

Overview of Storage and Indexing

(Chapter 9, 3rd edition)

"How index-learning turns no student pale Yet holds the eel of science by the tail." -- Alexander Pope (1688-1744) Cost Model for Our Analysis

- Measuring number of page I/O's ignores gains of pre-fetching blocks of pages and hits in the buffer; thus, even I/O cost is only approximated.
- Block access is not assumed (where seek time is incurred only once)
- Average-case analysis; based on several simplistic assumptions.
- Clock and LRU seem to give same or very similar results (why?)

 \boxtimes Good enough to show the overall trends!

Management Systems 3ed, R. Ramakrishnan and J. Gehrke

Cost Model for Our Analysis

Management Systems 3ed, R. Ramakrishnan and J. Gehrke

We ignore CPU costs, for simplicity:

- **B:** The number of data pages
- R: Number of records per page
- D: (Average) time to read or write disk page
- C: Average time to process a record (in memory)
- H: Time required to apply the hash function

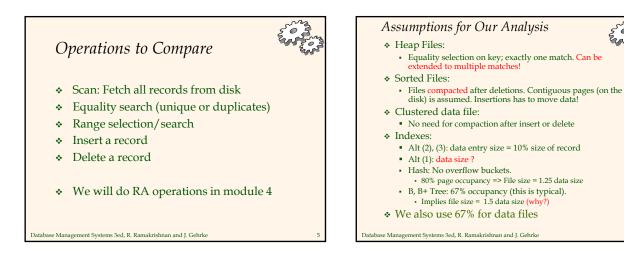
Typically, D is 15 msecs, C and H are 100 nano secs Hence the assumption that cost of I/O dominates.

We will look Big-O average complexity; you should understand best and worst complexity as well!

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

Comparing File Organizations Heap files (random order; insert at eof) Sorted files, sorted on <age, sal> (contiguous) Clustered B+ tree file, search key <age, sal> Heap file with unclustered B + tree index on search key <age, sal> Heap file with unclustered hash index on search key <age, sal> Is it possible to have clustered hash index?

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke



Cost of	Oper	ations			505	
B # data pages D pg i/o cost R recs per page	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) delete	200
(1) Heap						
(2) Sorted (contiguous)						
(3) Clustered B+ tree index						
(4) Unclustered B+ Tree Index						
(5) Unclustered Hash index						7

Cost of	Oper	ations	(averaz	ges)	Ser Star
B # data pages D page i/o cost R recs per page	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) delete
(1) Heap	B(D+RC) Or BD	0.5B (D+RC) Or ½ *BD	B(D+RC)	2D + C (added at the front) At the end?	Search + D + C (no compacting)
(2) Sorted (contiguous)					
(3) Clustered					
(4) Unclustered Tree Index					
(5) Unclustered Hash index					

õ

Cost of Operation			B: # pages, R: recs/page; D: i/o cost/page		
B # data pages D pg i/o cost R recs per page	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) delete
(1) Heap	B(D+RC) Cannot do better!	0.5B (D+RC) Not good	B(D+RC) Not good	2D + C (at the end)	Search (b) + C+ D Not good
(2) Sorted (contiguous)		0	U		
(3) Clustered Tree Index					
(4) Unclustered Tree Index					
(5) Unclustered Hash index					

Cost of C	Opera	tions B: #	ŧ pages, R: recs/page	: D: i/o cost/page	Service Se	200
B # data pages D pg i/o cost R recs per page	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) delete	2
(1) Heap	B(D+RC) Cannot do better!	0.5B (D+RC) Not good	B(D+RC) Not good	2D + C (at the end) Good	Search (b) + C+ D Not good	
(2) Sorted (contiguous)	B(D+RC)	$Dlog_2 B + 2$ comparisons for each page + $Clog_2 R$	Dlog 2 B + + matching pages(mp)*D + mp*RC	Search(b) + 2*0.5B (D+RC) (for moving records)	Search(b) + 2*0.5B (D+RC) (needs moving records)	
(3) Clustered						
(4) Unclustered Tree Index						
(5) Unclustered Hash index Database management	oystems oed,	r. ramarnsnnan	anu j. Genrke			10

Cost of	Operi	ations	B: # pages, R: recs/pa	ge; D: i/o cost/page	e con	na.
B # data pages D pg i/o cost R recs per page	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) delete	9 A
(1) Heap	B(D+RC) Cannot do better	0.5B (D+RC) Not good	B(D+RC) Not good	2D + C (at the end) Good	Search (b) + C+ D Not good	
(2) Sorted (contiguous)	B(D+RC) Cannot do better	Dlog ₂ B+ Clog ₂ R Good	Dlog 2 B + + matching pages*D + mp*RC Good	Search(b) + 2*0.5B (D+RC) Not at all good	Search(b) + 2*0.5B (D+RC) Not at all good	
(3) Clustered B+ tree						
(4) Unclustered B+ Tree Index (Alt 2)						
(5) Unclustered Hash index						11

(Cost of (Operat	ions B:	# pages, R: recs/page;	D: i/o cost/page	Error and
	B # data pages D pg i/o cost R recs per page	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) delete
	(1) Heap	B(D+RC) Cannot do better	0.5B (D+RC) Not good	B(D+RC) Not good	2D + C (at the end) Good	Search (b) + C+ D Not good
	(2) Sorted (contiguous)	B(D+RC) Cannot do better	Dlog ₂ B + Clog ₂ R Good	Dlog ₂ B + + matching pages*D + mp*RC Good	Search(b) + 2*0.5B (D+RC) Not good	Search(b) + 2*0.5B (D+RC) Not good
	(3) Clustered B+ tree	1.5B(D+RC) no need to use index! scan	Dlog _F .15B + D + Clog ₂ R	Dlog _F .15B + + mp*D + mp*RC	Search + D + $Clog_2 R$ (assumes free space)	Search + D + Clog ₂ R
	(4) Unclustered B+ Tree Index (Alt 2)					
D	(5) Unclustered Hash index					<u> 1</u>

Cost of C	Opera	tions Bit	‡ pages, R: recs/page	; D: i/o cost/page	and	Sal
B # data pages D pg i/o cost R recs per page	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) delete	all a
(1) Heap	B(D+RC)	0.5B (D+RC) Not good	B(D+RC)	2D + C (at the end) Good	Search (b) + C+ D Not good	
(2) Sorted (contiguous)	B(D+RC)	Dlog ₂ B + Clog ₂ R Good	Dlog ₂ B + + matching pages*D + mp*Clog ₂ R	Search(b) + 2*0.5B (D+RC) Not good	Search(b) + 2*0.5B (D+RC) Not good	
(3) Clustered B+ tree (why don't we use this all the time?)	1.5B(D+R C) Cannot do better	Dlog _F .15B + D + Clog ₂ R Good	Dlog _F .15B + + mp*D + mp*Clog ₂ R Good	Search + D + Clog ₂ R Good	Search + D + Clog ₂ R Good	
(4) Unclustered B+ Tree Index (Alt 2)						
(5) Unclustered Hash index D						13

Cost of C	peration	15			~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	
B # data pages D pg i/o cost R recs per page	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) delete	200
(1) Heap	B(D+RC)	0.5B (D+RC) Not good	B(D+RC)	2D + C (at the end) Good	Search (b) + C+ D Not good	
(2) Sorted (contiguous)	B(D+RC)	Dlog ₂ B+ Clog ₂ R Good	Dlog ₂ B + + matching pages*D + mp*Clog ₂ R	Search(b) + 2*0.5B (D+RC) Not good	Search(b) + 2*0.5B (D+RC) Not good	
(3) Clustered B+ tree	1.5B(D+RC) Cannot do better	Dlog _F .15B + D + Clog ₂ R Good	Dlog _F .15B + + mp*D + mp*Clog ₂ R Good	Search + D + $\operatorname{Clog}_2 R$ Good	Search + D + $\operatorname{Clog}_2 R$ Good	
(4) Unclustered B+ Tree Index (Alt 2)	1.5B(D+RC)	DLog _F 0.15B + D + RC	DLog _F 0.15B + mp*R*D + mp*C*R	DLog _F 0.15B + D + 2D + C	Search(b) + 2D (index + data write)	
(5) Unclustered Hash index						
Database Managem	ent Systems 3ed,	, R. Ramakrishna	n and J. Gehrke			14

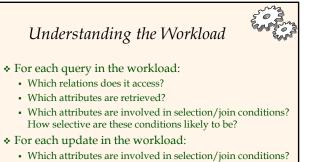
(Cost of C	pera	tions			eron a	00
	B # data pages D pg i/o cost R recs per page	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) delete	South Barrier
	(1) Heap	B(D+RC) Cannot do better	0.5B (D+RC) Not good	B(D+RC)	2D + C (at the end) Good	Search (b) + C+ D Not good	
	(2) Sorted (contiguous)	B(D+RC) Cannot do better	Dlog ₂ B + Clog ₂ R Good	Dlog ₂ B + + matching pages*D + mp*Clog ₂ R	Search(b) + 2*0.5B (D+RC) Not good	Search(b) + 2*0.5B (D+RC) Not good	
	(3) Clustered B+ tree	1.5B(D+R C) Cannot do better	Dlog _F .15B + Clog ₂ R Good	Dlog _F .15B + + mp*D + mp*Clog ₂ R Good	Search + D + $\operatorname{Clog}_2 R$ Good	Search + D + $Clog_2 R$ Good	
	(4) Unclustered B+ Tree Index (Alt 2)	1.5B(D+R C) Cannot do better	DLog _F 0.15B + D + RC Good	DLog _F 0.15B + mp*R*D + mp*C*R Not Good	DLog _F 0.15B + D + 2D + C Good	Search(b) + 2D (index + data write) Good	
D	(5) Unclustered Hash index						15

Cost of Op	perations				e and	
B # data pages D pg i/o cost R recs per page	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) delete	Sand?
(1) Heap	B(D+RC) Cannot do better	0.5B (D+RC) Not good	B(D+RC)	2D + C (at the end) Good	Search (b) + C+ D Not good	
(2) Sorted (contiguous)	B(D+RC) Cannot do better	Dlog ₂ B + Clog ₂ R Good	Dlog ₂ B + + matching pages*D + mp*Clog ₂ R	Search(b) + 2*0.5B (D+RC) Not good	Search(b) + 2*0.5B (D+RC) Not good	
(3) Clustered B+ tree	1.5B(D+RC) Cannot do better	Dlog _F .15B + Clog ₂ R Good	Dlog _F .15B + + mp*D + mp*Clog ₂ R Good	Search + D + $Clog_2 R$ Good	Search + D + $Clog_2 R$ Good	
(4) Unclustered B+ Tree Index (Alt 2) (data file is heap file underneath)	1.5B(D+RC) Cannot do better	DLog _F 0.15B + D + RC Good	DLog _F 0.15B + mp* R *D + mp*C*R Not Good	DLog _F 0.15B + D + 2D + C Good	Search(b) + 2D (index + data write) good	
(5) Unclustered Hash index D	1.25B(D + RC) Do not use index	H + D + 0.5*8RC + D for first match	1.25B(D + RC)	2D + C + H + 2D + C	H + 2D + 4RC + 2D + 2D	16

Cost of Open	rations B: # p	ages, R: recs/page; E): i/o cost/page		m
	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) delete
(1) Heap	B(D+RC) Cannot do better	0.5B (D+RC) Not good	B(D+RC)	2D + C (at the end) Good	Search (b) + C+ D Not good
(2) Sorted (contiguous)	B(D+RC) Cannot do better	Dlog ₂ B + Clog ₂ R Good	Dlog ₂ B + + matching pages*D + mp*Clog ₂ R	Search(b) + 2*0.5B (D+RC) Not good	Search(b) + 2*0.5B (D+RC) Not good
(3) Clustered B+ tree	1.5B(D+RC) Cannot do better	Dlog _F .15B + Clog ₂ R Good	Dlog _F .15B + + mp*D + mp*Clog ₂ R Good	Search + D + $\operatorname{Clog}_2 R$ Good	Search + D + $\operatorname{Clog}_2 R$ Good
(4) Unclustered B+ Tree Index (Alt 2)	1.5B(D+RC) Cannot do better	DLog _F 0.15B + D + RC Good	DLog _F 0.15B + mp*R*D + mp*C*R Not Good	DLog _F 0.15B + D + 2D + C Good	Search(b) + 2D (index + data write) Good
(5) Unclustered Hash index Da	1.25B(D + RC) Cannot do	H + 0.5*8RC + 2D Good	1.25B(D + RC)	2D + C + H + 2D + C Good	H + 2D + 4RC + 2D Good

	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) delete
(1) Heap	BD Cannot do better	0.5BD Not good	BD Not good	2D Good	Search (b) + D Not good
(2) Sorted (contiguous)	BD Cannot do better	Dlog ₂ B Good	Dlog ₂ B + + matching pages*D Good	Search(b) + BD Not good	Search(b) + BD Not good
(3) Clustered B+ tree	1.5BD Cannot do better	Dlog _F .15B Good	Dlog _F .15B + + mp*D Good	Search + D Good	Search + D Good
(4) Unclustered B+ Tree Index (Alt 2)	1.5BD Cannot do better	DLog _F 0.15B + D Good	DLog _F 0.15B + mp*R*D Not Good	DLog _F 0.15B + D + 2D	Search(b) + 2D (index + data write)
(5) Unclustered Hash index	1.25BD Cannot do better	2D Good	1.25BD Not good	4D Good	Search + 4D Good

Cost of Operations					
	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) Delete
(1) Heap	BD	0.5BD	BD	2D	Search +D
(2) Sorted	BD	Dlog 2B	Dlog 2 B + # matches	Search + BD	Search +BD
(3) Clustered	1.5BD	Dlog f 1.5B	Dlog F 1.5B + # matches	Search + D	Search +D
(4) Unclustered Tree index	BD(R+0.15)	D(1 + log f 0.15B)	Dlog F 0.15B + # matches		Search + 2D
(5) Unclustered Hash index	BD(R+0.1 25)	2D	BD	4D	Search + 2D
Several assumptions underlie these (rough) estimates!					

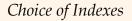


• which attributes are involved in selection/join conditions How selective are these conditions likely to be?

20

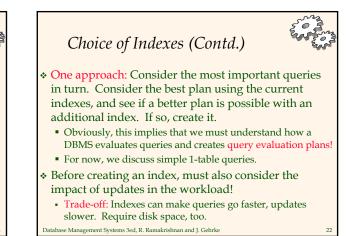
• The type of update (INSERT/DELETE/UPDATE), and the attributes that are affected.

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke



- * What indexes should we create?
 - Which relations should have indexes? What field(s) should be the search key? Should we build several indexes?
- For each index, what kind of an index should it be?
 - Clustered? Hash/tree?

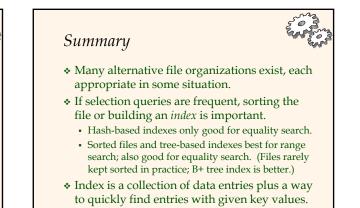
se Management Systems 3ed, R. Ramakrishnan and I. G



Index Selection Guidelines

- * Attributes in WHERE clause are candidates for index keys.
 - Exact match condition suggests hash index.
 - Range query suggests tree index.
 - Clustering is especially useful for range queries; can also help on equality queries if there are many duplicates.
- Multi-attribute search keys should be considered when a WHERE clause contains several conditions.
 - · Order of attributes is important for range queries.
 - Such indexes can sometimes enable index-only strategies for important queries.
- For index-only strategies, clustering is not important!
 Try to choose indexes that benefit as many queries as
- possible. Since only one index can be clustered per relation, choose it wisely based on important queries that would benefit the most from clustering.

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke



Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

Summary (Contd.)



- Data entries can be actual data records, <key, rid> pairs, or <key, rid-list> pairs.
 - Choice orthogonal to *indexing technique* used to locate data entries with a given key value.
- Can have several indexes on a given file of data records, each with a different search key.
- Indexes can be classified as clustered vs. unclustered, primary vs. secondary, and dense vs. sparse. Differences have important consequences for utility/performance.

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

Summary (Contd.)

- Understanding the nature of the *workload* for the application, and the performance goals, is essential to developing a good design.
 - What are the important queries and updates? What attributes/relations are involved?
- Indexes must be chosen to speed up important queries (and perhaps some updates!).
 - Index maintenance overhead on updates to key fields.
 - Choose indexes that can help many queries, if possible.
 - Build indexes to support index-only strategies.
 - Clustering is an important decision; only one index on a given relation can be clustered!
 - Order of fields in composite index key can be important.

atabase Management Systems 3ed, R. Ramakrishnan and J. Gehrke

