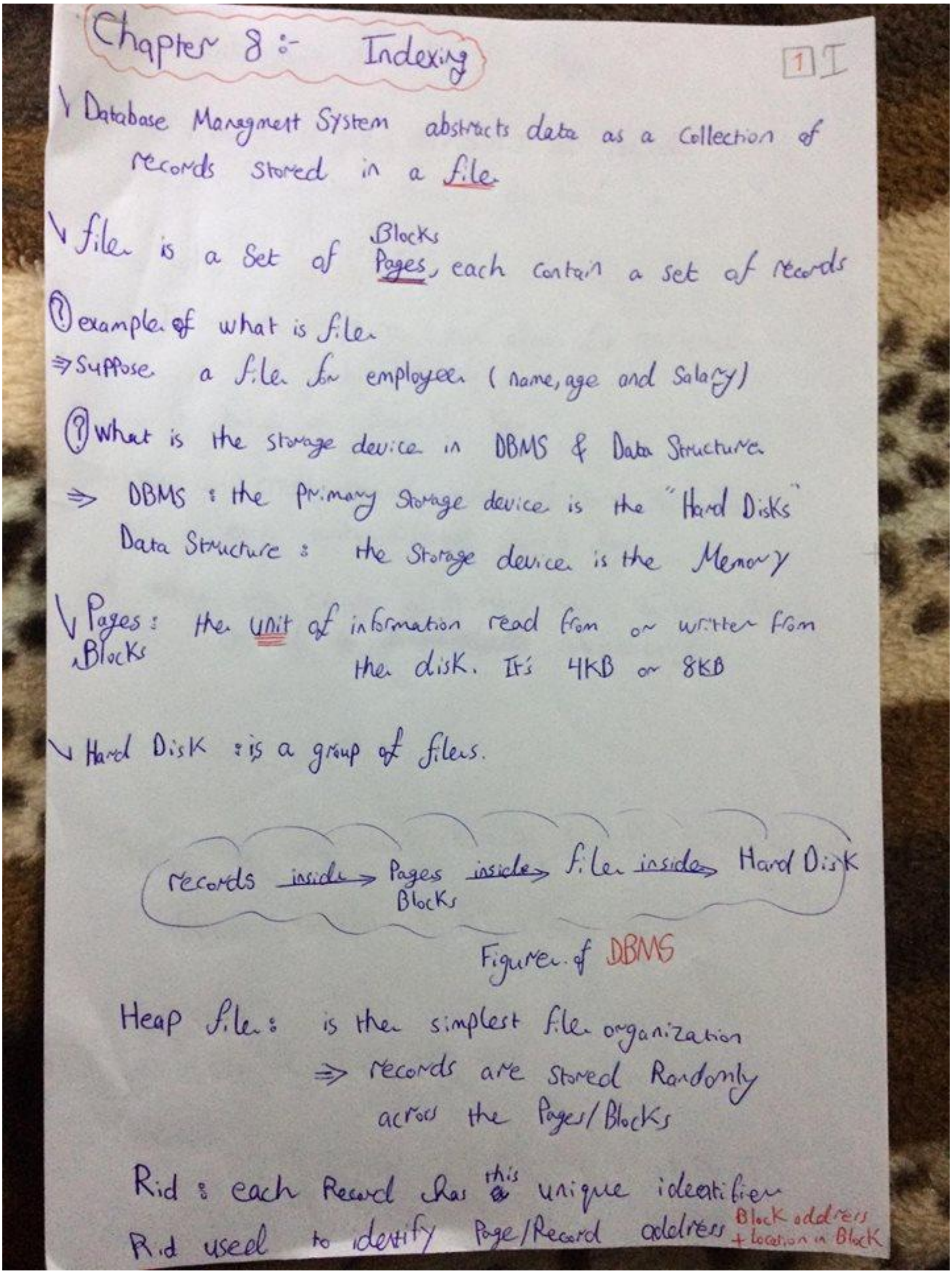


Chapter 8 :- Indexing

1 I

√ Database Managment System abstracts data as a Collection of records stored in a file.

√ file is a Set of $\overset{\text{Blocks}}{\text{Pages}}$, each contain a set of records

? example of what is file

⇒ Suppose a file for employee (name, age and Salary)

? what is the storage device in DBMS & Data Structure

⇒ DBMS : the primary storage device is the "Hard Disks"

Data Structure : the storage device is the Memory

√ Pages : the unit of information read from or written from
Blocks the disk. It's 4KB or 8KB

√ Hard Disk : is a group of files.

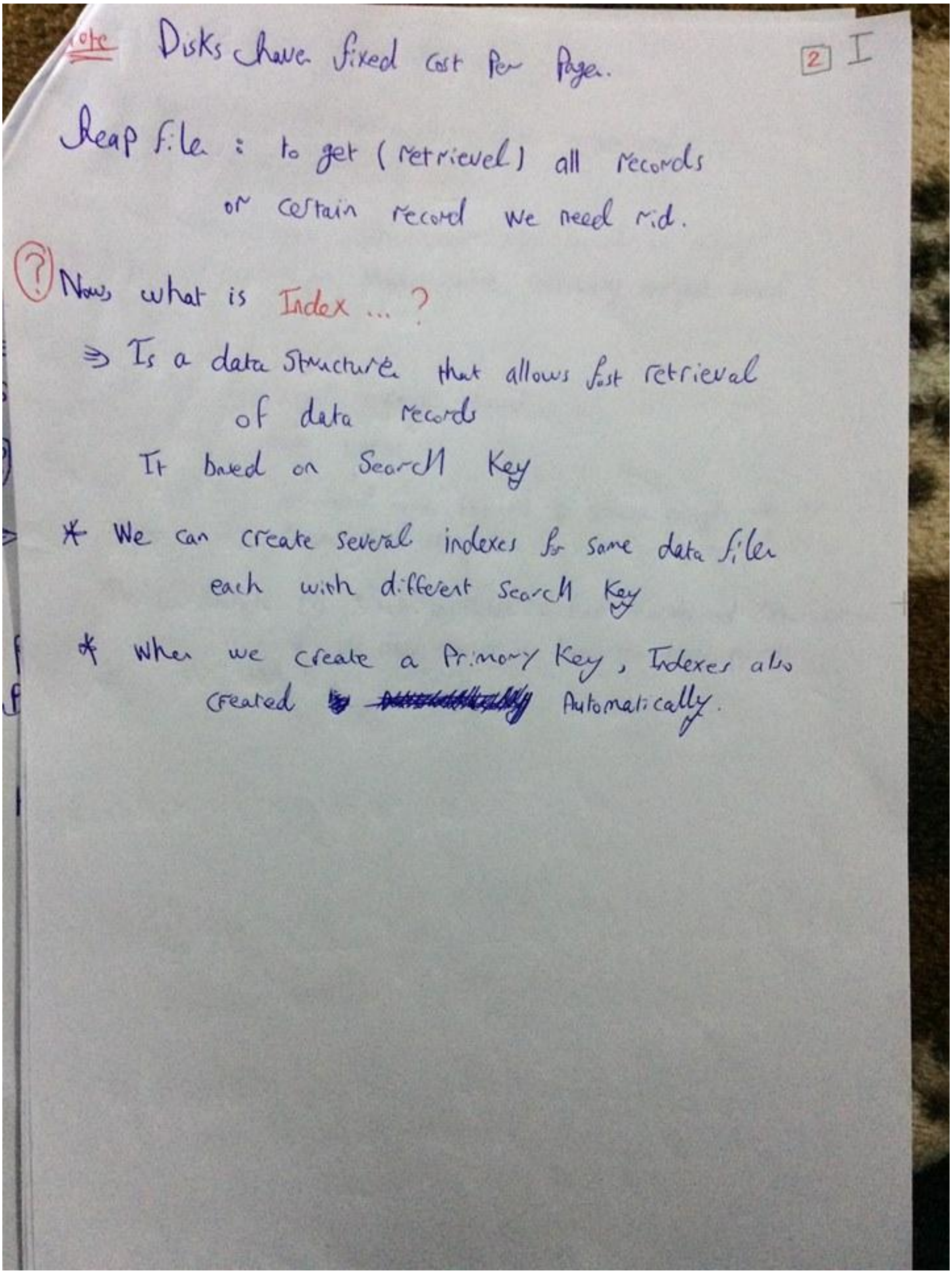
records $\xrightarrow{\text{inside}}$ Pages Blocks $\xrightarrow{\text{inside}}$ file $\xrightarrow{\text{inside}}$ Hard Disk

Figure of DBMS

Heap file : is the simplest file organization
⇒ records are stored Randomly across the Pages/Blocks

Rid : each Record has this unique identifier

Rid used to identify Page/Record address Block address + location in Block



ote Disks have fixed cost per Page.

2 I

Heap file : to get (retrievel) all records or certain record we need rid.

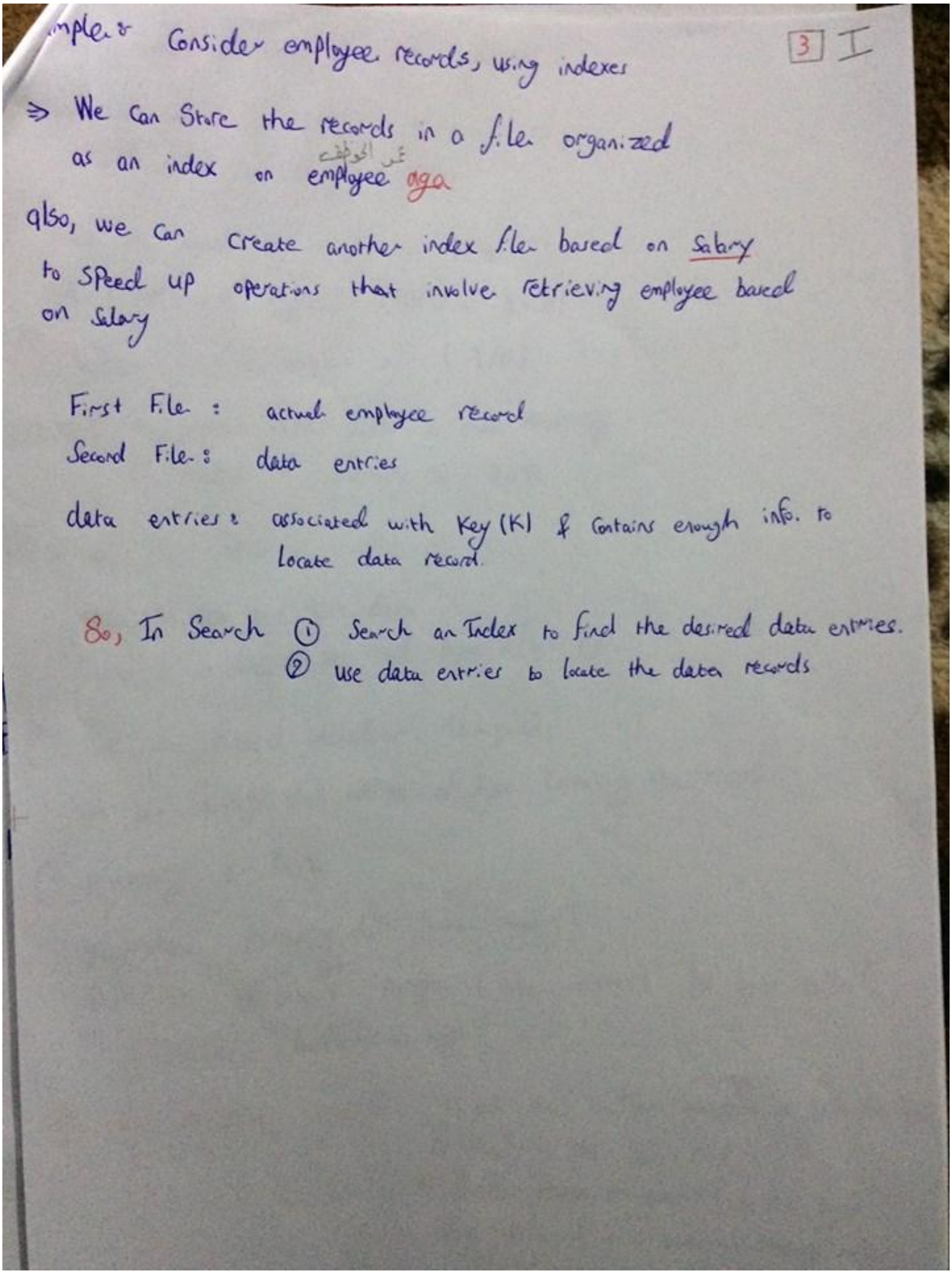
? Now, what is Index ... ?

⇒ Is a data Structure that allows fast retrieval of data records

It based on Search Key

* We can create several indexes for same data files each with different Search Key

* When we create a Primary Key, Indexes also created Automatically.



mple : Consider employee records, using indexes

3 I

⇒ We Can Store the records in a file organized as an index on employee عمر الموظف aga

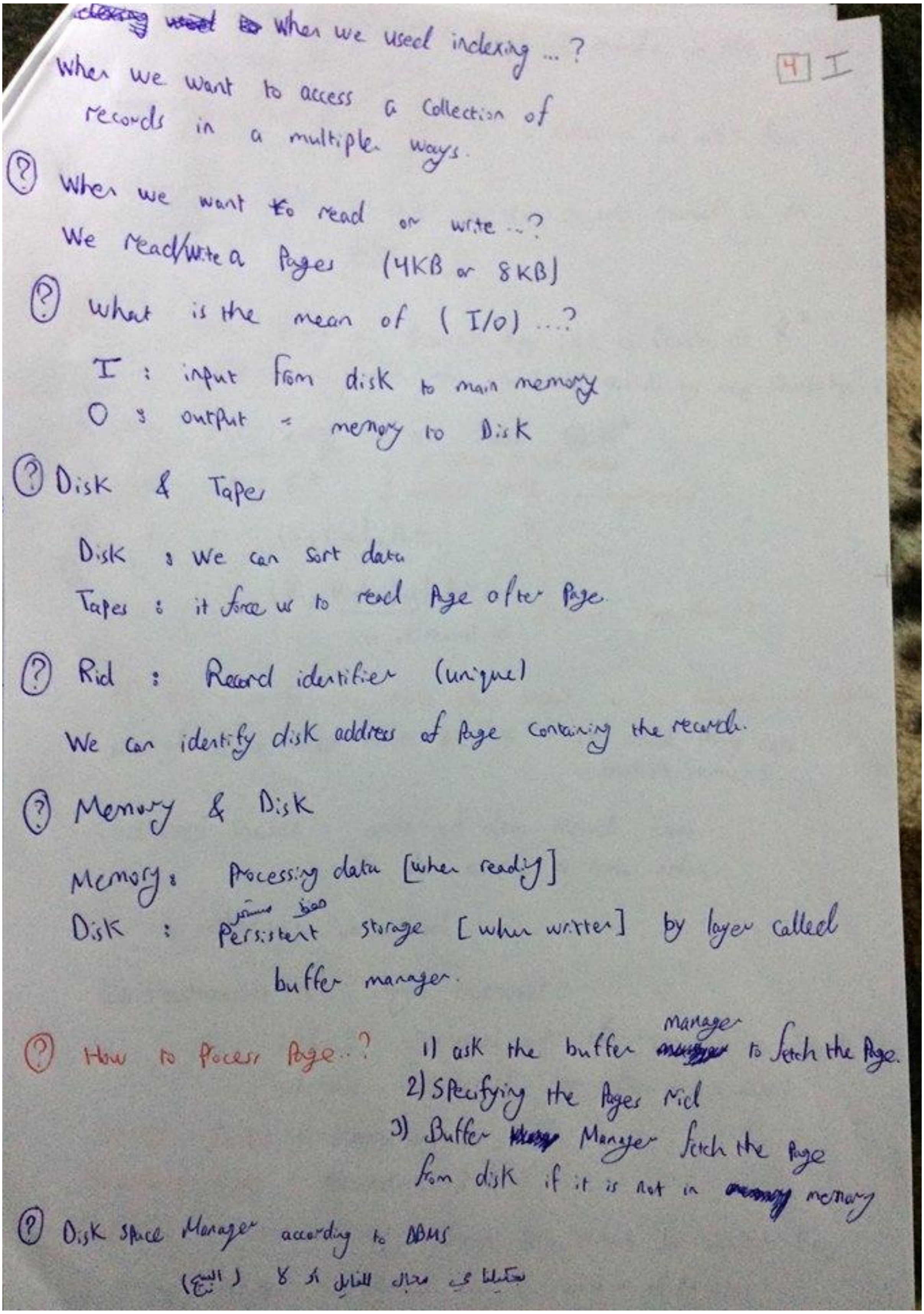
also, we can create another index file based on Salary to Speed up operations that involve retrieving employee based on Salary

First File : actual employee record

Second File : data entries

data entries : associated with Key (K) & Contains enough info. to Locate data record.

So, In Search ① Search an Index to find the desired data entries.
② use data entries to locate the data records



~~indexing used~~ ~~to~~ When we used indexing ...?

4 I

When we want to access a Collection of records in a multiple ways.

? When we want to read or write ..?

We read/write a Pages (4KB or 8KB)

? what is the mean of (I/O) ...?

I : input from disk to main memory

O : output = memory to Disk

? Disk & Tapes

Disk : We can sort data

Tapes : it force us to read Page after Page.

? Rid : Record identifier (unique)

We can identify disk address of Page containing the record.

? Memory & Disk

Memory : Processing data [when ready]

Disk : Persistent (حفظ مستمر) storage [when written] by layer called buffer manager.

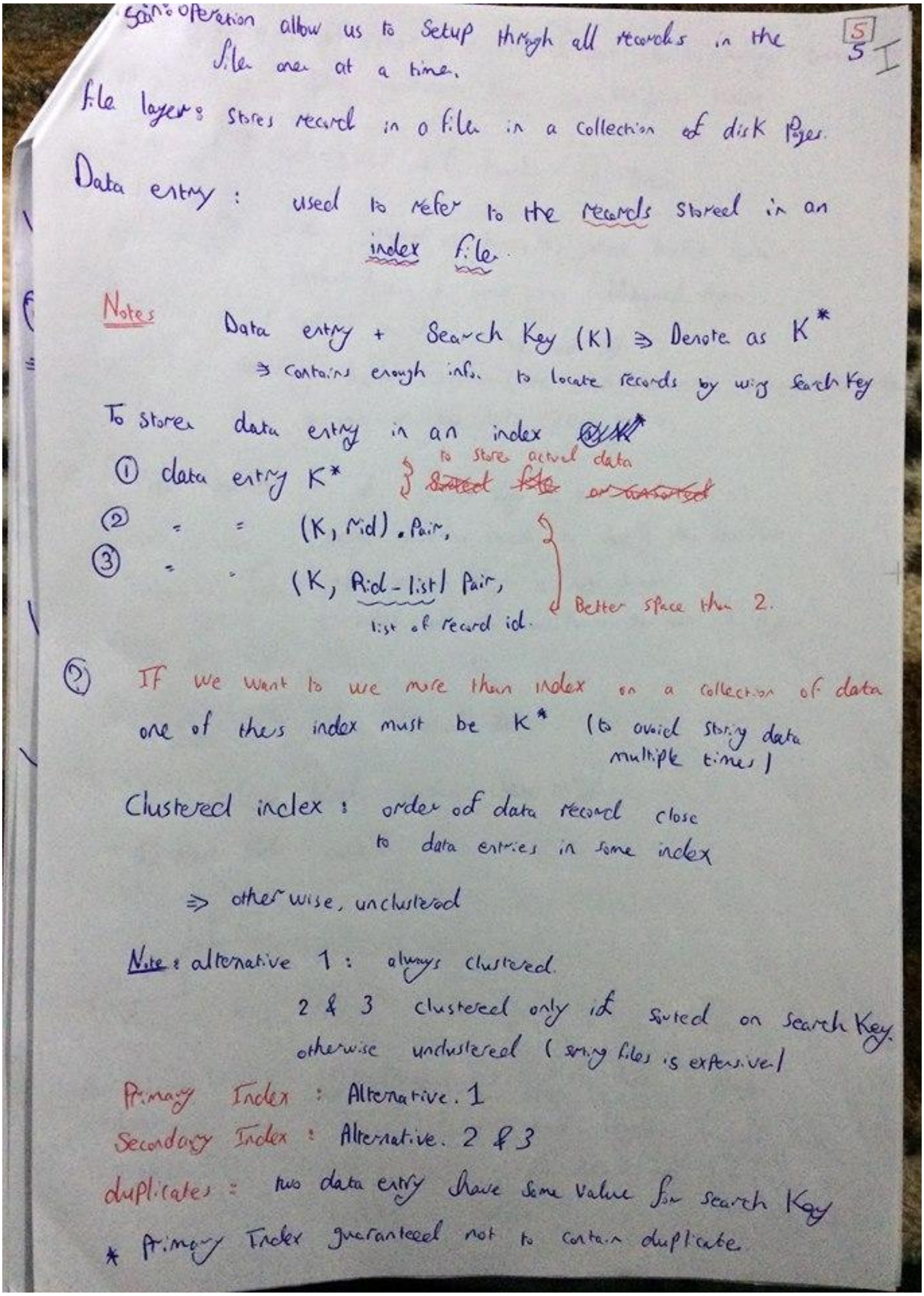
? How to Process Page..? 1) ask the buffer ~~manager~~ manager to fetch the Page.

2) Specifying the Pages Rid

3) Buffer ~~Manager~~ Manager fetch the Page from disk if it is not in ~~memory~~ memory

? Disk Space Manager according to DBMS

متطلبات في مجال التحليل ك 8 (العين)



Scan: operation allow us to Setup through all records in the file one at a time.

$\frac{\boxed{5}}{5}$ I

file layer: stores record in a file in a collection of disk Pages.

Data entry : used to refer to the records stored in an index file.

Notes Data entry + Search Key (K) ⇒ Denote as K^*

⇒ contains enough info. to locate records by using Search Key

To store data entry in an index

① data entry K^* } to store actual data ~~Sorted file or unsorted~~

② " " (K, rid) . Pair,

③ " " (K, Rid-list) Pair, } Better space than 2.

list of record id.

② If we want to use more than index on a collection of data one of this index must be K^* (to avoid storing data multiple times)

Clustered index : order of data record close to data entries in some index

⇒ other wise, unclustered

Note : alternative 1 : always clustered.

2 & 3 clustered only if sorted on Search Key. otherwise unclustered (sorting files is expensive)

Primary Index : Alternative. 1

Secondary Index : Alternative. 2 & 3

duplicates : two data entry have Same Value for Search Key

* Primary Index guaranteed not to contain duplicate.

~~Secondary~~ Secondary ⇒ duplicate is exist, if not ⇒ Search Key contain Some Candidate Key ⇒ Unique index

6 I

Indexing Techniques : ① hash ② Tree.

Hash :⇒ files grouped as buckets, where buckets Contain a Primary Page. & Some times additional Page Linked in a chain

⇒ Using hash function to search Key we can determine Primary Page bucket in one or two disk I/Os

Hash Search : if Search Key is Known

⇒ Hash function used to identify the bucket

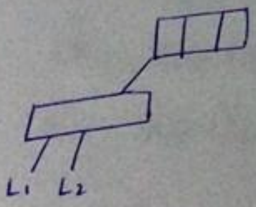
if Search Key is not Known & index on sth , we have to Scan all Pages

احنا بنعرف ي ها
الكاشر انو هيا
Known

Tree : Contain Sorted data

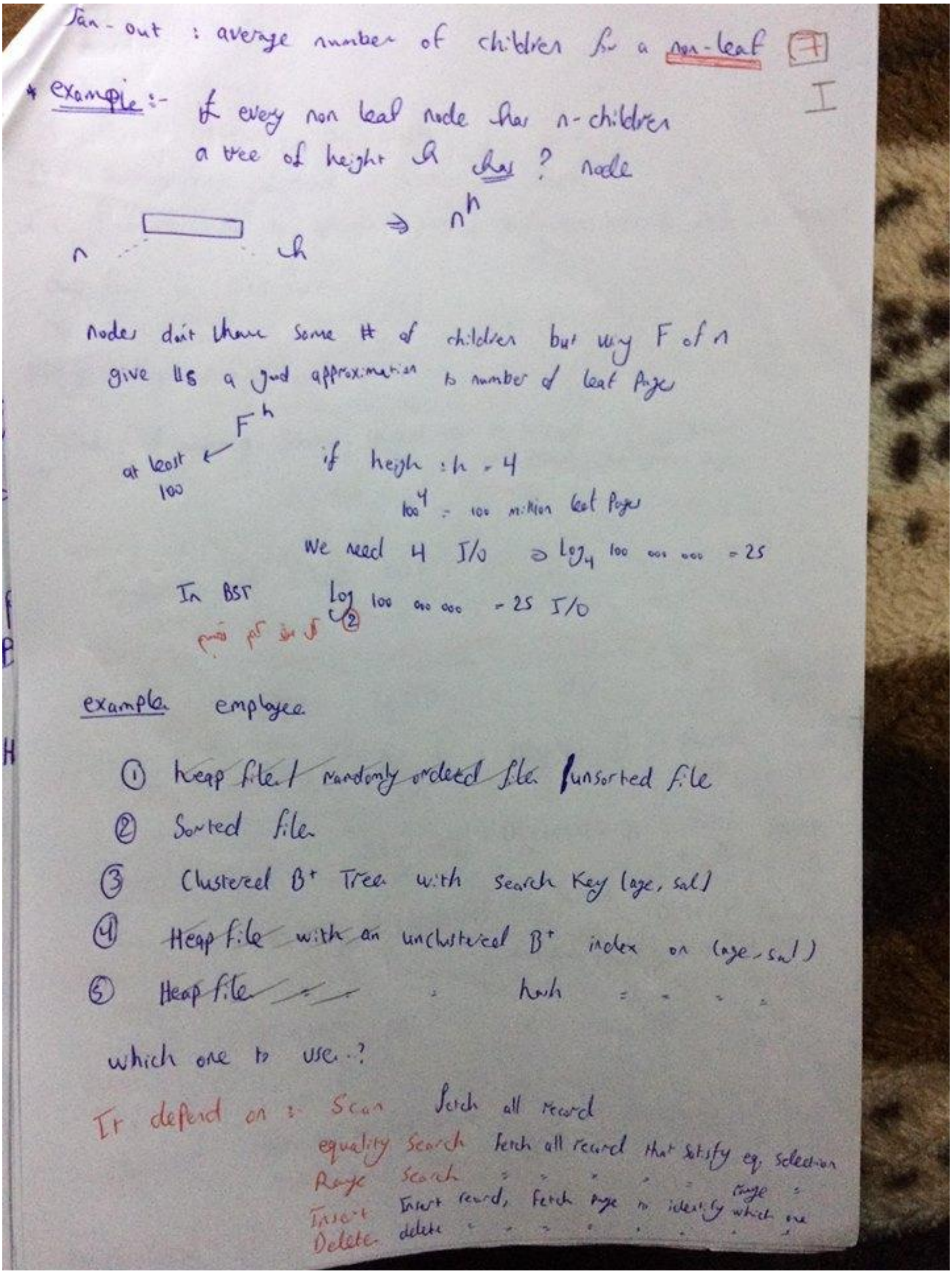
⇒ leaf level Contain data entry

Top Most Node called root



Next to L1 is L2
⇒ Pointer

B+ Tree : 1- More than 2 Nodes
2- Same length for all Path from root to leaf [Balanced height for all node]



Fan-out : average number of children for a non-leaf $\boxed{7}$

I

* example :- If every non leaf node has n-children
a tree of height h has ? node

n ... h ⇒ n^h

nodes don't have some # of children but using F of n
give us a good approximation to number of leaf Page

F^h
at least 100

if heigh : h = 4
100^4 = 100 million leaf Pages

We need 4 I/O ⇒ $\log_4 100\,000\,000 = 25$

In BST $\log_2 100\,000\,000 = 25$ I/O

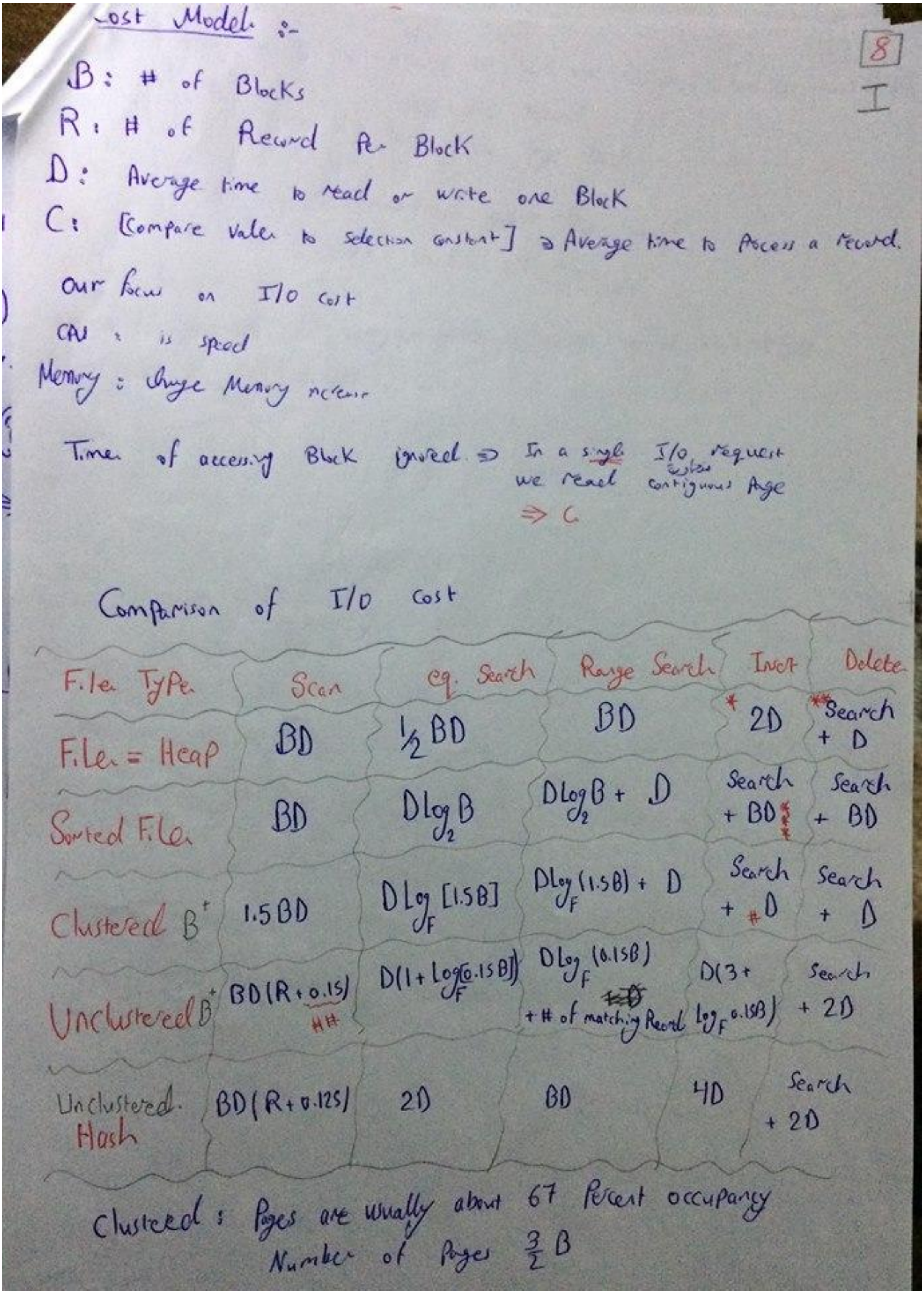
[illegible]

example employee

① heap file / randomly ordered file / unsorted file
② Sorted file
③ Clustered B^+ Tree with search Key (age, sal)
④ Heap file with an unclustered B^+ index on (age, sal)
⑤ Heap file " " " hash " " " "

which one to use..?

It depend on :- Scan fetch all record
equality Search fetch all record that satisfy eq. selection
Range Search " " " " " range "
Insert Insert record, fetch page to identify which one
Delete delete " " " " " " " "



Cost Model :-

B : # of Blocks

R : # of Record Per Block

D : Average time to read or write one Block

C : [Compare Value to selection constant] ⇒ Average time to Process a record.

Our focus on I/O cost

CPU : is speed

Memory : huge Memory increase

Time of accessing Block ignored ⇒ In a single I/O request we read contiguous Page
system

⇒ C

Comparison of I/O cost

File Type	Scan	eq. Search	Range Search	Insert	Delete
File = Heap	BD	$\frac{1}{2}$BD	BD	* 2D	** Search + D
Sorted File	BD	$D\log_2 B$	$D\log_2 B + D$	Search + BD ***	Search + BD
Clustered B^+	1.5BD	$D\log_F[1.5B]$	$D\log_F(1.5B) + D$	Search + # D	Search + D
Unclustered B^+	BD(R + 0.15) ##	$D(1 + \log_F[0.15B])$	$D\log_F(0.15B)$ + # of matching Record	$D(3 + \log_F 0.15B)$	Search + 2D
Unclustered. Hash	BD(R + 0.125)	2D	BD	4D	Search + 2D

Clustered : Pages are usually about 67 Percent occupancy
Number of Pages $\frac{3}{2}$B

9 I

: Record to be insert will be last one always (so last Page must fetch)
So the cost is 1- add the record
2- write the Page back ? assum we chage the mode ?

** : Search + write modified ~~back~~ Page back

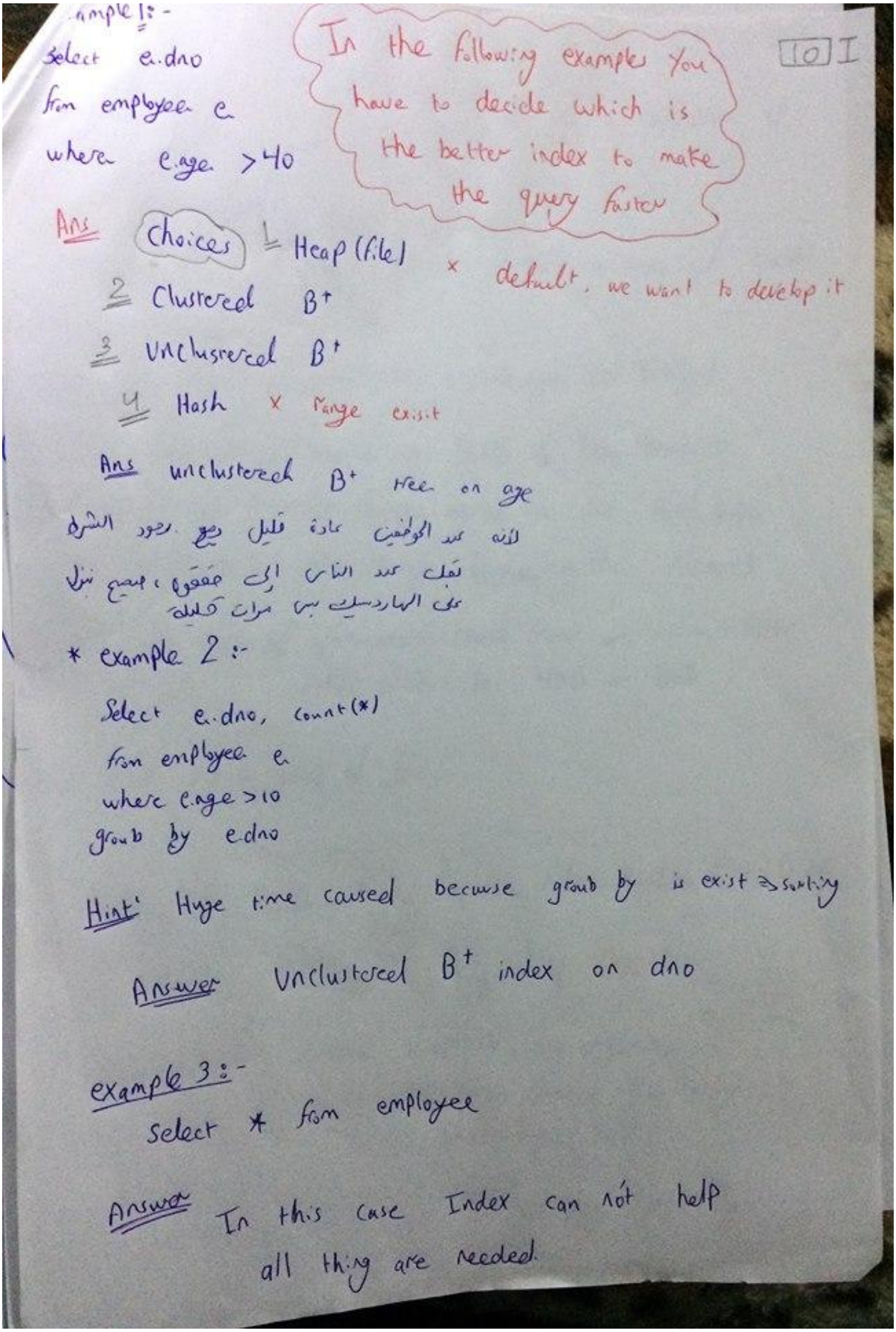
*** : BD after insert shift (rewrite...)

: D : write

: assumed data entry in index is tenth size of an employee

⇒ 0.1 (1.5B) = 0.15B

tenth ↙



-ample 1:-

In the following examples you have to decide which is the better index to make the query faster

10 I

Select e.dno
from employee e
where e.age >40

Ans (Choices) 1 Heap (file) × default, we want to develop it

2 Clustered B+

3 Unclusrered B+

4 Hash × range exist

Ans unclustered B+ tree on age

لأنه عدد الموظفين عادة قليل وبوجود الشرط

بقل عدد الناس الي بطلعو ، بصير ننزل

على الهاردسك بس مرات قليلة

* example 2 :-

Select e.dno, count(*)
from employee e
where e.age >10
groub by e.dno

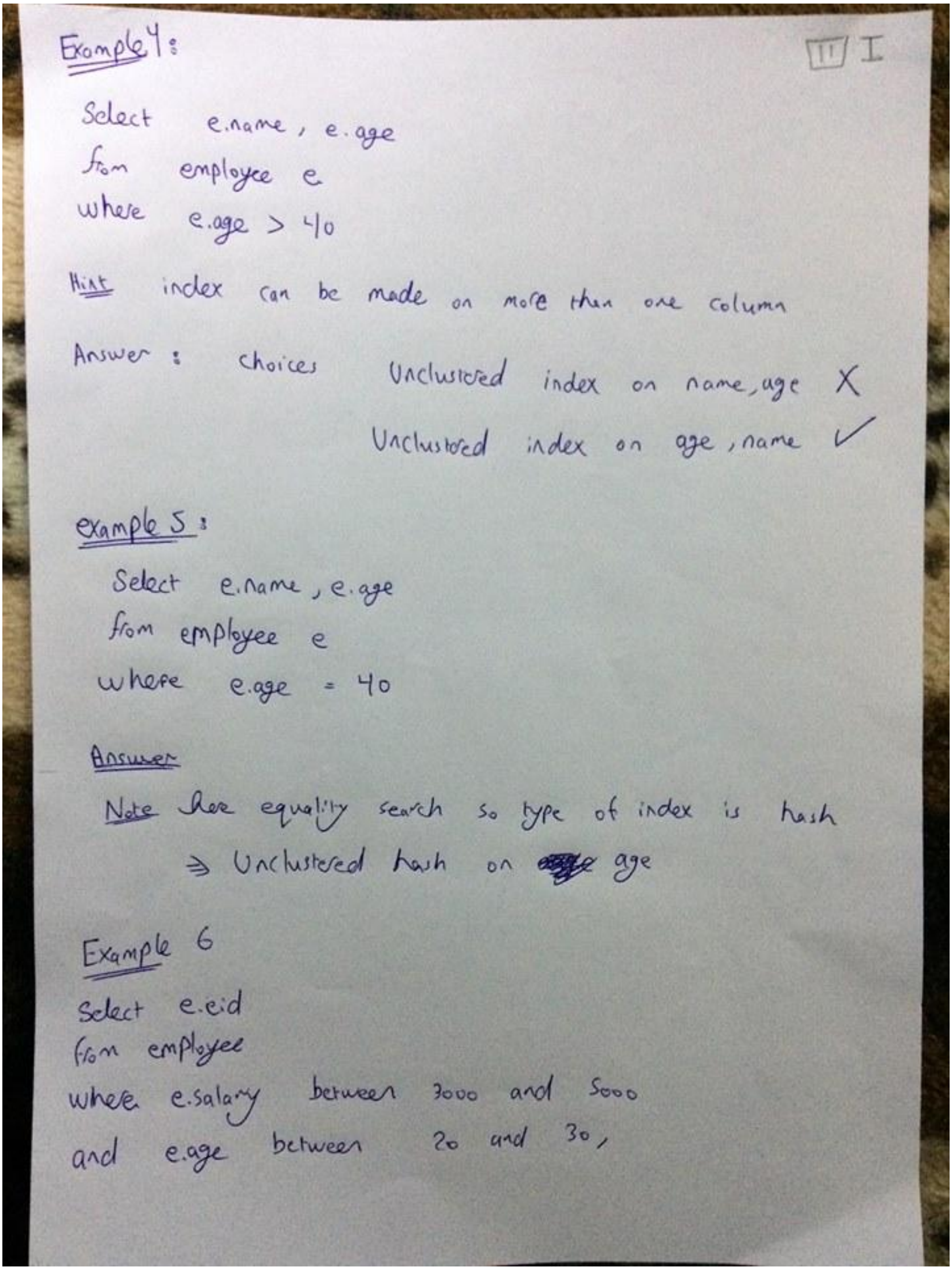
Hint: Huge time caused because groub by is exist => sorting

Answer Unclustered B+ index on dno

example 3 :-

Select * from employee

Answer In this case Index can not help all thing are needed.



Example 4:

```
Select  e.name , e.age
from    employee e
where   e.age > 40
```

Hint index can be made on more then one column

Answer : choices Unclustered index on name, age X

Unclustored index on age , name ✓

example 5 :

```
Select e.name , e.age
from employee e
where e.age = 40
```

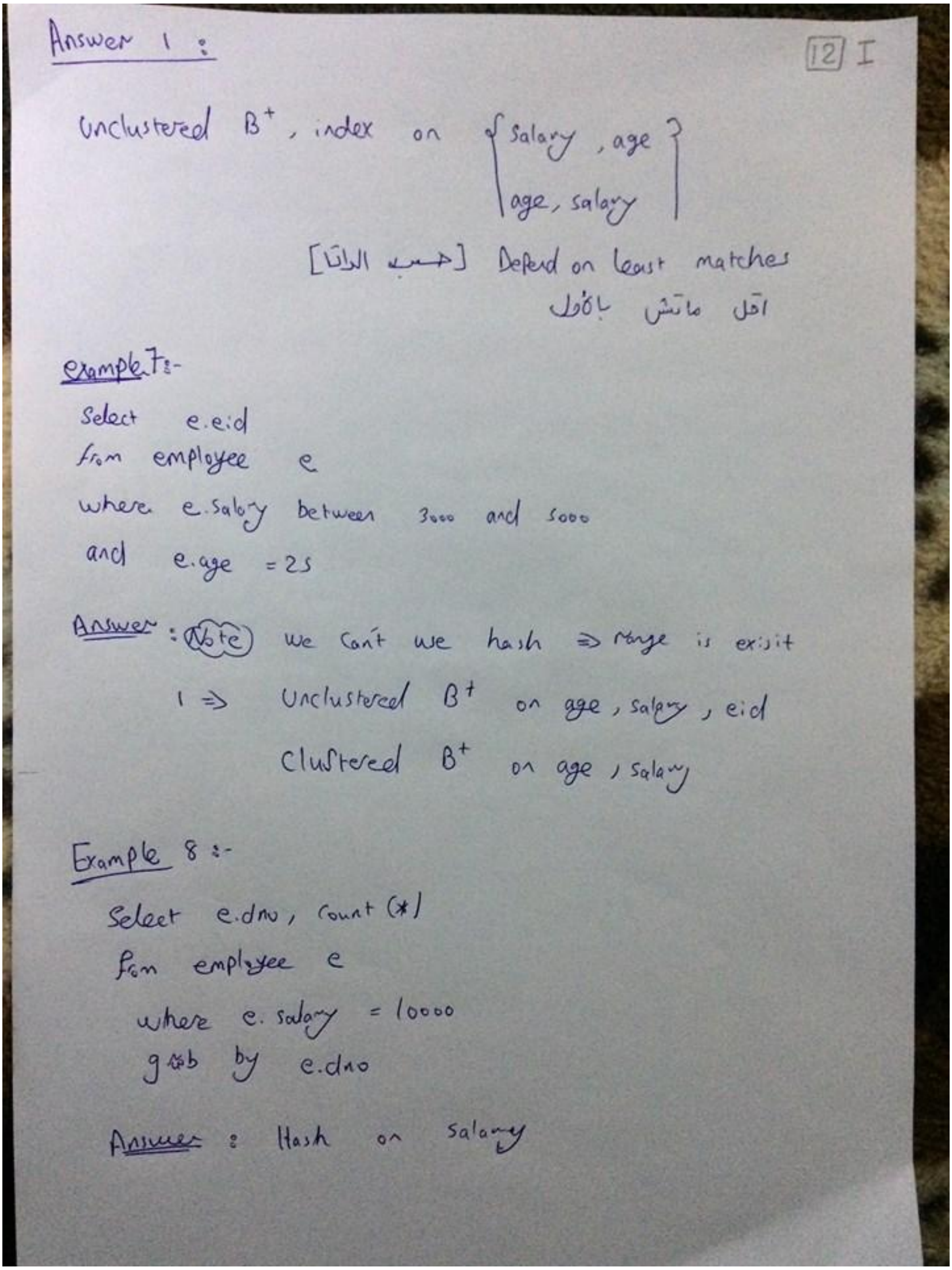
Answer

Note here equality search so type of index is hash

⇛ Unclustered hash on age

Example 6

```
Select e.eid
from employee
where e.salary between 3000 and 5000
and e.age between 20 and 30,
```



Answer 1 :

Unclustered B+, index on {salary , age / age, salary}

[حسب الداتا] Depend on least matches

اقل ماتش بالأول

example 7 :-

```
Select  e.eid
from  employee  e
where  e.salary  between  3000  and  5000
and  e.age  = 25
```

Answer : Note we can't we hash ⇒ range is exisit

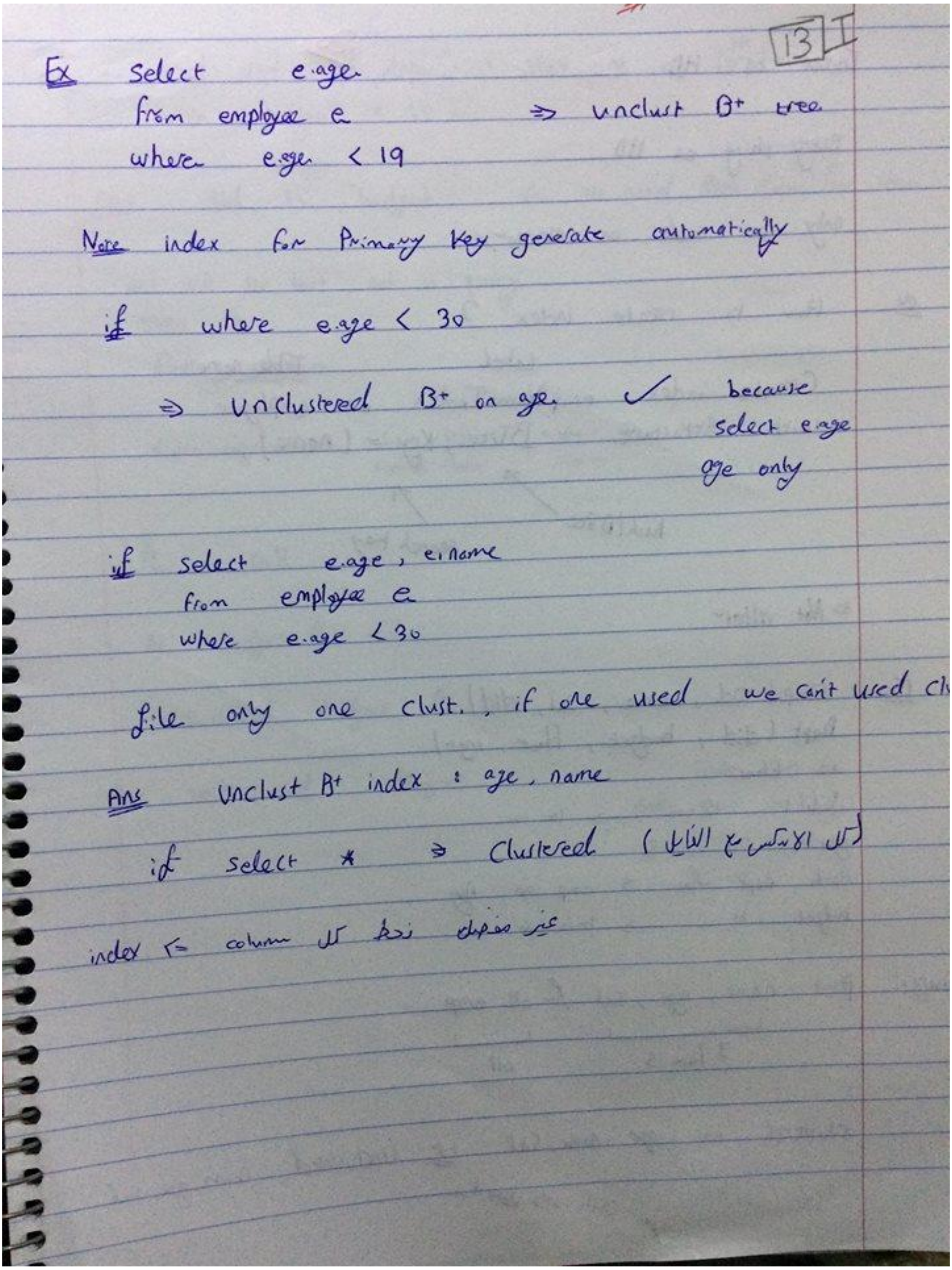
1 ⇒ Unclustered B+ on age , salary , eid

Clustered B+ on age , salary

Example 8 :-

```
Select  e.dno,  count (*)
from  employee  e
 where  e. salary  = 10000
 group by  e.dno
```

Answer : Hash on salary



Ex select e.age
from employee e ⇒ unclust B^+ tree
where e.age < 19

Note index for Primary Key generate automatically

if where e.age < 30

⇒ unclustered B^+ on age ✓ because select e.age age only

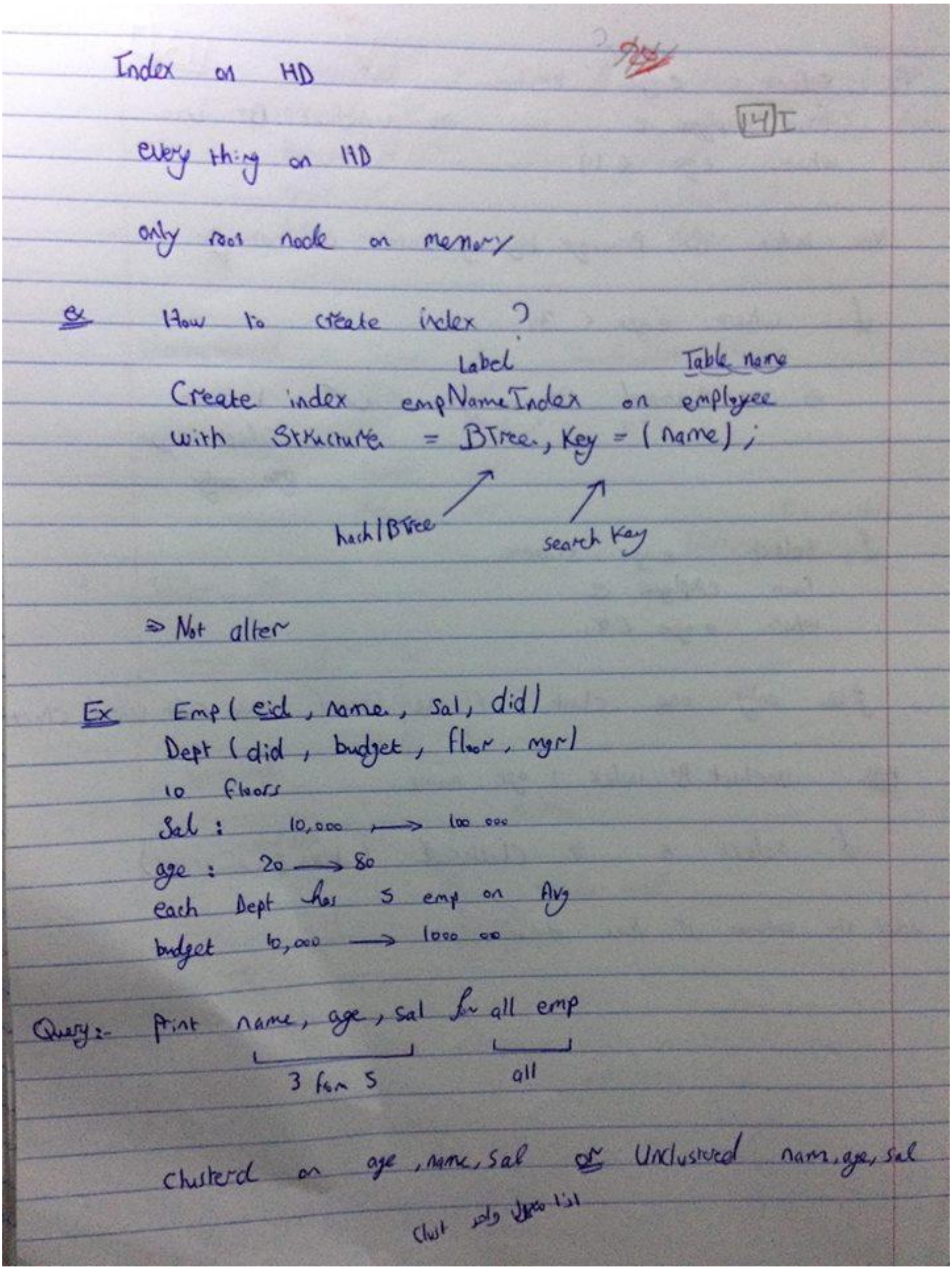
if select e.age, e.name
from employee e
where e.age < 30

file only one clust., if one used we can't used cl

Ans unclust B^+ index : age, name

if select * ⇒ Clustered (كل الاندكس مع الفايل)

index ≠ غير منطقي نحط كل column



Index on HD

$\boxed{14}$ I

every thing on HD

only root node on memory

Ex How to create index ?

```
                       Label                Table name
Create index   empNameIndex   on   employee
with  Structure  =  BTree,  Key  =  ( name ) ;
```

BTree ← hash/BTree

Key ← search Key

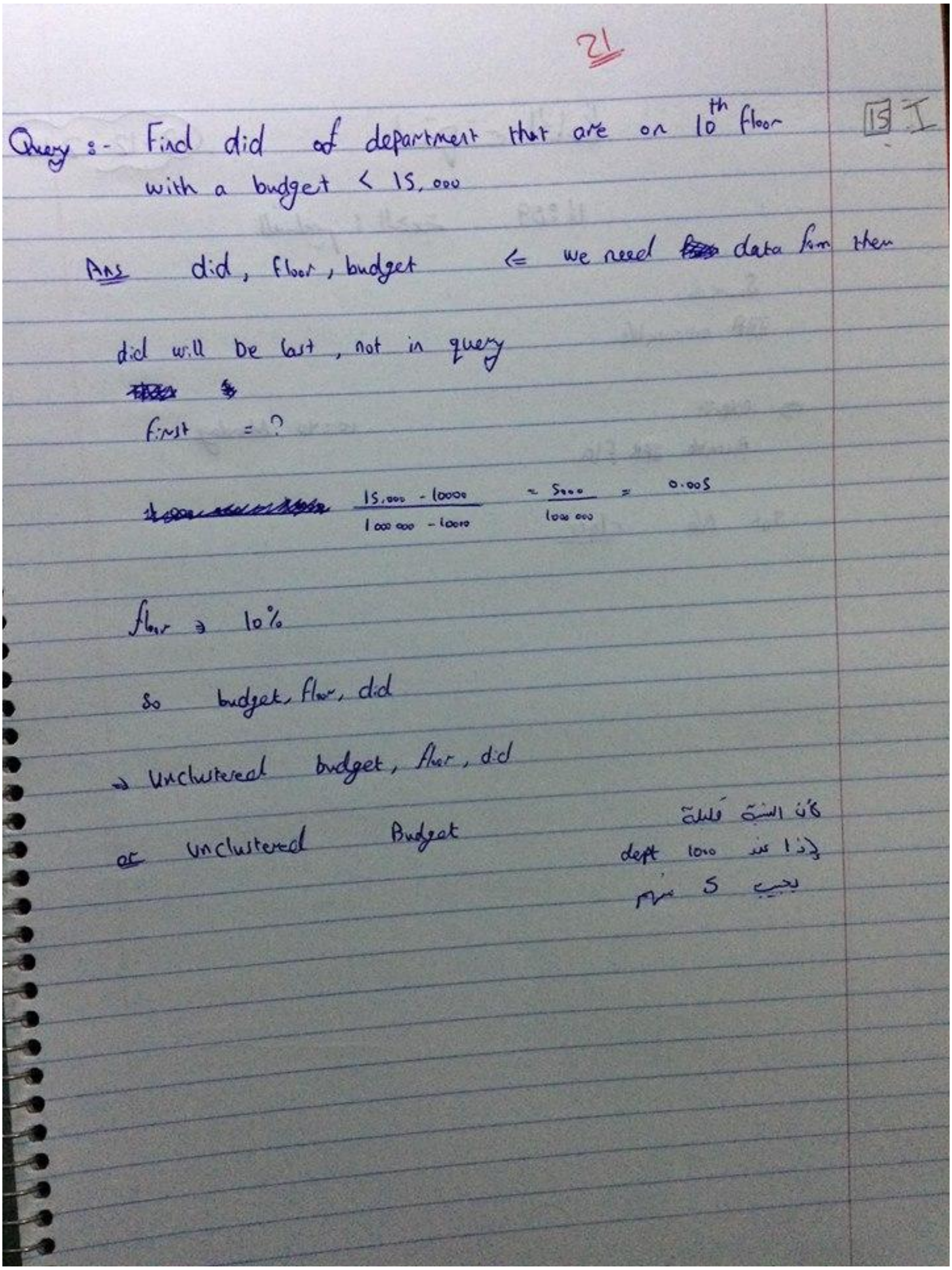
⇒ Not alter

Ex Emp (eid, name, sal, did)
Dept (did, budget, floor, mgr)
10 floors
Sal : 10,000 → 100 000
age : 20 → 80
each Dept has 5 emp on Avg
budget 10,000 → 1000 00

Query :- print $\underbrace{\text{name, age, sal}}_{\text{3 from 5}}$ for $\underbrace{\text{all emp}}_{\text{all}}$

Clusterd on age , name, Sal or Unclustered nam, age, sal

اذا معمول واحد Clust



21

$\boxed{15}$ I

Query :- Find did of department that are on 10th floor with a budget < 15,000

Ans did, floor, budget ⇐ we need data from them

did will be last, not in query

first = ?

$$\frac{15,000 - 10000}{1000000 - 10000} = \frac{5000}{1000000} = 0.005$$

floor ⇒ 10%

so budget, floor, did

⇒ Unclustered budget, floor, did

or unclustered Budget

لأن النسبة قليلة

إذا عند 1000 dept

بجيب 5 منهم