered on name and we want to create an ordered index on id (4 bytes) where each index entry has the form <id, *pointer*>.

a) What is bf_{Aindex} if the index is single-level?

bf_{Aindex}: $\lfloor 512$ bytes per page / (4 + 6) bytes per index entry$\rfloor$ = 51

b) How many index entries are needed? (Briefly explain your answer.)

Index entries: 100,000 index entries **Why?**

Explanation: A dense index is needed (i.e., one entry per Actor record) since the file is ordered on name, not on id.

c) How many pages are required for the Actor index entries?

Pages needed: $\lceil 100{,}000$ Actor records / 51 index entries per page$\rceil$ = 1961

$bf = \lfloor \# \text{ bytes per page} / \# \text{ bytes per record} \rfloor$
$\# \text{ pages} = \lceil \# \text{ records} / bf_r \rceil$

Film records: 30,000
Actor records: 100,000
Page size: 512 bytes
Pointer size: 6 bytes
Film record size: 84 bytes; $bf_F = 6$
Actor record size: 28 bytes; $bf_A = 18$

EXERCISE 2 (CONT'D)

d) What is the page I/O cost of retrieval based on a single id value using the Actor index (e.g., "Find actor with id 100")?

Actor file ordered on name.

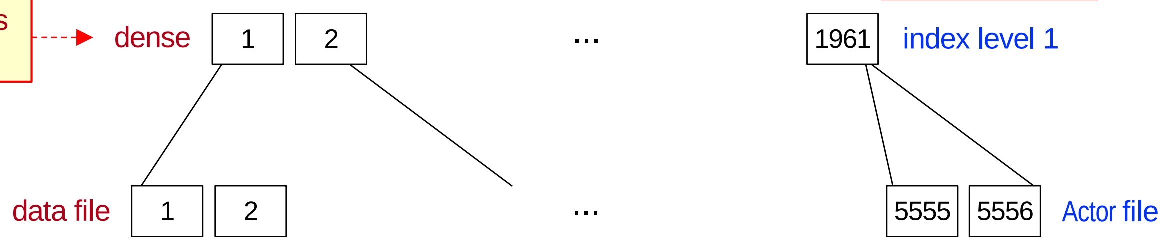
Page I/O cost: $\lceil \log_2 1961 \rceil + 1 = \underline{12}$ page I/Os

Actor single-level index

$bf_{Aindex} = 51$

100,000 index entries, 51 entries/page ⟹ 1961 pages (need to point to every record)

100,000 records, 18 records/page ⟹ 5556 pages



$bf = \lfloor$# bytes per page / # bytes per record$\rfloor$
pages = $\lceil$# records / $bf_r\rceil$

EXERCISE 2 (CONT'D)

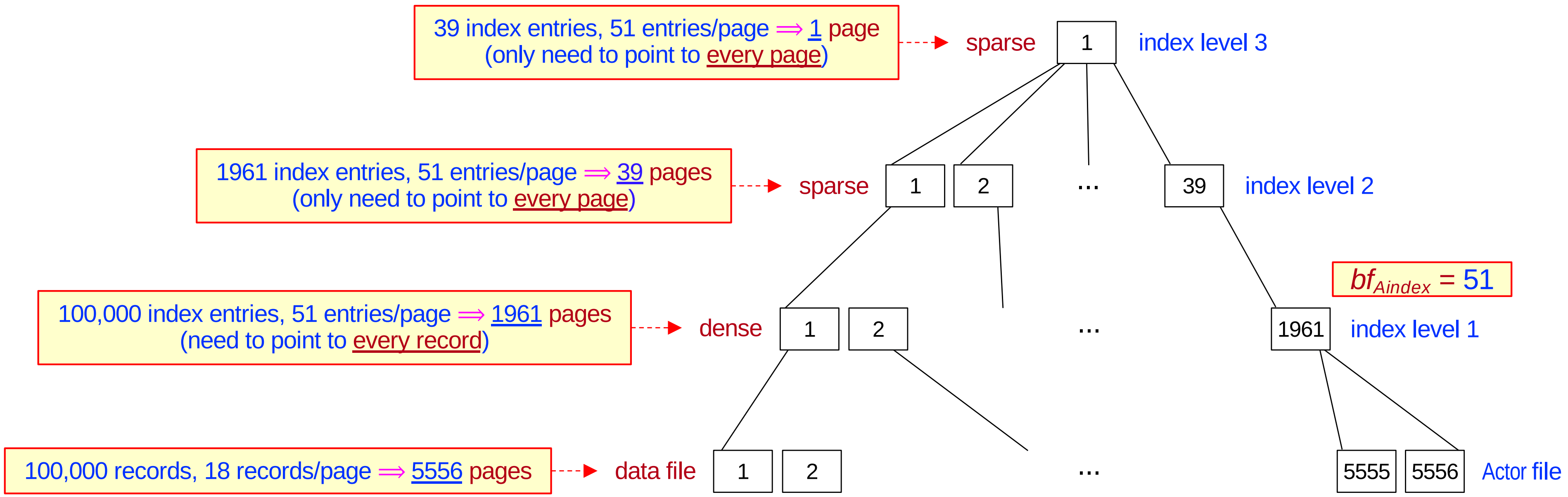
Film records: 30,000
Actor records: 100,000
Page size: 512 bytes
Pointer size: 6 bytes
Film record size: 84 bytes; $bf_F = 6$
Actor record size: 28 bytes; $bf_A = 18$

e) If the single-level index is converted into a multi-level index, how many levels are needed (assuming full pages)? (Briefly explain your answer.)

Index levels: 3

Explanation:

Actor multi-level index



$bf = \lfloor \text{\# bytes per page / \# bytes per record} \rfloor$
$\text{\# pages} = \lceil \text{\# records} / bf_r \rceil$

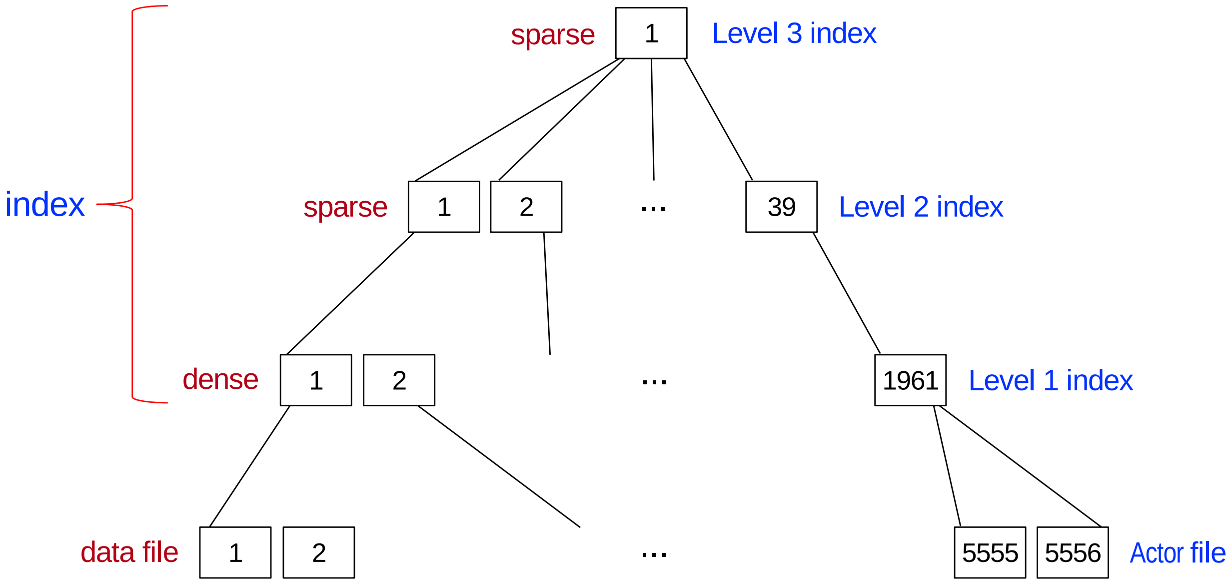
Film records: 30,000
Actor records: 100,000
Page size: 512 bytes
Pointer size: 6 bytes
Film record size: 84 bytes; $bf_F = 6$
Actor record size: 28 bytes; $bf_A = 18$

EXERCISE 2 (CONT'D)

f) Using the multi-level index, what is the page I/O cost of answering the query "Find the actor with id 100"?

Page I/O cost: 4 page I/Os **Why?**

Explanation: 3 page I/Os for the index plus 1 page I/O to retrieve the record.



EXERCISE 3

A company database has the following file and sizes of each field

Employee(employeeId: 6 bytes, employeeName: 10 bytes, departmentId: 4 bytes)

where departmentId is the id of the department where the employee works.

There are 100,000 employee records.

There are 1,000 departments (each department has 100 employees).

A page is 1,000 bytes.

A pointer is 4 bytes.

Assume that the file is *ordered on departmentId* and there is *no index*.

$bf = \lfloor$# bytes per page / # bytes per record$\rfloor$
pages = $\lceil$# records / $bf_r\rceil$

Employee records: 100,000
Departments: 1,000
Page size: 1000 bytes
Record size: 20 bytes
Pointer size: 4 bytes

EXERCISE 3 (CONT'D)

a) What is the blocking factor for the Employee file?

$bf_{Employee}$: $\lfloor$1000 bytes per page / 20 bytes per record$\rfloor$ = 50

b) How many pages are needed to store the Employee file?

Pages needed: $\lceil$100,000 records / 50 records per page$\rceil$ = 2000

EXERCISE 3 (CONT'D)

Employee records: 100,000
Departments: 1,000
Page size: 1000 bytes
Record size: 20 bytes
Pointer size: 4 byte
Pages: 2000

c) What is the page I/O cost for retrieving the records of all employees working in a department with a given departmentId (e.g., departmentId = 64)?

Page I/O cost: 12 page I/Os

Explanation: Finding the first record requires $\lceil \log_2 2000 \rceil = 11$ page I/Os plus 1 more page access to search the remaining records. Since each department has 100 employees and a page can hold 50 records, these records are distributed in at least 2 pages.

The answer $\lceil \log_2 2000 \rceil$ plus 2 can also be considered correct if a department's employee records are distributed across three pages (e.g., the first page contains 25 records – the second one 50 – and the third one 25). Total page I/Os = 13.

Employee records: 100,000
Departments: 1,000
Page size: 1000 bytes
Record size: 20 bytes
Pointer size: 4 byte
Pages: 2000

EXERCISE 4

For the Employee file of Exercise 3, assume we add a single-level ordered index on employeeId (6 bytes) where each entry has the form <employeeId, *pointer*> and the number of pointers is the same as the number of search keys.

a) How many index entries are needed?

Index entries needed: 100,000

Explanation: Since the file is ordered on departmentId, the index is secondary and therefore it must be dense. Thus, one index entry is needed for each employee.

$bf = \lfloor \#\text{ bytes per page} / \#\text{ bytes per record} \rfloor$
$\#\text{ pages} = \lceil \#\text{ records} / bf_r \rceil$

Employee records: 100,000
Departments: 1,000
Page size: 1000 bytes
Record size: 20 bytes
Pointer size: 4 byte
Pages: 2000

EXERCISE 4 (CONT'D)

b) How many pages are required for these index entries?

$bf_{employeeIdindex}$: $\lfloor$1000 bytes per page / 10 bytes per index entry$\rfloor$
= 100

Index pages: $\lceil$100,000 records / 100 index entries per page$\rceil$
= 1000

c) What is the page I/O cost of retrieving the record of an employee with a given employeeId?

Page I/O cost: $\lfloor$1000 bytes per page / 10 bytes per index entry$\rfloor$
= 100

EXERCISE 4 (CONT'D)

Employee records: 100,000
Departments: 1,000
Page size: 1000 bytes
Record size: 20 bytes
Pointer size: 4 byte
Pages: 2000

d) If the single-level index is converted into a multi-level index, how many levels are needed (assuming full pages)?

Levels needed: 3

Explanation: At the next level we index 1000 index pages (i.e., the index contains $\lceil$1000 pages / 100 index entries per page$\rceil$ = 10 pages. We also need an additional top level with 1 page.