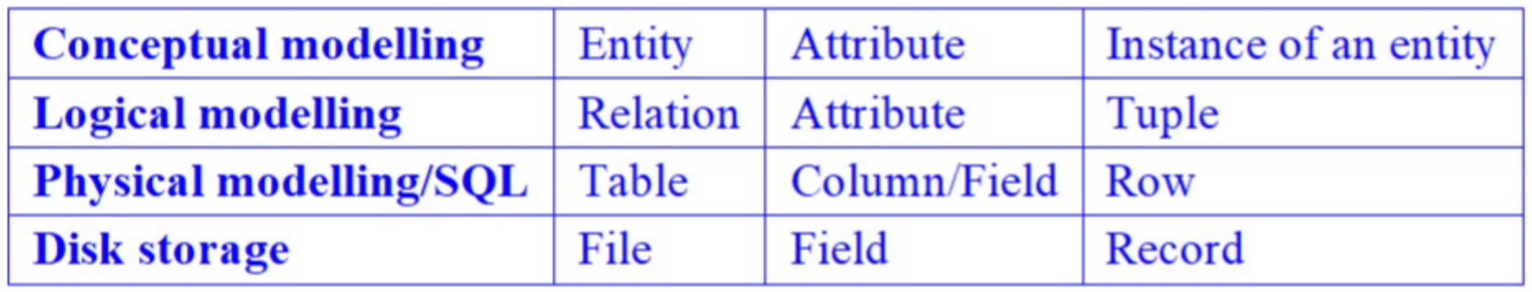
INFO20003 Database Systems

Week 6

Storage & Indexing

- DBMS store information on disks (normally hard disks)

- READ operation: transfer of data from the disk to main memory (RAM)
- WRITE operation: transfer data from RAM to the disk.
- High cost



Conceptual modelling	Entity	Attribute	Instance of an entity
Logical modelling	Relation	Attribute	Tuple
Physical modelling/SQL	Table	Column/Field	Row
Disk storage	File	Field	Record

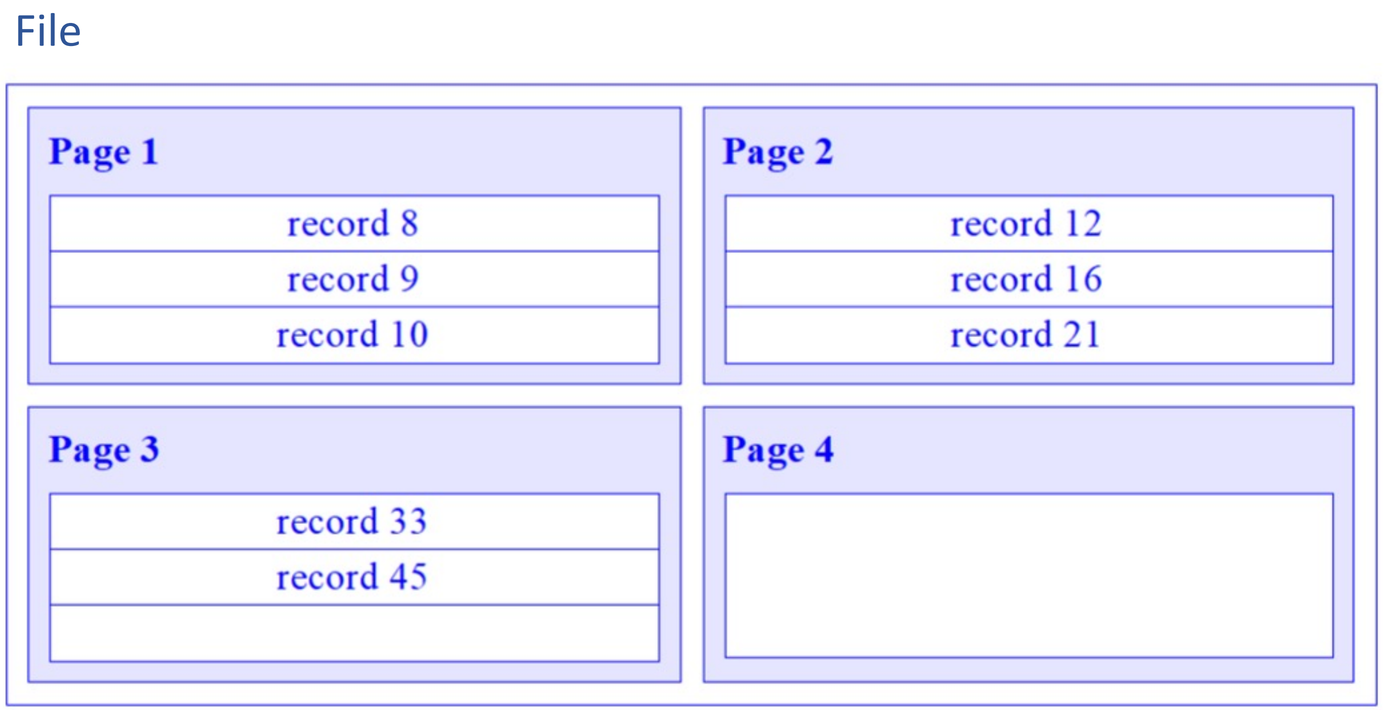
Files, pages and records

- Record: an individual row of a table and has an unique rid
  - rid: identify disk address of the record
  - (3, 7) = the seventh record on the third page

- Page: an allocation of space on disk or in memory containing a collection of records
  - Typically, every page is the same size

- File: consists of a collection of pages
  - In simple database scenarios, a file corresponds to a single table

Files, pages and records

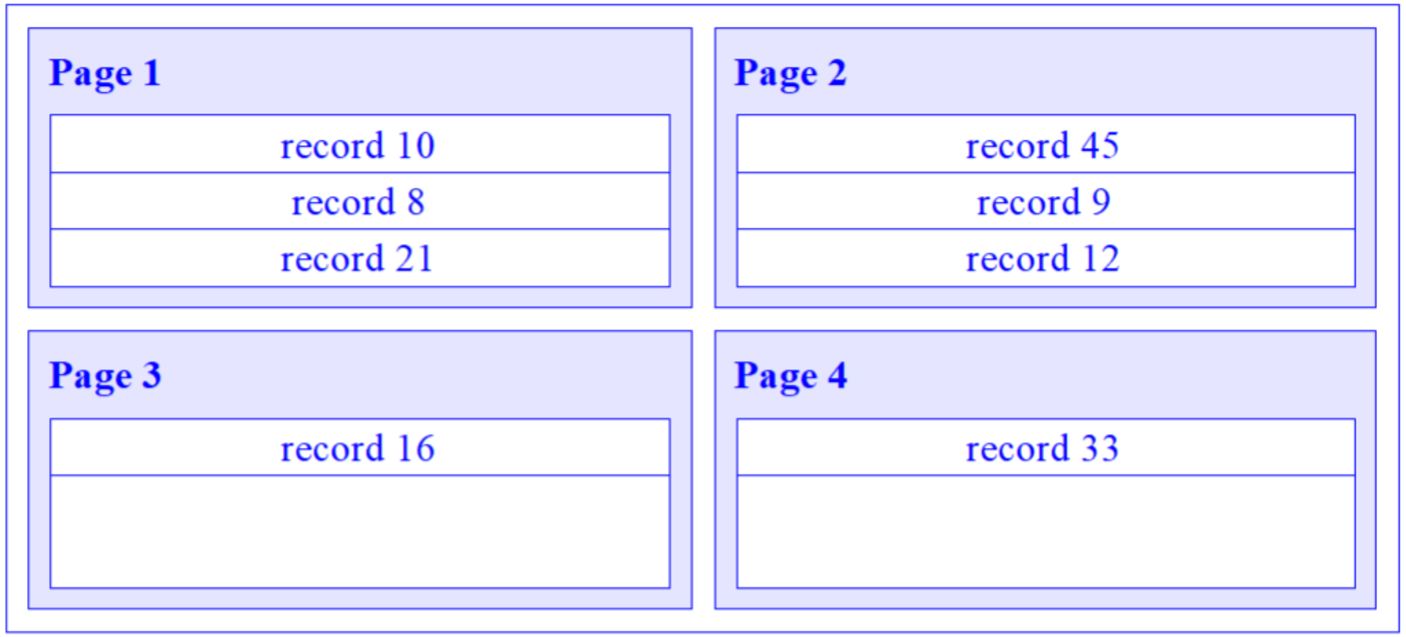


File organisation

- Heap file organisation
- Sorted file organisation
- Index file organisation

Heap File Organisation

- No ordering, sequencing or indexing



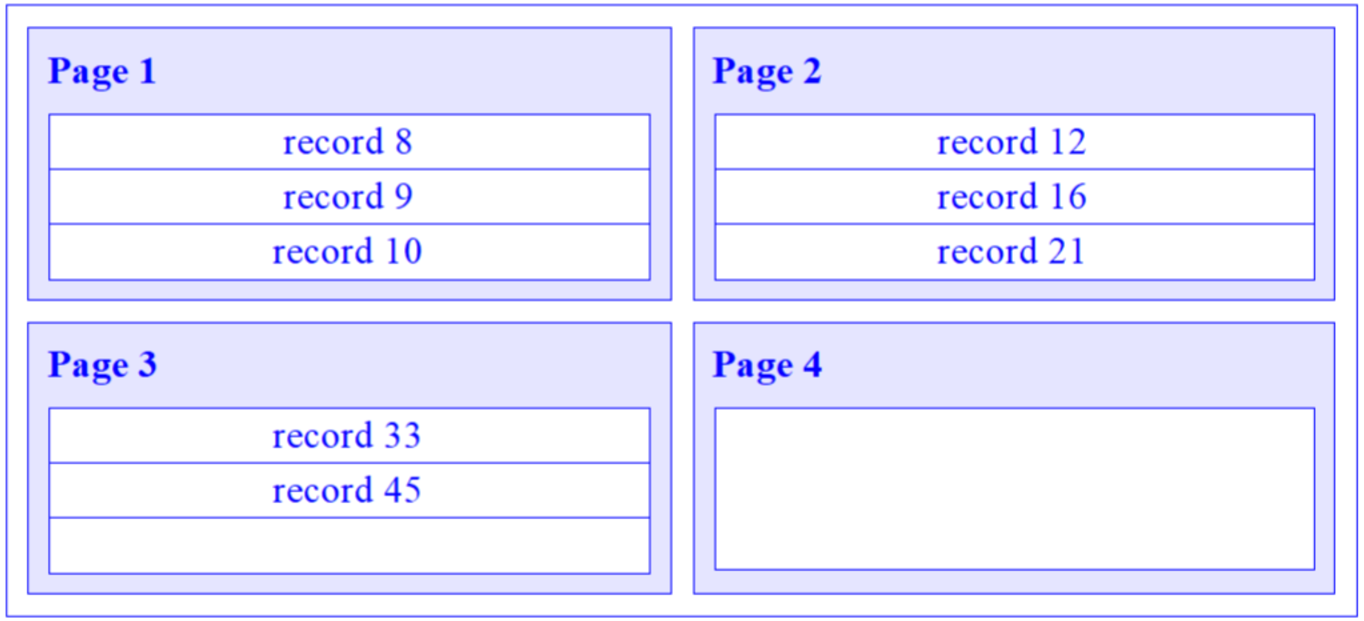
- Advantage: fast insertion & deletion
- Disadvantage: inefficient selection queries with conditions

File organisation

- Heap file organisation
- Sorted file organisation
- Index file organisation

Sorted File Organisation

- Records are ordered based on the search key



- Advantage: efficient range/equality query
- Disadvantage: inefficient insertion & deletion

File organisation

- Heap file organisation
- Sorted file organisation
- Index file organisation

Index File Organisation

- Index = a data structure built on top of data files
- Any subset of the fields of a table can be indexed based on queries that are frequently run against the database

Index

- speeds up selection on the search key field

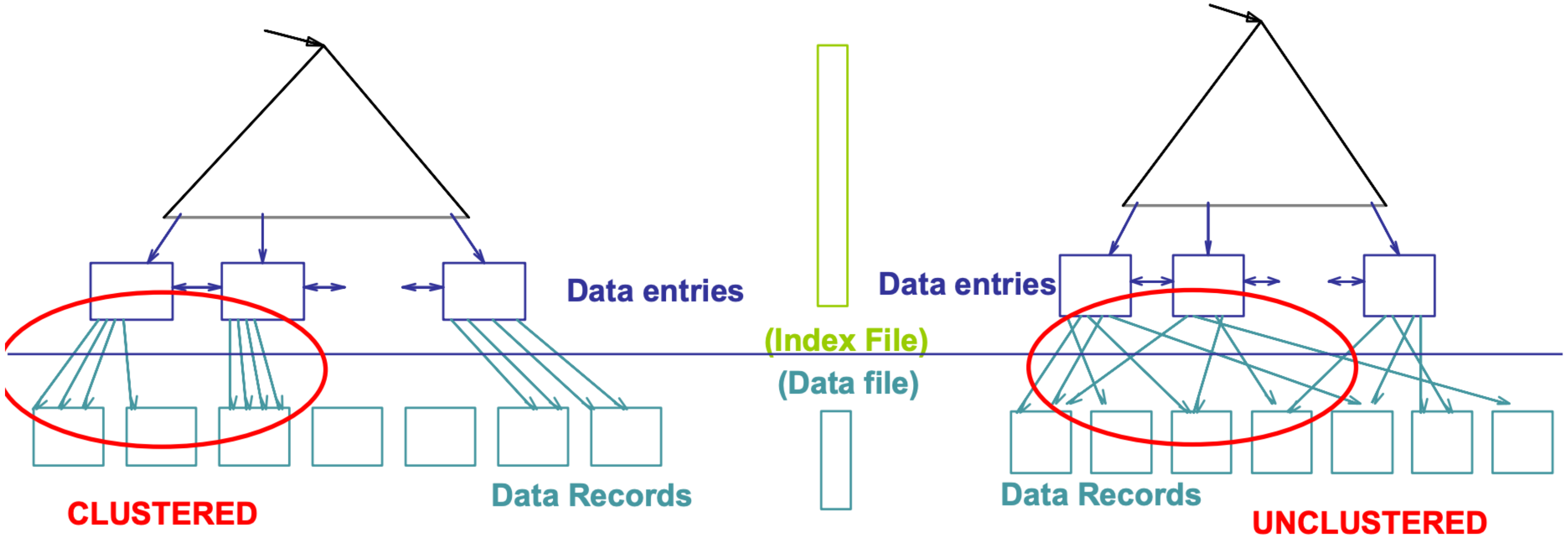
- Stored in an index file

- Made up of data entries, which refer back to the data in the relation
    - Data entries: (k, rid)
        - K = search key
        - Rid = record ID in the relation

- Index-only scan: an index scan without accessing data pages

Index Properties

- Clustered v.s. Unclustered
  - Whether the index & the data has the same ordering



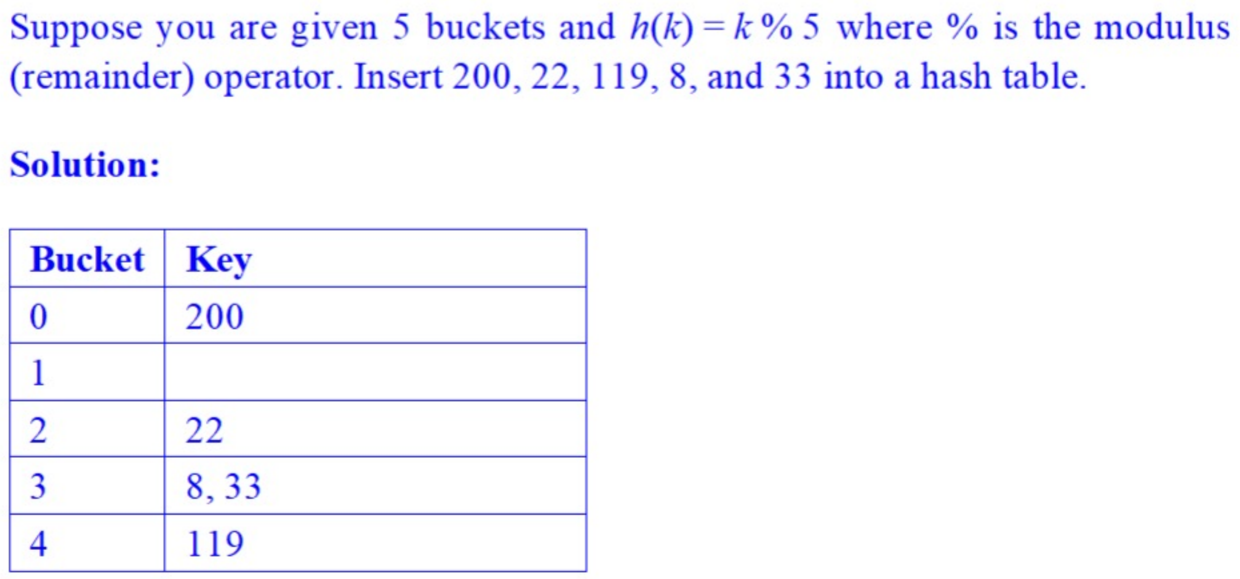
- Primary v.s. secondary
  - Whether the search key is the primary key of the relation

Hash-based indexing

- hash function:

  h(r.search_key) = bucket in which record r belongs

- Best for ***equality selections***



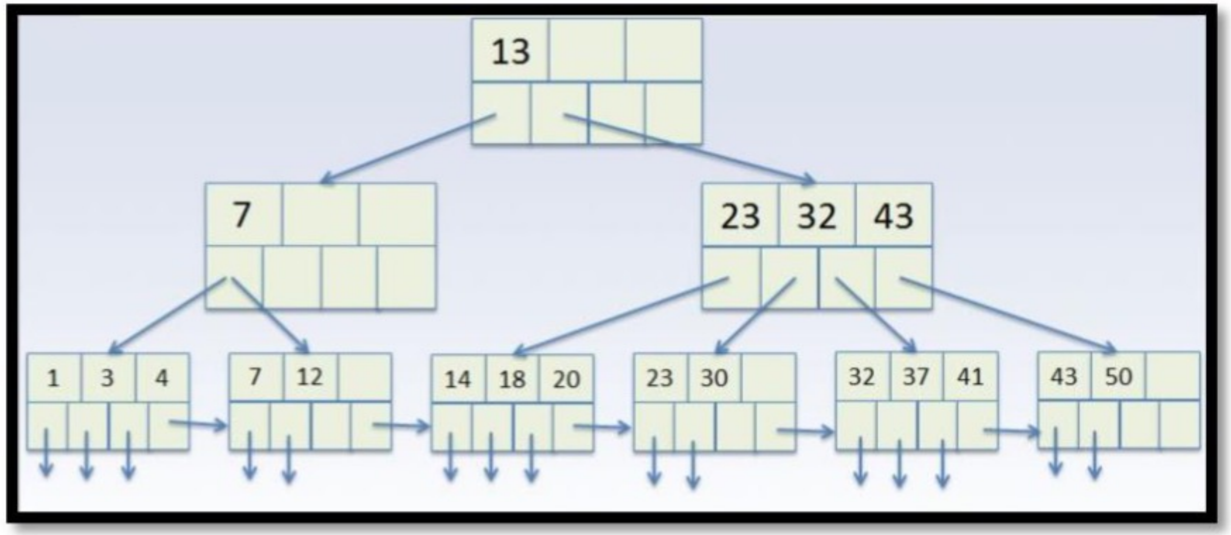
Suppose you are given 5 buckets and $h(k) = k \% 5$ where % is the modulus (remainder) operator. Insert 200, 22, 119, 8, and 33 into a hash table.

Solution:

Bucket	**Key**
0	200
1	
2	22
3	8, 33
4	119

B-tree indexing

- created by
    1. sorting the data on the search key
    2. maintaining a hierarchical search data structure (B+ tree)
- Insertion is costly
- Best for ***<u>Range selection</u>***



Q1) Choosing an index

- Primary vs. Secondary index
- Clustered vs. Unclustered index
- Hash vs. Tree index

Primary vs. Secondary index

Primary index

- Used when the records are retrieved based on the ***primary key values***
- Generally, a table should always have a primary index
  - (MySQL creates one automatically)

Secondary index

- Used when the fields are ***frequently used*** in the queries

Q1) Choosing an index

- Primary vs. Secondary index
- Clustered vs. Unclustered index
- Hash vs. Tree index

Clustered vs. Unclustered index

Clustered index

- preferred for frequently-executed ***range query***
  - choose the most frequently used combination
- More expensive to maintain

Unclustered index

- ***Equality query***
  - Especially when there is no duplicate on search key value

Q1) Choosing an index

- Primary vs. Secondary index
- Clustered vs. Unclustered index
- Hash vs. Tree index

Hash vs. Tree index

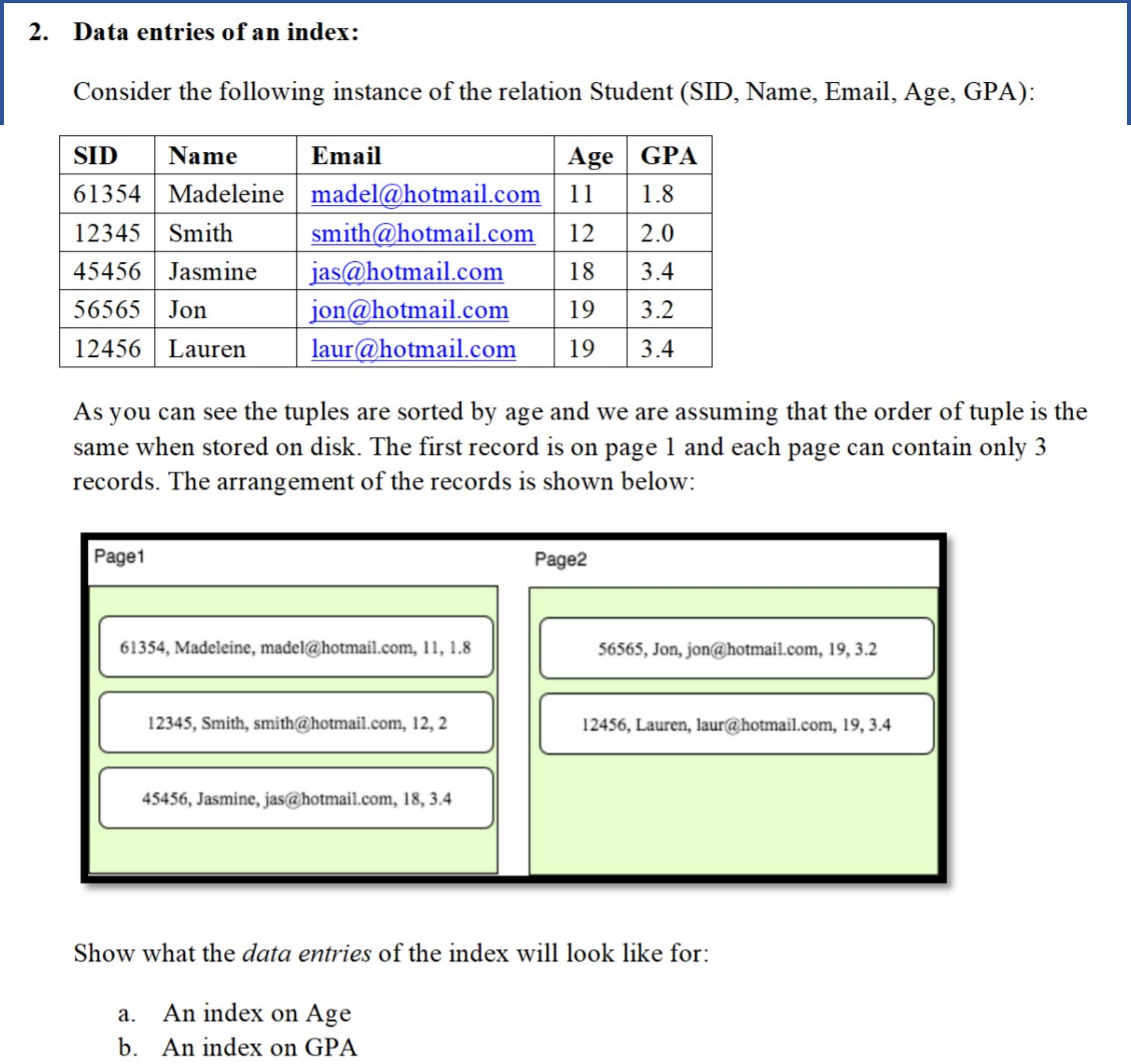
Tree index

- Preferred when performing ***range queries***

Hash index

- Preferred when performing ***equality queries***

Q2)

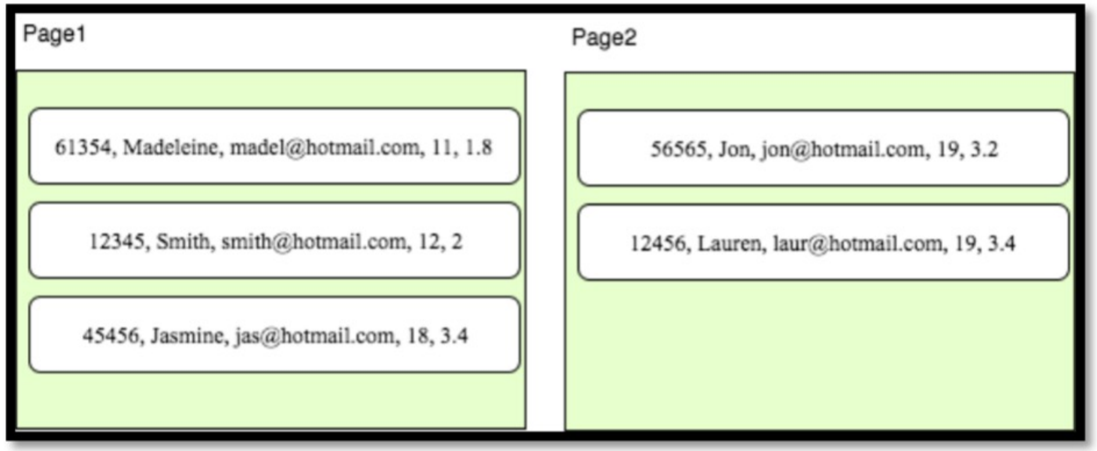


2. Data entries of an index:

Consider the following instance of the relation Student (SID, Name, Email, Age, GPA):

SID	Name	Email	Age	GPA
61354	Madeleine	madel@hotmail.com	11	1.8
12345	Smith	smith@hotmail.com	12	2.0
45456	Jasmine	jas@hotmail.com	18	3.4
56565	Jon	jon@hotmail.com	19	3.2
12456	Lauren	laur@hotmail.com	19	3.4

As you can see the tuples are sorted by age and we are assuming that the order of tuple is the same when stored on disk. The first record is on page 1 and each page can contain only 3 records. The arrangement of the records is shown below:

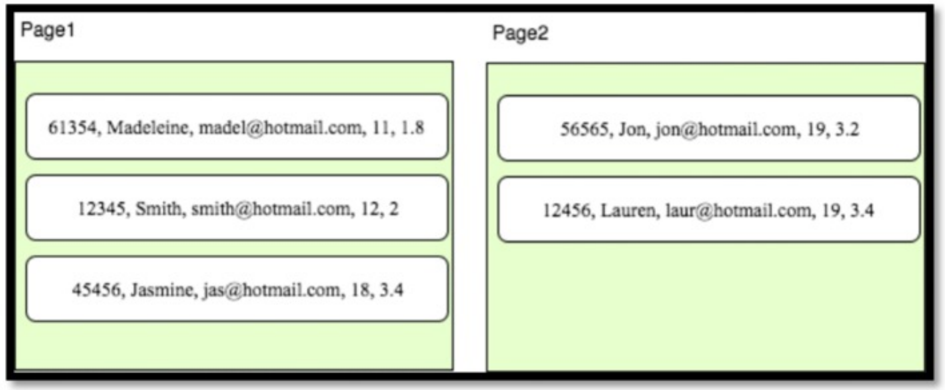

Page1
61354, Madeleine, madel@hotmail.com, 11, 1.8
12345, Smith, smith@hotmail.com, 12, 2
45456, Jasmine, jas@hotmail.com, 18, 3.4
Page2
56565, Jon, jon@hotmail.com, 19, 3.2
12456, Lauren, laur@hotmail.com, 19, 3.4

Show what the *data entries* of the index will look like for:

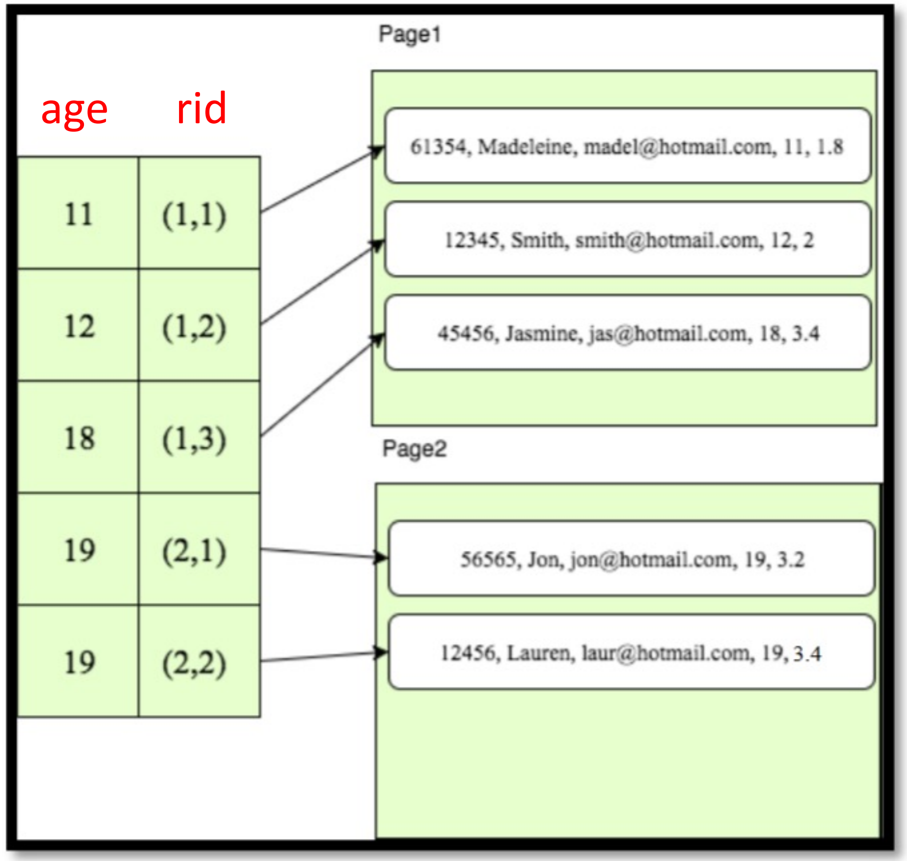
a. An index on Age
b. An index on GPA

Q2a) Age

- Data entry: (age, rid)
- Clustered index



Page1	Page2
61354, Madeleine, madel@hotmail.com, 11, 1.8	56565, Jon, jon@hotmail.com, 19, 3.2
12345, Smith, smith@hotmail.com, 12, 2	12456, Lauren, laur@hotmail.com, 19, 3.4
45456, Jasmine, jas@hotmail.com, 18, 3.4	



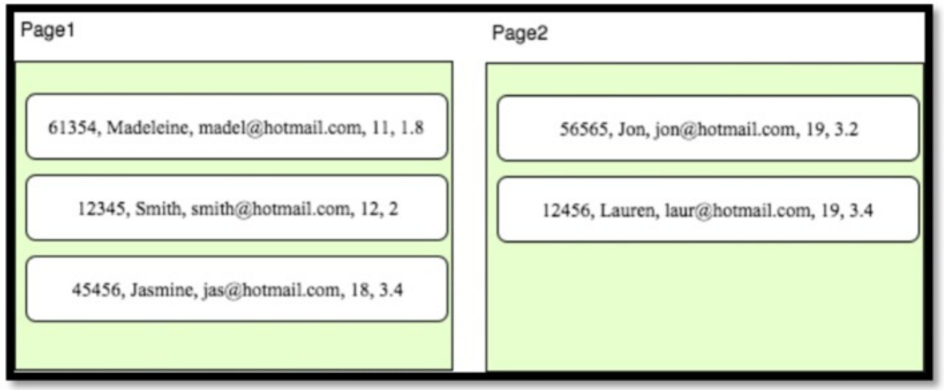
age	rid
11	(1,1)
12	(1,2)
18	(1,3)
19	(2,1)
19	(2,2)

Page1:
- 61354, Madeleine, madel@hotmail.com, 11, 1.8
- 12345, Smith, smith@hotmail.com, 12, 2
- 45456, Jasmine, jas@hotmail.com, 18, 3.4

Page2:
- 56565, Jon, jon@hotmail.com, 19, 3.2
- 12456, Lauren, laur@hotmail.com, 19, 3.4

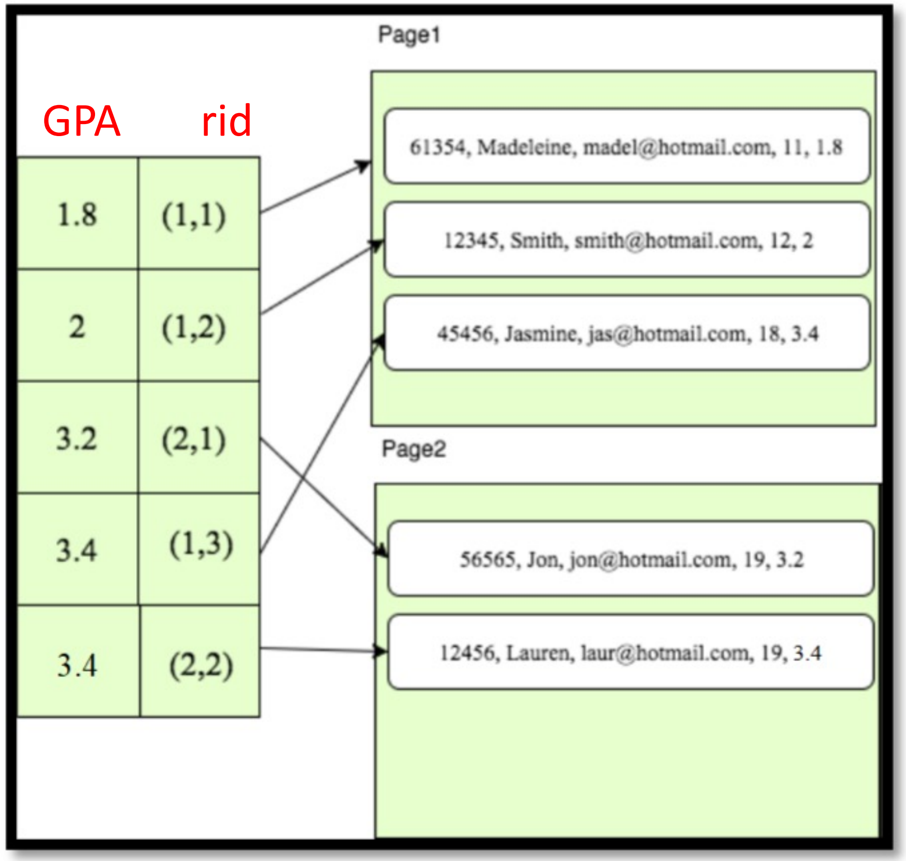
Q2b) GPA

- Data entry: (GPA, rid)
- Unclustered index



Page1
61354, Madeleine, madel@hotmail.com, 11, 1.8
12345, Smith, smith@hotmail.com, 12, 2
45456, Jasmine, jas@hotmail.com, 18, 3.4

Page2
56565, Jon, jon@hotmail.com, 19, 3.2
12456, Lauren, laur@hotmail.com, 19, 3.4



GPA	rid
1.8	(1,1)
2	(1,2)
3.2	(2,1)
3.4	(1,3)
3.4	(2,2)

Page1
61354, Madeleine, madel@hotmail.com, 11, 1.8
12345, Smith, smith@hotmail.com, 12, 2
45456, Jasmine, jas@hotmail.com, 18, 3.4

Page2
56565, Jon, jon@hotmail.com, 19, 3.2
12456, Lauren, laur@hotmail.com, 19, 3.4

Q3)

a.
```
SELECT DepartmentID
FROM Department
WHERE DepartmentFloor = 10
  AND DepartmentBudget < 15000;
```

A) Clustered Hash index on DepartmentFloor
B) Unclustered Hash Index on DepartmentFloor
C) Clustered B+ tree index on (DepartmentFloor, DepartmentBudget)
D) Unclustered hash index on DepartmentBudget
E) No need for an index

b.
```
SELECT EmployeeName, Age, Salary
FROM Employee;
```

A) Clustered hash index on (EmployeeName, Salary)
B) Unclustered hash Index on (EmployeeName, Age)
C) Clustered B+ tree index on (EmployeeName, Age, Salary)
D) Unclustered hash index on (EmployeeID, DepartmentID)
E) No need for an index