CSE462/562: Database Systems (Spring 23)

Lecture 10: Indexing and cost analysis

3/2/2023

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Index classification

- Representation of data entries in index
  - i.e., what kind of info is the index actually storing?
  - 3 alternatives
- What selections does it support
- Indexing techniques: tree/hash/other

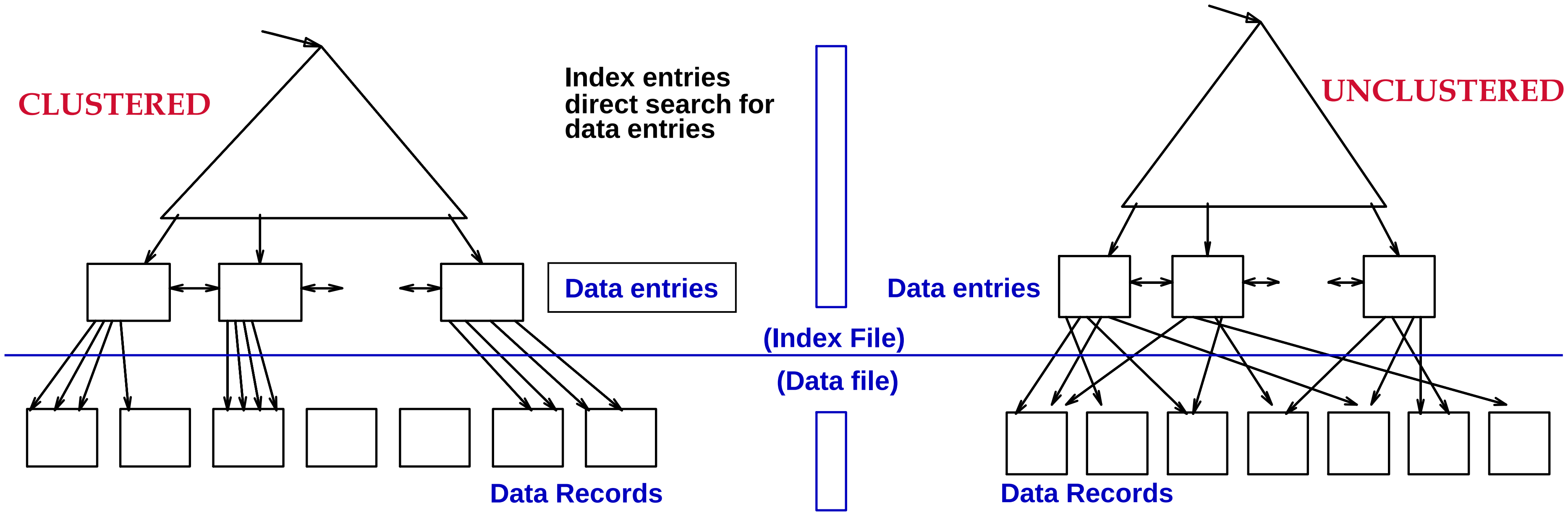
- Today
  - Primary vs. Secondary Indexes
    - Unique indexes
  - Clustered vs. Unclustered Indexes
  - Single Key vs. Composite Indexes

Clustered vs unclustered index

- Clustered index
  - An index over a file such that the order of the data records is the same as, or "close to" that of the index data entries
    - A file can only be clustered on one index key
    - Sorted file can be used for clustering, but may be expensive to maintain
      - Can we use heap file? Yes, but with some tricks.
  - Using Alternative 1 in a B+-tree implies clustered, *but not vice-versa*.
    - aka clustered file

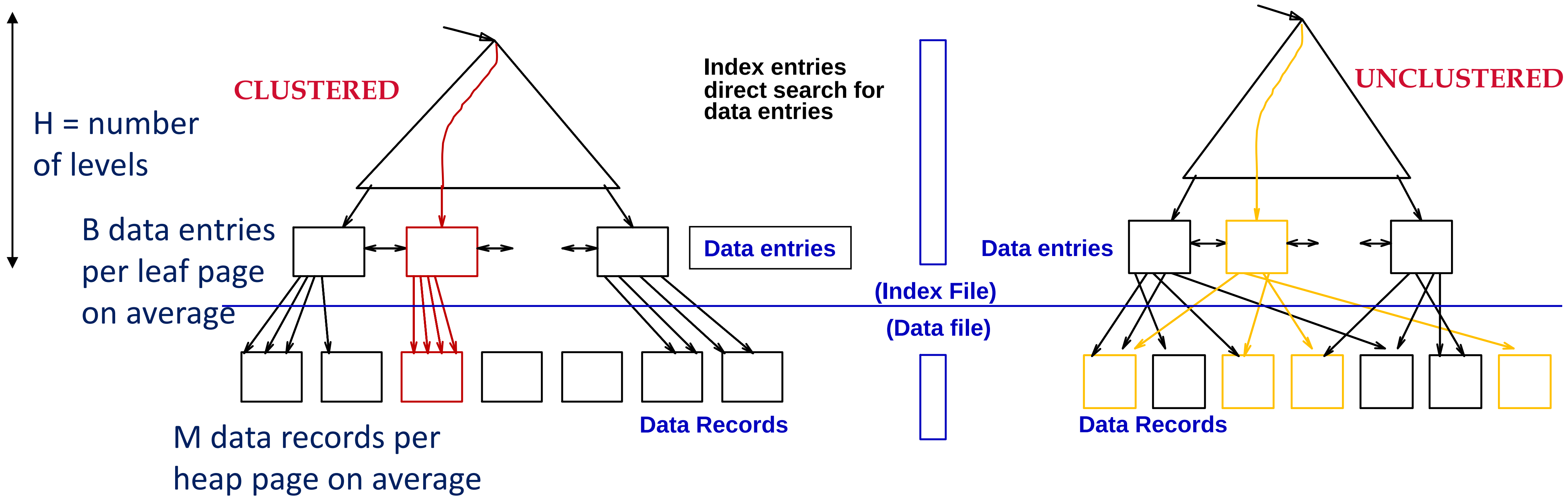
Clustered vs unclustered index

- Assume alternative 2 for data entries, and data records are stored in a heap file.
  - To build clustered index
    - first sort the heap file, with some free space on each block for future updates/inserts.
    - The percentage of free space in the initial sort/append is called *fill factor*
  - Overflow pages may be needed for inserts/updates.
    - Thus, the order of data records is "close to", if not not identical to, the sort order.



Access cost of clustered vs unclustered index

- Cost of accessing data records through index varies *greatly* based on whether index is clustered!
  - e.g. range scan with n matching data records in a B-Tree
    - assuming we ignore the buffer pool's effect
    - clustered: $H + \left\lceil \frac{n}{M} \right\rceil$ I/Os
    - unclustered: $H + \left\lceil \frac{n}{B} \right\rceil - 1 + n$ I/Os



Example: cost analysis

- Example
  - page size = 4096 B
  - For Table A(x, y, z), record length = 64, sizeof(x) == 8 and sizeof(y) == 8.
    - number of records = $2^{20} = 1{,}048{,}576$
  - There're equal number of records with $x > 0$ and $x \leq 0$
  - There's $2^{10} = 1024$ records with $y = 1$
- Assumptions:
  - No page header overhead
  - record id and page id are both 8 bytes
  - no alignment padding needed for index and data entries, no record header overhead
  - Fill factor = 80% for all pages.
  - Ignore the caching effect of buffer pool -> each page access = 1 I/O

- Heap file:
  - Number of pages:
  - Cost of finding all records with $y = 1 \; and \; x > 0$:
  - Cost of finding all records with $x = 1$:
  - Cost of insertion of a record:
  - Cost of deletion of all records with y = 1:

Example: cost analysis

- Example
  - page size = 4096 B
  - For Table A(x, y, z), record length = 64, sizeof(x) == 8 and sizeof(y) == 8.
    - number of records = $2^{20} = 1{,}048{,}576$
  - There're equal number of records with $x > 0$ and $x \leq 0$
  - There's $2^{10} = 1024$ records with $y = 1$
- Assumptions:
  - No page header overhead
  - record id and page id are both 8 bytes
  - no alignment padding needed for index and data entries, no record header overhead
  - Fill factor = 80% for all pages.
  - Ignore the caching effect of buffer pool -> each page access = 1 I/O

- B-tree file over (y), alt. 1:
  - Number of pages:
  - Cost of finding all records with $y = 1 \ and \ x > 0$:
  - Cost of finding all records with $x = 1$:
  - Cost of insertion of a record:
  - Cost of deletion of all records with y = 1:

Example: cost analysis

- Example
  - page size = 4096 B
  - For Table A(x, y, z), record length = 64, sizeof(x) == 8 and sizeof(y) == 8.
    - number of records = $2^{20} = 1{,}048{,}576$
  - There're equal number of records with $x > 0$ and $x \leq 0$
  - There's $2^{10} = 1024$ records with $y = 1$
- Assumptions:
  - No page header overhead
  - record id and page id are both 8 bytes
  - no alignment padding needed for index and data entries, no record header overhead
  - Fill factor = 80% for all pages.
  - Ignore the caching effect of buffer pool -> each page access = 1 I/O

- B-tree file over (y), alt. 2 and clustered:
  - Number of pages:
  - Cost of finding all records with $y = 1\ and\ x > 0$:
  - Cost of finding all records with $x = 1$:
  - Cost of insertion of a record:
  - Cost of deletion of all records with y = 1:

Example: cost analysis

- Example
  - page size = 4096 B
  - For Table A(x, y, z), record length = 64, sizeof(x) == 8 and sizeof(y) == 8.
    - number of records = $2^{20} = 1{,}048{,}576$
  - There're equal number of records with $x > 0$ and $x \leq 0$
  - There's $2^{10} = 1024$ records with $y = 1$
- Assumptions:
  - No page header overhead
  - record id and page id are both 8 bytes
  - no alignment padding needed for index and data entries, no record header overhead
  - Fill factor = 80% for all pages.
  - Ignore the caching effect of buffer pool -> each page access = 1 I/O

- B-tree file over (y), alt. 2 and unclustered:
  - Number of pages:
  - Cost of finding all records with $y = 1\ and\ x > 0$:
  - Cost of finding all records with $x = 1$:
  - Cost of insertion of a record:
  - Cost of deletion of all records with y = 1:

Example: cost analysis

- Example
  - page size = 4096 B
  - For Table A(x, y, z), record length = 64, sizeof(x) == 8 and sizeof(y) == 8.
    - number of records = $2^{20} = 1{,}048{,}576$
  - There're equal number of records with $x > 0$ and $x \leq 0$
  - There's $2^{10} = 1024$ records with $y = 1$
- Assumptions:
  - No page header overhead
  - record id and page id are both 8 bytes
  - no alignment padding needed for index and data entries, no record header overhead
  - Fill factor = 80% for all pages.
  - Ignore the caching effect of buffer pool -> each page access = 1 I/O

- B-tree file over (y, x), alt. 2 and clustered:
  - Number of pages:
  - Cost of finding all records with $y = 1\ and\ x > 0$:
  - Cost of finding all records with $x = 1$:
  - Cost of insertion of a record:
  - Cost of deletion of all records with y = 1:

Example: cost analysis

- Example
  - page size = 4096 B
  - For Table A(x, y, z), record length = 64, sizeof(x) == 8 and sizeof(y) == 8.
    - number of records = $2^{20} = 1{,}048{,}576$
  - There're equal number of records with $x > 0$ and $x \leq 0$
  - There's $2^{10} = 1024$ records with $y = 1$
- Assumptions:
  - No page header overhead
  - record id and page id are both 8 bytes
  - no alignment padding needed for index and data entries, no record header overhead
  - Fill factor = 80% for all pages.
  - Ignore the caching effect of buffer pool -> each page access = 1 I/O

- B-tree file over (y, x), alt. 2 and unclustered:
  - Number of pages:
  - Cost of finding all records with $y = 1\ and\ x > 0$:
  - Cost of finding all records with $x = 1$:
  - Cost of insertion of a record:
  - Cost of deletion of all records with y = 1:

Tradeoffs between clustered and unclustered indexes

- What are the tradeoffs?
- Clustered Pros
  - Efficient for range searches for records: sequential access in a sorted file
  - May be able to do some types of compression
  - Locality benefits
- Clustered Cons
  - Expensive to maintain (on the fly or sloppy with reorganization)
- Unclustered
  - Pros: easy and efficient to maintain, allow multiple indexes
  - Cons: expensive for range scans for records: 1 random IO for each matching record.

Primary, secondary and unique index

- Primary index: index key contains the primary key
  - e.g., for student table, an index over (sid) is its primary index
  - at most one per relation
- Unique index: index key contains a candidate key
  - Primary index is a unique index, but not vice versa
  - Can be clustered or unclustered.
- Secondary index (not well-defined but often used)
  - It may have different meanings
    - an index that is not indexed over the primary key
    - unclustered
    - or both

Unconventional “index”

- There might be alternative file organization also considered/called as “index”
  - e.g., columnstore index in MS SQL Server
    - Good compression, fast scan, but more expensive to update in general

Column segment: compressed and has min/max stats

row group 1

sid	name	major	adm_year
100	Alice	CS	2021
101	Bob	CE	2020

row group 2

sid	name	major	adm_year
102	Charlie	CS	2021
103	David	CS	2020

- What it really means:
  - It may be used as the primary storage format (aka clustered columnstore)
    - i.e., may be thought of as a clustered file or a file organization
  - It may also be used as a copy of the (subset of) data (aka unclustered columnstore)
    - i.e., may be thought of as a secondary and unclustered index

Summary

- This lecture
  - Indexing and cost analysis

- Next lecture
  - Relational model and SQL

- Reminder:
  - Mid-term next Thursday (3/9/23, 7:15 PM - 8:45 PM at Knox 104)
    - Open book, paper materials only (lecture slides, textbook, homework assignments, notes)
      - No electronic devices (cell phone/laptop/tablets/…)
      - Calculators allows
    - Please arrive at least 10 minutes early.

  - Mid-term Q&A during lecture on 3/9