

Hash-Based Indexes

Chapter 11

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 1

Index Design Space

Organization Structure for k^*

- **Hash-based**
 - + Equality search
 - Static hashing
 - Extendible hashing
 - Linear hashing
- **Tree-based**
 - + Range, equality search

Data Entry (k^*) Contents

1. Actual Data record
 - index = file
2. $\langle k, rid \rangle$
 - actual records in a different file
3. $\langle k, list\ of\ rids \rangle$

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 2

Static Hashing

- # primary bucket pages fixed, allocated sequentially, never de-allocated; overflow pages if needed.
- $h(key) \bmod N =$ bucket to which data entry with key belongs. ($N =$ # of buckets)

The diagram shows a key being hashed by a function h . The result $h(key) \bmod N$ points to a bucket in a vertical list of primary bucket pages labeled 0, 1, ..., N-1. Arrows from each bucket point to a corresponding overflow page. A yellow box asks: "What are important characteristics of $h()$?"

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 3

Static Hashing (Contd.)

- Buckets contain *data entries*
- Number of buckets (N) is fixed ahead of time
- Static structure can be problematic
 - Consider many insertions
 - Long overflow chains can develop (and degrade performance!)
- Might consider periodically doubling N and "rehashing" file
 - Entire file has to be read and written
 - Index unavailable while rehashing
 - *Extendible* and *Linear Hashing*: Dynamic techniques to fix this problem.

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 4

Extendible Hashing

- **Main Idea:** Use a directory of pointers to buckets
- On overflow, double the directory (not # number of buckets)
- Why does this help?
 - Directory much smaller than file
 - Only one page of data entries is split at a time
 - No overflow pages

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 5

Example

- Directory an array
- Search for k:
 - Apply hash function $h(k)$
 - Take last *global depth* # bits of $h(k)$
- Insert:
 - If bucket has space, insert, done
 - If bucket if full, split it, re-distribute
 - If necessary, double the directory

LOCAL DEPTH

GLOBAL DEPTH

DIRECTORY

DATA PAGES

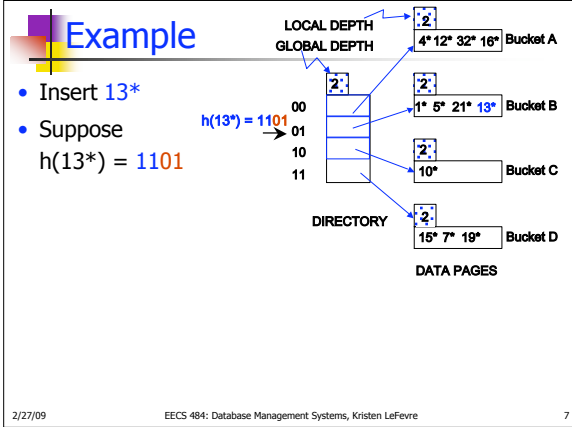
00 → Bucket A (4° 12° 32° 16°)

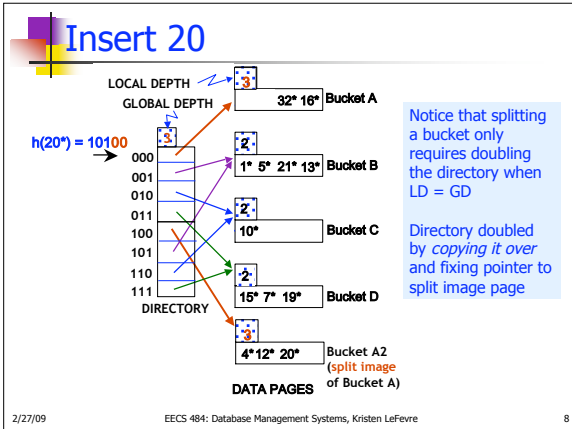
01 → Bucket B (1° 5° 21°)

10 → Bucket C (10°)

11 → Bucket D (15° 7° 19°)

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 6





- ### Comments on Extendible Hashing
- How many disk accesses for equality search?
 - One if directory fits in memory, else two.
 - Directory grows in spurts, and, if the distribution of hash values is skewed, directory can grow large.
 - Do we ever need overflow pages?
 - Multiple entries with same hash value cause problems!
 - **Delete:** Reverse of inserts – see textbook.
- 2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 9

Exercise

- Suppose our DBMS implements extendible hash indexes and B+ trees (both using data entry Alternative 2). We know that hash indexes are no good for range queries. Can you think of a case (database and one or more query) where the hash index is better?
 - Describe the database and queries as precisely as possible.
 - Calculate the I/O cost for each index.

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 10

Linear Hashing

- Another dynamic hashing scheme
- Eliminates long overflow chains without using a directory.
- **Main idea:** Use a family of hash functions h_0, h_1, h_2, \dots
 - h_{i+1} doubles the range of h_i (similar to directory doubling)
 - Hash family typically obtained by choosing hash function $h()$ and initial number of buckets N
 - Define $h_i(value) = h(value) \bmod (2^i N)$
 - If N is a power of 2, apply hash function $h()$, and look at last d_i bits
 - d_0 number of bits needed to represent N
 - $d_i = d_0 + 1$

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 11

Linear Hashing

- Splitting proceeds in **rounds**
 - During round $level$, only h_{level} and $h_{level+1}$ are in use
- Variables
 - **Level:** Initialized to 0
 - **Next:** Pointer to the bucket being split
- At the beginning of round # **Level**, the # buckets in the file = $N * 2^{Level}$
 - N is initial number of buckets

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 12

Buckets During Round i

Range of h_i

Use h_{i+1} in this range

← Next
Split image buckets
(created by splitting other buckets
in this round)

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 13

Linear Hashing (Contd.)

- **Search:** Find bucket with key value k^* by applying h_{Level}
 - If this leads to an un-split bucket, all set
 - Otherwise, apply $h_{Level+1}$

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 14

Linear Hashing (Contd.)

- **Insert:** Find bucket by applying $h_{Level} / h_{Level+1}$:
 - If bucket being inserted into is full:
 - Add overflow page and insert data entry.
 - (Maybe) Split *Next* bucket and increment *Next*.
- *Can choose any criterion to trigger split.*
 - e.g. split on a overflow (as in example)
 - e.g. space utilization on the page > 90%
- Since buckets are split round-robin, long overflow chains don't develop!
- **Deletes:** see textbook

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 15

Linear Hashing Example

Level 0, N = 4

	H_0	
	00	Next=0 32* 44* 36* □
	01	9* 25* 5* □
	10	14* 18* 10* 30*
	11	31* 35* 7* 11*

Primary bucket Pages in the Hash File

This is not actually stored

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 16

Linear Hashing: Insert 43

Level 0, N = 4

	H_1	H_0	
	000	00	32* □ □ □
	001	01	Next=1 9* 25* 5* 37*
	010	10	14* 18* 10* 30*
	011	11	31* 35* 7* 11* → 43* □ □ □
	100		44* 36* □ □
	...	not actually stored	

Primary bucket Pages in the Hash File


What happens if we now insert an entry into bucket 01?

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 17

Summary

- Discussed 3 kinds of hash-based indexes
- Static Hashing can lead to long overflow chains.
- Extendible Hashing
 - Directory to keep track of buckets, doubles periodically.
 - Always splits the "right" bucket.
- Linear Hashing
 - Split buckets round-robin, and use overflow pages.
 - Duplicates handled easily.
 - Space utilization could be lower than EH.

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 18

 **Announcements**

- Review problems for hash indexes
 - 11.1, 11.3, 11.7, 11.9

2/27/09 EECS 484: Database Management Systems, Kristen LeFevre 19
