**Carnegie Mellon Univ.
Dept. of Computer Science
15-415/615 – DB Applications**

Faloutsos & Pavlo

Lecture #10 (R&G ch8)

File Organizations and Indexing

Overview

➡️ Review
- Index classification
- Cost estimation

Alternative File Organizations

Many alternatives exist, *each good for some situations, and not so good in others:*

- Heap files: Suitable when typical access is a file scan retrieving all records.
- Sorted Files: Best for retrieval in some order, or for retrieving a `range' of records.
- Index File Organizations: (ISAM, or B+ trees)

How to find records quickly?

- E.g., student.gpa = '3'

Q: On a heap organization, with *B* blocks, how many disk accesses?

Heap File Implemented Using Lists

Full Pages

Pages with Free Space
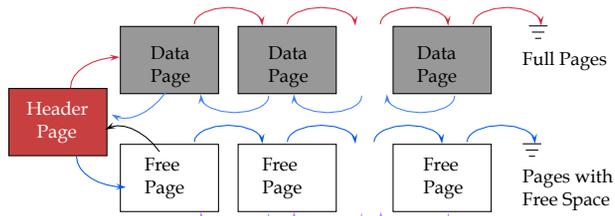
- The header page id and Heap file name must be stored someplace.
- Each page contains 2 `pointers' plus data.

5

How to find records quickly?

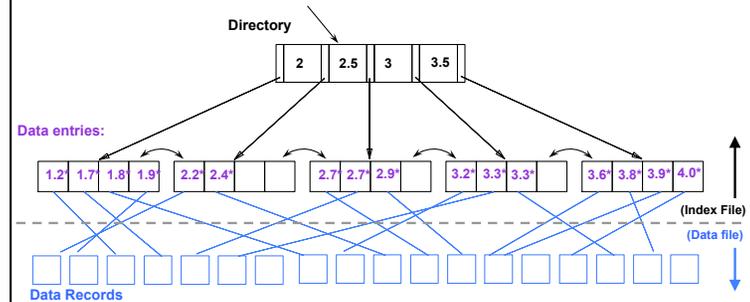- E.g., student.gpa = '3'

Q: On a heap organization, with B blocks, how many disk accesses?

A: B

6

How to accelerate searches?

- A: Indices, like:

7

Example: Simple Index on GPA

Directory

Data entries:

(Index File)

(Data file)

Data Records

An index contains a collection of **data entries**, and supports efficient retrieval of **records** matching a given **search condition**

8

Indexes

- Sometimes, we want to retrieve records by specifying the *values in one or more fields*, e.g.,
  - Find all students in the "CS" department
  - Find all students with a gpa > 3
- An *index* on a file speeds up selections on the *search key fields* for the index.
  - Any subset of the fields of a relation can be the search key for an index on the relation.
  - *Search key* is not the same as *key* (e.g., doesn't have to be unique).

Index Search Conditions

- Search condition = *<search key, comparison operator>*

Examples…

(1) Condition: Department = "CS"
- Search key: "CS"
- Comparison operator: equality (=)

(2) Condition: GPA > 3
- Search key: 3
- Comparison operator: greater-than (>)

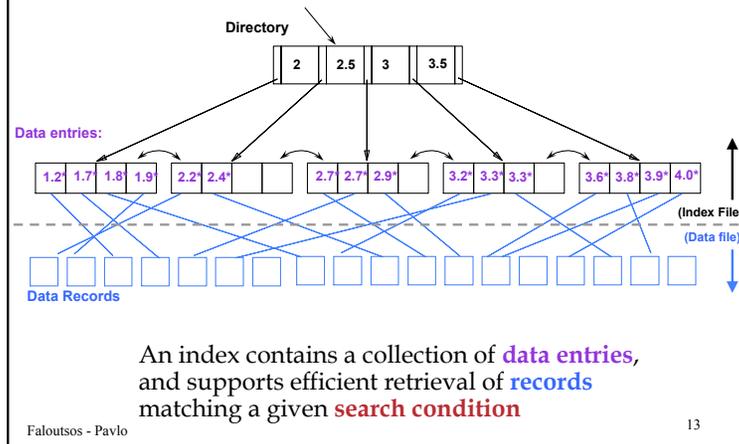## Overview

- Review
- Index classification
  - → Representation of data entries in index
  - Clustered vs. Unclustered
  - Primary vs. Secondary
  - Dense vs. Sparse
  - Single Key vs. Composite
  - Indexing technique
- Cost estimation

Details

- 'data entries' == what we store at the bottom of the index pages
- what would you use as data entries?
- (3 alternatives here)

Example: Simple Index on GPA

Directory

2	2.5	3	3.5

Data entries:

1.2*	1.7*	1.8*	1.9*		2.2*	2.4*			2.7*	2.7*	2.9*		3.2*	3.3*	3.3*		3.6*	3.8*	3.9*	4.0*

(Index File)

(Data file)

Data Records

An index contains a collection of **data entries**, and supports efficient retrieval of **records** matching a given **search condition**

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

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

123	Smith; Main str; 412-999.9999

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

$40	Rid-1

$40	Rid-2

…

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

$40	Rid-1	Rid-2	…

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

1. Actual data record (with key value **k**)
2. <**k**, rid of matching data record>
3. <**k**, list of rids of matching data records>

- Choice is orthogonal to the indexing technique.
  - Examples of indexing techniques: B+ trees, hash-based structures, R trees, …
  - Typically, index contains auxiliary info that directs searches to the desired data entries
- Can have multiple (different) indexes per file.
  - E.g. file sorted on *age*, with a hash index on *name* and a B+tree index on *salary*.

Alternatives for Data Entries (Contd.)

123	Smith; Main str; 412-999.9999

Alternative 1:

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

- Then, this is a clustering/sparse index, and constitutes a file organization (like Heap files or sorted files).
- At most one index on a given collection of data records can use Alternative 1.
- Saves pointer lookups but can be expensive to maintain with insertions and deletions.

Alternatives for Data Entries (Contd.)

Alternative 2

| $40 | Rid-1 |
| $40 | Rid-2 |

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

and Alternative 3

| $40 | Rid-1 | Rid-2 | ... |

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

– Easier to maintain than Alternative 1.

– If more than one index is required on a given file, at most one index can use Alternative 1; rest must use Alternatives 2 or 3.

– Alternative 3 more compact than Alternative 2, but leads to *variable sized data entries* even if search keys are of fixed length.

– Even worse, for large rid lists the data entry would have to span multiple pages!

Faloutsos - Pavlo   17

Overview

- Review
- Index classification
  - Representation of data entries in index
  → – Clustered vs. Unclustered
  - Primary vs. Secondary
  - Dense vs. Sparse
  - Single Key vs. Composite
  - Indexing technique
- Cost estimation

Faloutsos - Pavlo   CMU SCS 15-415   18

Indexing - clustered index example

Clustering/sparse index on ssn

Faloutsos - Pavlo   19

Indexing - non-clustered

Non-clustering / dense index
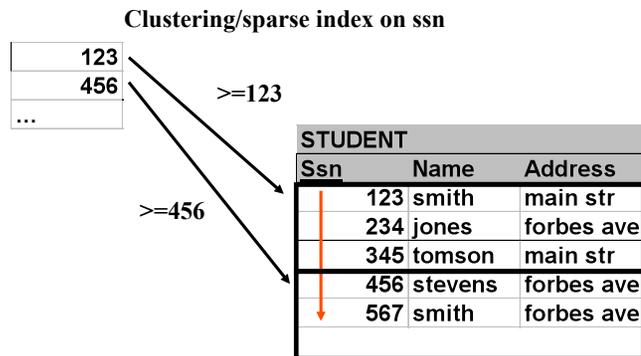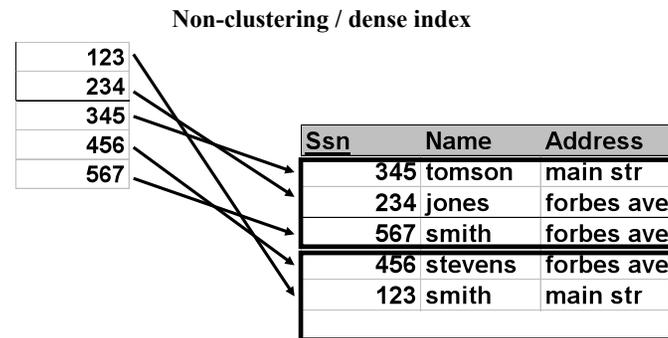
Faloutsos - Pavlo   20

Index Classification - clustered

- *Clustered* vs. *unclustered*: If order of data records is the same as, or `close to', order of index data entries, then called *clustered index*.

CLUSTERED

Index entries direct search for data entries

UNCLUSTERED

Data entries

Data entries

(Index File)

(Data file)

Data Records
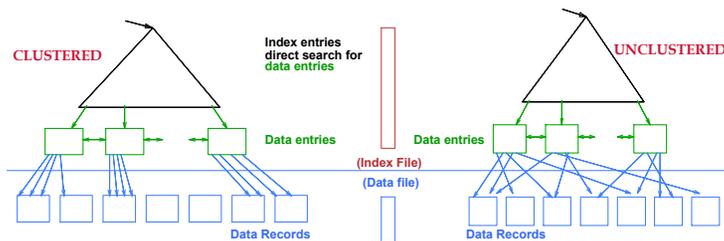
Data Records

Faloutsos - Pavlo

21

Index Classification - clustered

- A file can have a clustered index on at most one search key.
- Cost of retrieving data records through index varies *greatly* based on whether index is clustered!
- Note: Alternative 1 implies clustered, *but not vice-versa*.

But, for simplicity, you may think of them as equivalent..

Faloutsos - Pavlo

22

Clustered vs. Unclustered Index

- Cost of retrieving records found in range scan:
  - Clustered: cost =
  - Unclustered: cost ≈
- What are the tradeoffs????

Faloutsos - Pavlo

23

Clustered vs. Unclustered Index

- Cost of retrieving records found in range scan:
  - Clustered: cost = # pages in file w/matching records
  - Unclustered: cost ≈ # of matching index <u>data entries</u>
- What are the tradeoffs????

Faloutsos - Pavlo

24

Clustered vs. Unclustered Index

- Cost of retrieving records found in range scan:
  - Clustered: cost = # pages in file w/matching records
  - Unclustered: cost ≈ # of matching index <u>data entries</u>
- What are the tradeoffs????
  - Clustered Pros:
    - Efficient for range searches
    - May be able to do some types of compression
  - Clustered Cons:
    - Expensive to maintain (on the fly or sloppy with reorganization)

Faloutsos - Pavlo

25

Overview

- Review
- Index classification
  - Representation of data entries in index
  - Clustered vs. Unclustered
  - → Primary vs. Secondary
  - Dense vs. Sparse
  - Single Key vs. Composite
 - Indexing technique
- Cost estimation

Faloutsos - Pavlo   CMU SCS 15-415

26

Primary vs. Secondary Index

- *Primary*: index key includes the file's primary key
- *Secondary*: any other index

  - Sometimes confused with Alt. 1 vs. Alt. 2/3
  - Primary index never contains duplicates
  - Secondary index may contain duplicates
    - If index key contains a candidate key, no duplicates => unique index

Faloutsos - Pavlo

27

Overview

- Review
- Index classification
  - Representation of data entries in index
  - Clustered vs. Unclustered
  - Primary vs. Secondary
  - → Dense vs. Sparse
  - Single Key vs. Composite
  - Indexing technique
- Cost estimation

Faloutsos - Pavlo   CMU SCS 15-415

28

Dense vs. Sparse Index

- *Dense:* at least one data entry per key value

- *Sparse:* an entry per data page in file
  - **Every sparse index is clustered**!
  - Sparse indexes are smaller; however, some useful optimizations are based on dense indexes.

| Ashby, 25, 3000 |
| Basu, 33, 4003 |
| Bristow, 30, 2007 |

| Ashby |
| Cass |
| Smith |

| Cass, 50, 5004 |
| Daniels, 22, 6003 |
| Jones, 40, 6003 |

| Smith, 44, 3000 |
| Tracy, 44, 5004 |

| 22 |
| 25 |
| 30 |
| 33 |

| 40 |
| 44 |
| 44 |
| 50 |

Sparse Index on Name **Data File** **Dense Index on Age**

Faloutsos - Pavlo   29

Dense vs. Sparse Index

- Sparse <-> Clustering <-> Alt#1 (full record)

| Ashby, 25, 3000 |
| Basu, 33, 4003 |
| Bristow, 30, 2007 |

| Ashby |
| Cass |
| Smith |

| Cass, 50, 5004 |
| Daniels, 22, 6003 |
| Jones, 40, 6003 |

| Smith, 44, 3000 |
| Tracy, 44, 5004 |

| 22 |
| 25 |
| 30 |
| 33 |

| 40 |
| 44 |
| 44 |
| 50 |

Sparse Index on Name **Data File** **Dense Index on Age**

Faloutsos - Pavlo   30

Overview

- Review

- Index classification
  - Representation of data entries in index
  - Clustered vs. Unclustered
  - Primary vs. Secondary
  - Dense vs. Sparse
  - Single Key vs. Composite
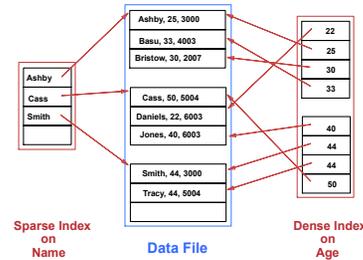 - Indexing technique

- Cost estimation

Faloutsos - Pavlo   CMU SCS 15-415   31

Composite Search Keys

- Search on *combination* of fields.
  - Equality query: Every field is equal to a constant value. E.g. wrt <sal,age> index:
    - age=12 and sal =75
  - Range query: Some field value is not a constant. E.g.:
    - age =12; or age=12 and sal > 20
- Data entries in index sorted by search key for range queries.
  - "Lexicographic" order.

Examples of composite key indexes using lexicographic order.

| 11,80 |
| 12,10 |
| 12,20 |
| 13,75 |

<age, sal>

| 10,12 |
| 20,12 |
| 75,13 |
| 80,11 |

<sal, age>

name	age	sal
bob	12	10
cal	11	80
joe	12	20
sue	13	75

Data records sorted by *name*

| 11 |
| 12 |
| 12 |
| 13 |

<age>

| 10 |
| 20 |
| 75 |
| 80 |

<sal>

Faloutsos - Pavlo   32

Overview

- Review
- Index classification
  - Representation of data entries in index
  - Clustered vs. Unclustered
  - Primary vs. Secondary
  - Dense vs. Sparse
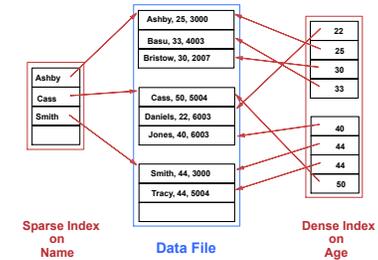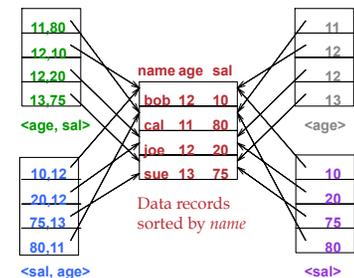 - Single Key vs. Composite
  - Indexing technique
- Cost estimation

Tree vs. Hash-based index

- Hash-based index
  - Good for equality selections.
    - File = a collection of *buckets.* Bucket = *primary* page plus 0 or more *overflow* pages.
    - *Hash function* **h**: **h**(*r.search_key*) = bucket in which record *r* belongs.
- Tree-based index
  - Good for range selections.
    - Hierarchical structure (Tree) directs searches
    - Leaves contain data entries sorted by search key value
    - B+ tree: all root->leaf paths have equal length *(height)*

Overview

- Review
- Index classification
  - Representation
  - ...
- Cost estimation

Cost estimation

- Heap file
- Sorted
- Clustered
- Unclustured tree index
- Unclustered hash index

Methods   Operations(?)

Cost estimation

- Heap file
- Sorted
- Clustered
- Unclustured tree index
- Unclustered hash index

- scan
- equality search
- range search
- insertion
- deletion

Methods   Operations

- Consider only I/O cost;
- suppose file spans *B* pages

Faloutsos - Pavlo   37

Cost estimation

	scan	eq	range	ins	del
Heap					
sorted					
Clust.					
u-tree					
u-hash					

Assume that:
- Clustered index spans 1.5*B* pages (due to empty space)
- Data entry= 1/10 of data record

Faloutsos - Pavlo   38

Cost estimation

	scan	eq	range	ins	del
Heap	B				
sorted	B				
Clust.	1.5B				
u-tree	~B				
u-hash	~B				

Faloutsos - Pavlo   39

Cost estimation

- heap: seq. scan
- sorted: binary search
- index search

Faloutsos - Pavlo   40

Cost estimation

index – cost? In general
- levels of index +
- blocks w/ qual. tuples

for primary key – cost:

h   for clustering index

h' +1   for non-clustering

h'

#1

#2

...

#B

41

Cost estimation

	scan	eq	range	ins	del
Heap	B	B/2			
sorted	B	$\log_2 B$			
Clust.	1.5B	h			
u-tree	~B	1+h'			
u-hash	~B	~2			

h= height of btree $\sim \log_F (1.5B)$
h' = height of unclustered index btree $\sim \log_F (1.5B)$

42

Cost estimation

index – cost?
- levels of index +
- blocks w/ qual. tuples

sec. key – clustering index

h + #qual-pages

h

#1

#2

...

#B

43

Cost estimation

index – cost?
- levels of index +
- blocks w/ qual. tuples

sec. key – non-clust. index

h' + #qual-records

(actually, a bit less...)

h'

#1

#2

...

#B

44

Cost estimation

	scan	eq	range	ins	del
Heap	B	B/2	B		
sorted	B	$\log_2 B$	<- +m		
Clust.	1.5B	h	<- +m		
u-tree	~B	1+h'	<- +m'		
u-hash	~B	~2	B		

m: # of qualifying pages
m' : # of qualifying **records**

Cost estimation

	scan	eq	range	ins	del
Heap	B	B/2	B	2	Search+1
sorted	B	$\log_2 B$	<- +m	Search+B	Search+B
Clust.	1.5B	h	<- +m	Search+1	Search+1
u-tree	~B	1+h'	<- +m'	Search+2	Search+2
u-hash	~B	~2	B	Search+2	Search+2

Cost estimation - big-O notation:

	scan	eq	range	ins	del
Heap	B	B	B	2	B
sorted	B	$\log_2 B$	$\log_2 B$	(B)	(B)
Clust.	B	$\log_F B$	$\log_F B$	$\log_F B$	$\log_F B$
u-tree	B	$\log_F B$	$\log_F B$	$\log_F B$	$\log_F B$
u-hash	B	1	(B)	1	1

Index specification in SQL:1999

CREATE INDEX IndAgeRating ON Students
    WITH STRUCTURE=BTREE,
    KEY = (age, gpa)

Summary

- To speed up selection queries: **index**.
- Terminology:
  - Clustered / non-clustered index
    - Clustered = sparse = alt#1
  - primary / secondary index
- Typically, B-tree index
- hashing is only good for equality search
- At most one clustered index per table
  - many non-clustered ones are possible
  - composite indexes are possible

Faloutsos - Pavlo

49