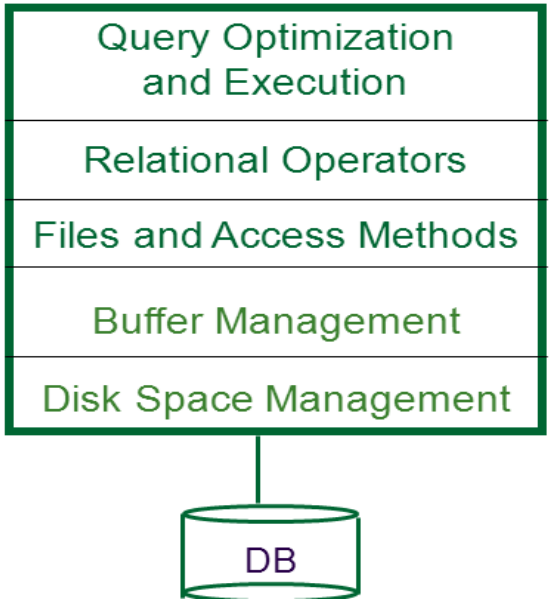
Overview of File Organizations and Indexing

SUN YAT-SEN UNIVERSITY



- 重要概念:

  Record id = <page id, slot #>

  即 *Record id* 由存储这个记录的槽所在的页ID和槽的编号组成。

- 深刻理解文件、页和记录之间的关系
  - 在逻辑上，文件由记录组成；
  - 在物理上，文件由页组成，而每个页包含一组记录。

- 结论：从随机访问的角度来说，读写一条记录需要一次磁盘IO。
- 一次磁盘（块）IO= *seek time (寻道时间)*
  + *rotational delay (旋转延迟)*
  + *transfer time (块传输)*

Goal for Today

- Big picture of overheads for data access – 数据访问开销
  - We'll simplify things to get focused
  - Still, a bit of discipline:
    - Clearly identify assumptions
    - Then estimate cost(估算开销) in a principled way
- Foundation for query optimization(查询优化)
  - Can't choose the fastest scheme without an estimate of speed!

Alternative File Organizations(文件组织)

- Many alternatives exist, *each good for some situations, and not so good in others:*
  - Heap files(堆文件): Suitable when typical access is a file scan retrieving all records.
  - Sorted Files(排序文件): Best for retrieval in *search key* order, or only a "range" of records is needed.
  - Clustered Files (with Indexes) (聚簇文件): Coming soon…

Cost Model(代价模型) for Analysis

- **B:** The number of data blocks
- **R:** Number of records per block
- **D:** (Average) time to read or write a disk block
  - 一次磁盘（块）IO的开销（代价）
- *Average-case* analyses for *uniform random* workloads
- We will ignore:
  - Sequential vs. Random I/O
  - Pre-fetching
  - Any in-memory costs

More Assumptions

- Single record insert and delete.
- Equality selection(等值选择)
  - exactly one match
- For Heap Files:
  - Insert always appends to end of file.
- For Sorted Files:
  - Files compacted after deletions.
  - Selections on search key.

Cost of Operations

B: The number of data pages
R: Number of records per page
D: (Average) time to read or write disk page

	Heap File	Sorted File	Clustered File
Scan all records			
Equality Search			
Range Search			
Insert			
Delete			

Cost of Operations

B: The number of data pages
R: Number of records per page
D: (Average) time to read or write disk page

	Heap File	Sorted File	Clustered File
Scan all records	BD	BD	
Equality Search			
Range Search			
Insert			
Delete			

Cost of Operations

B: The number of data pages
R: Number of records per page
D: (Average) time to read or write disk page

	Heap File	Sorted File	Clustered File
Scan all records	BD	BD	
Equality Search	0.5 BD	$(\log_2 B) * D$	
Range Search			
Insert			
Delete			

Cost of Operations

B: The number of data pages
R: Number of records per page
D: (Average) time to read or write disk page

	Heap File	Sorted File	Clustered File
Scan all records	BD	BD	
Equality Search	0.5 BD	$(\log_2 B) * D$	
Range Search (范围检索)	BD	$[(\log_2 B) +$ #match pg$]*D$	
Insert			
Delete			

Cost of Operations

B: The number of data pages
R: Number of records per page
D: (Average) time to read or write disk page

	Heap File	Sorted File	Clustered File
Scan all records	BD	BD	
Equality Search	0.5 BD	$(\log_2 B) * D$	
Range Search	BD	$[(\log_2 B) +$ #match pg]*D	

Insert	2D
	• 最后一块：
	  · 读1次，写1次
		$((\log_2 B)+B)D$
		• 查找
		• 插入新记录
		  · 后移后续所有记录：对于后半部分（0.5 B）的每一块，读1次，写1次	

Delete			

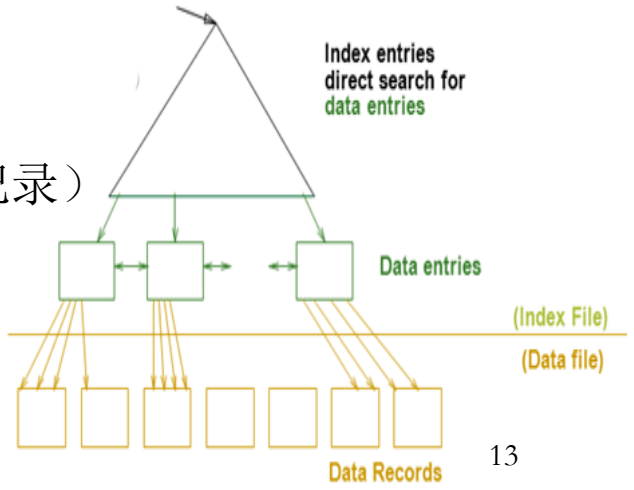
Cost of Operations

B: The number of data pages
R: Number of records per page
D: (Average) time to read or write disk page

	Heap File	Sorted File	Clustered File
Scan all records	BD	BD	
Equality Search	0.5 BD	$(\log_2 B) * D$	
Range Search	BD	$[(\log_2 B) +$ #match pg]*D	
Insert	2D	$((\log_2 B)+B)D$	
Delete	0.5BD + D • 查找 • 对找到的那块，写1次	$((\log_2 B)+B)D$ • 查找 • 删除记录 · 前移后续所有记录：对于后半部分（0.5 B）的每一块，读1次，写1次	

Indexes – 索引

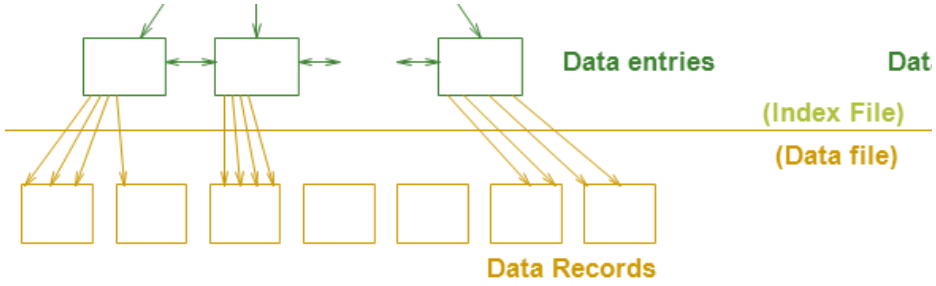
- 用途: Allow record retrieval *by value* in one or more fields
  - Find all students in the “CS” department
  - Find all students with a gpa > 3
- *Index* : disk-based data structure for fast lookup by value
  - *Search key(搜索键):* any subset of columns in the relation.
  - *Search key* need not be a *key* of the relation
    - Can have multiple items matching a lookup
  - 索引是为关系文件建立的索引文件
  - 索引文件由两部份组成
    1. 数据项部分
       - Data Entry(数据项) ⟺ data record（数据记录）
    2. 引导部份
       - 树索引技术
       - Hash索引



Indexes – 索引(Contd.)

索引是文件

- Index contains a collection of *data entries (数据项)*



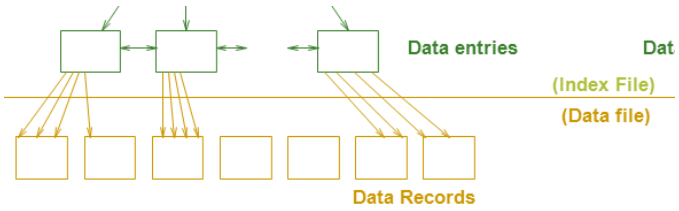
Data Entry(数据项)⟺ data record（数据记录）

- Items associated with each search key value ***k*** -- k*
- Data entries come in various forms, as we'll see

1st Question to Ask About Indexes

- What kinds of selections (lookups) do they support?
  - Selection: <key> <op> <constant>
  - Equality selections (op is =)?
  - Range selections (op is one of <, >, <=, >=, BETWEEN)?

Index Breakdown



- What selections does the index support
- Representation of data entries in index
  - i.e., what kind of info is the index actually storing?
    - 3 alternatives here
- Clustered vs. Unclustered Indexes
- Single Key vs. Composite Indexes
- Tree-based, hash-based, other

Alternatives for Data Entry **k*** in Index

- Three alternatives:
  1. Actual data record (with key value **k**) -- 数据记录
  2. <**k**, rid>, rid is record id of matching data record
  3. <**k**, rid-list>, rid-list is list of rids of matching data records

- Choice is orthogonal to the indexing technique.
  数据项形式的选择与采用的索引技术无关
  - B+ trees, hash-based structures, …

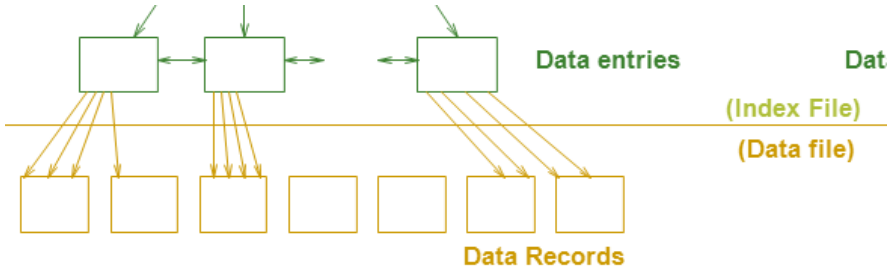
- Can have multiple (different) indexes per file.
  - E.g. file sorted by *age*, with
    - a hash index on *salary*,
    - and a B+tree index on *name*.

Alternatives for Data Entries (Contd.)

- Alternative 1: Data Entry(数据项)⟺ data record（数据记录）

  Actual data record (with key value ***k***)

  - Index as a file organization for records (这种索引也是一种记录文件组织方式)



    - A long side Heap files or sorted files

      Hash B+树

  - At most one Alternative 1 index per relation

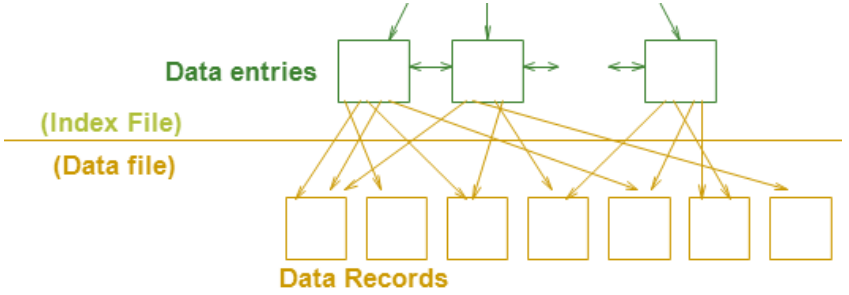
  - No “pointer lookups” to get data records

Alternatives for Data Entries (Contd.)

Data Entry(数据项) ⟺ data record（数据记录）

Alternative 2 **<k**, rid>

<**k**, rid of matching data record>



and Alternative 3 **<k**, rid-list>,

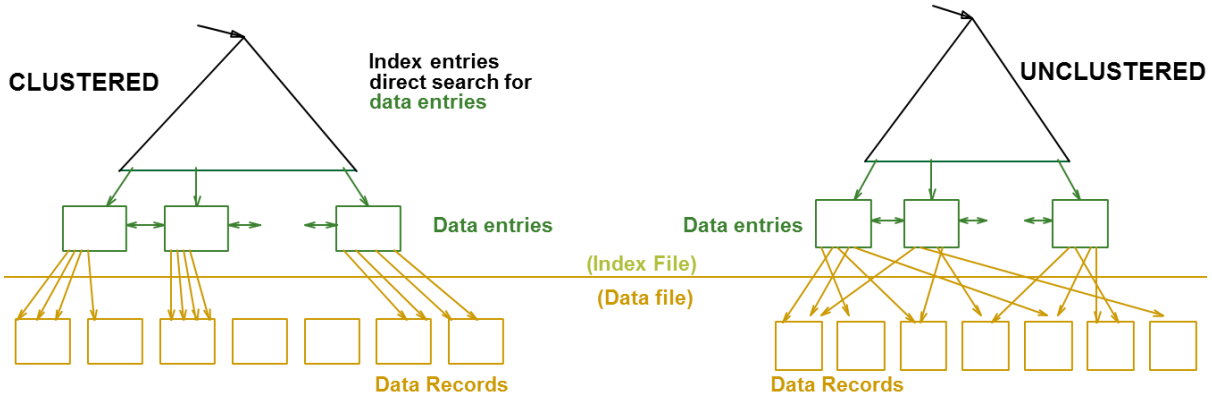
<**k**, list of rids of matching data records>

- Alternative 3 more compact than Alternative 2, but *variable sized data entries*

- Must use Alternatives 2 or 3 to support >1 index per relation.

Index Classification –索引分类

- *Clustered(聚簇索引、主索引)* vs. *Unclustered (非聚簇索引、辅助索引)*:
  - Cost of retrieving data records through index varies *greatly* based on whether index is clustered or not!
- Clustered index – 聚簇索引（或主索引）:
  - order of data records the same as, or `close to', order of index data entries

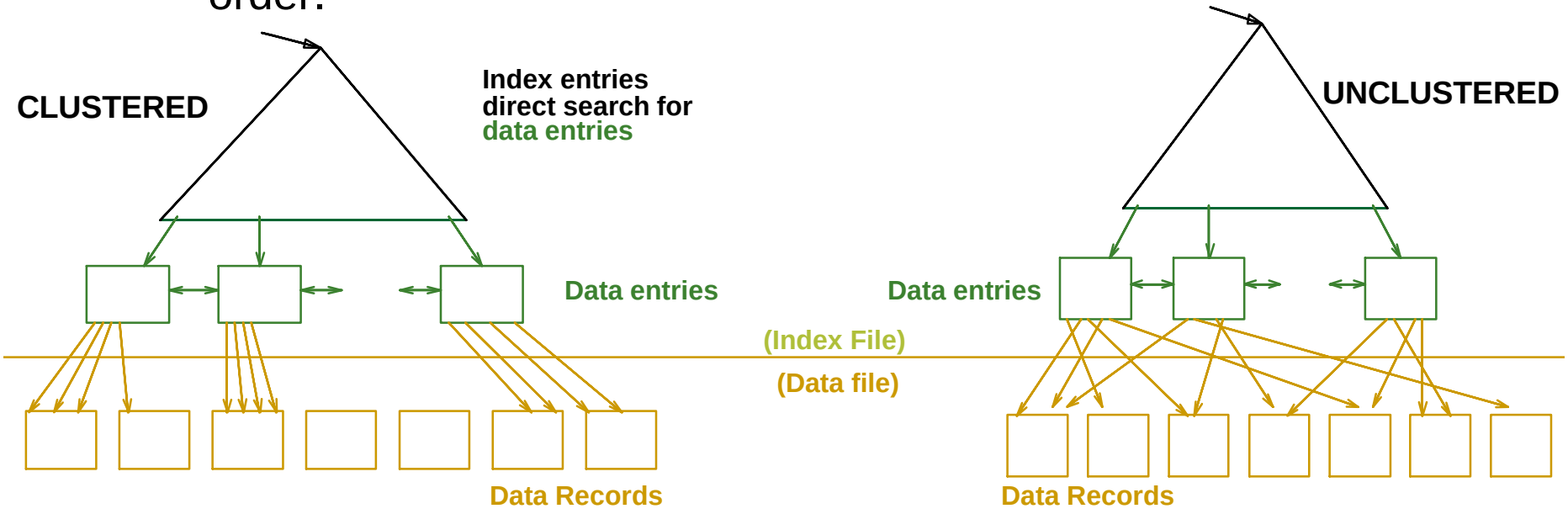


  - Alternative 1 implies clustered, *but not vice-versa*.

Data Entry(数据项)⟺ data record（数据记录）

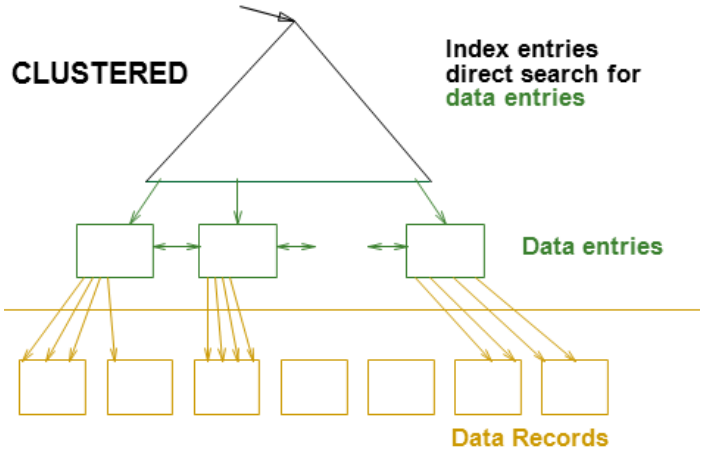
Clustered vs. Unclustered Index

- Alternative 2 data entries, data records in a Heap file.
  - To build clustered index, first sort the Heap file
    - with some free space on each block for future inserts
  - Overflow blocks(溢出块) may be needed for inserts.
    - Thus, order of data records is `close to', but not identical to, the sort order.



Unclustered vs. Clustered Indexes

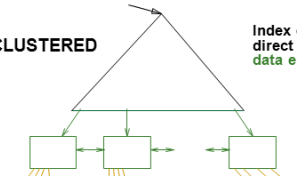
- Clustered Pros –优点
  - Efficient for range searches
  - Possible locality benefits
    - Disk scheduling, prefetching, etc.



- Clustered Cons-缺点
  - More expensive to maintain
    - on the fly or “sloppily” via reorganizations
    - Heap file usually only packed to 2/3 to accommodate inserts

Cost of Operations

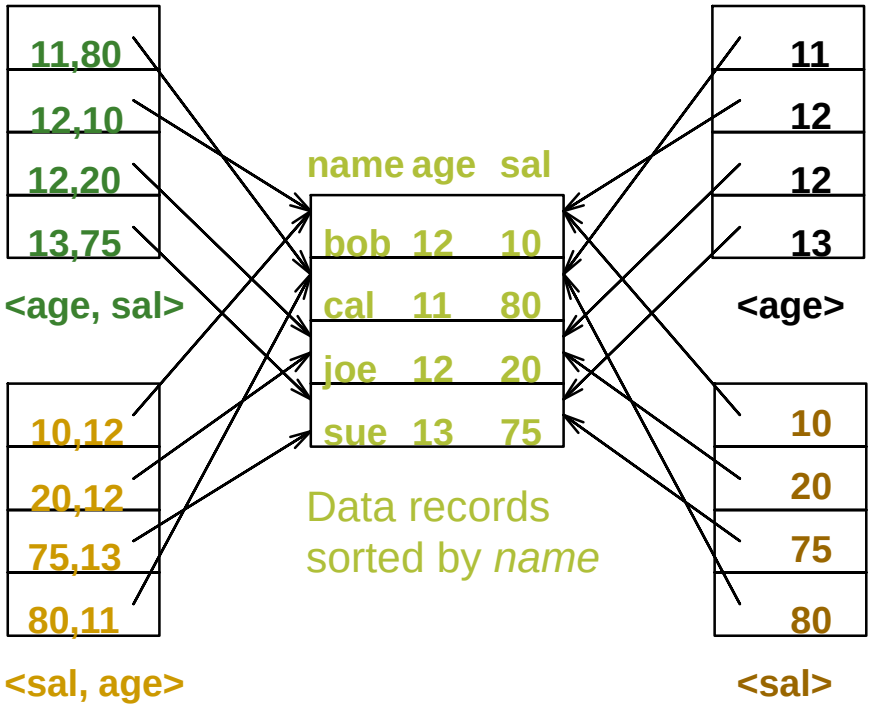
B: The number of data pages
R: Number of records per page
D: (Average) time to read or write disk page

	Heap File	Sorted File	Clustered File Alternative 1
Scan all records	BD	BD	1.5 BD 
Equality Search	0.5 BD	$(\log_2 B) * D$	$(\log_F 1.5B) * D$
Range Search	BD	$[(\log_2 B) +$ #match pg$]*D$	$[(\log_F 1.5B) +$ *#match pg*$]*D$
Insert	2D	$((\log_2 B)+B)D$	$((\log_F 1.5B)+1) * D$
Delete	0.5BD + D	$((\log_2 B)+B)D$ *(because R,W 0.5)*	$((\log_F 1.5B)+1) * D$

Composite Search Keys(复合搜索键)

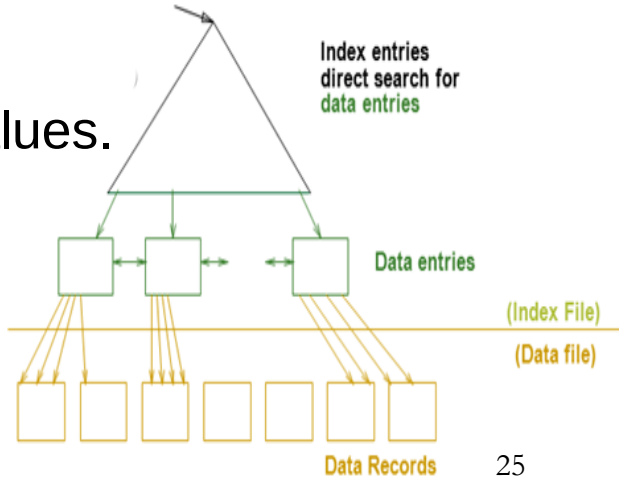
- Search on a combination of fields.
  - Equality query: Every field value is equal to a constant value. E.g. wrt <age,sal> index:
    - age=20 and sal =75
  - Range query: Some field value is not a constant. E.g.:
    - age > 20; or
      age=20 and sal > 10
- Data entries in index can be sorted by search key to support range queries.
  - Lexicographic order(字典次序)
  - Like the dictionary, but on fields, not letters!

Examples of composite key indexes using lexicographic order.



Summary

- Many alternative file organizations exist, each appropriate in some situation.
- If selection queries are frequent, sorting the file or building an *index* is important.
  - Hash-based indexes only good for equality search.
  - Sorted files and tree-based indexes best for range search; also good for equality search. (Files rarely kept sorted in practice; B+ tree index is better.)
- Index is a collection of data entries plus a way to quickly find entries with given key values.



Summary (Contd.)

- Data entries in index can be one of 3 alternatives: (1) actual data records, (2) <key, rid> pairs, or (3) <key, rid-list> pairs.
  - Choice orthogonal to *indexing structure (i.e., tree, hash, etc.)*.
- Usually have several indexes on a given file of data records, each with a different search key.
- Indexes can be classified as *clustered* vs. *unclustered*
  - Differences have important consequences for utility/performance.
- 要求: 深刻理解
  - Cost of Operations 表格
  - 区分索引中的 Data Entry(数据项) 与记录文件中的 data record（数据记录）
  - Data Entry(数据项)的3种形式