Sheet01: CHAPTER 13: DISK STORAGE, BASIC FILE STRUCTURES, AND HASHING

13.23 Consider a disk with the following: block size B=512 bytes, interblock gap size G=128 bytes, number of blocks per track=20, number of tracks per surface=400. A disk pack consists of 15 double-sided disks.

(a) What is the total capacity of a track and what is its useful capacity?

Total track size = 20 * (512+128) = 12800 bytes = 12.8 Kbytes
Useful capacity of a track = 20 * 512 = 10240 bytes = 10.24 Kbytes

(b) How many cylinders are there?

Number of cylinders = number of tracks = 400

(c) What is the total capacity and the useful capacity of a cylinder?

Total cylinder capacity = 15*2*20*(512+128) = 384000 bytes = 384 Kbytes
Useful cylinder capacity = 15 * 2 * 20 * 512 = 307200 bytes = 307.2 Kbytes

(d) What is the total capacity and the useful capacity of a disk pack?

Total capacity of a disk pack = 15 * 2 * 400 * 20 * (512+128) = 153600000 bytes = 153.6 Mbytes
Useful capacity of a disk pack = 15 * 2 * 400 * 20 * 512 = 122.88 Mbytes

(e) Suppose the disk drive rotates the disk pack at a speed of 2400 rpm (revolutions per minute); what is the transfer rate in bytes/msec and the block transfer time btt in msec? What is the average rotational delay rd in msec? What is the bulk transfer rate?

Transfer rate tr= (total track size in bytes)/(time for one disk revolution in msec)

tr= (12800) / ((60 * 1000) / (2400)) = (12800) / (25) = 512 bytes/msec

block transfer time btt = B / tr = 512 / 512 = 1 msec

average rotational delay rd = (time for one disk revolution in msec) / 2 = 25 / 2 = 12.5 msec

bulk transfer rate btr= tr * (B/(B+G)) = 512*(512/640) = 409.6 bytes/msec

(f) Suppose the average seek time is 30 msec. How much time does it take (on the average) in msec to locate and transfer a single block given its block address?

average time to locate and transfer a block = s+rd+btt = 30+12.5+1 = 43.5 msec

(g) Calculate the average time it would take to transfer 20 random blocks and compare it with the time it would take to transfer 20 consecutive blocks using double buffering to save seek time and rotational delay.

time to transfer 20 random blocks = 20 * (s + rd + btt) = 20 * 43.5 = 870 msec

time to transfer 20 consecutive blocks using double buffering = s + rd + 20*btt
= 30 + 12.5 + (20*1) = 62.5 msec

13.24 A file has r=20000 STUDENT records of fixed-length. Each record has the following fields: NAME (30 bytes), SSN (9 bytes), ADDRESS (40 bytes), PHONE (9 bytes), BIRTHDATE (8 bytes), SEX (1 byte), MAJORDEPTCODE (4 bytes), MINORDEPTCODE (4 bytes), CLASSCODE (4 bytes, integer), and DEGREEPROGRAM (3 bytes). An additional byte is used as a deletion marker. The file is stored on the disk whose parameters are given in previous exercise.

(a) Calculate the record size R in bytes.

R = (30 + 9 + 40 + 9 + 8 + 1 + 4 + 4 + 4 + 3) + 1 = 113 bytes

(b) Calculate the blocking factor bfr and the number of file blocks b assuming an unspanned organization.

bfr = floor(B / R) = floor(512 / 113) = 4 records per block

b = ceiling(r / bfr) = ceiling(20000 / 4) = 5000 blocks

(c) Calculate the average time it takes to find a record by doing a linear search on the file if (i) the file blocks are stored contiguously and double buffering is used, and (ii) the file blocks are not stored contiguously.

For linear search we search on average half the file blocks= 5000/2= 2500 blocks.

i. If the blocks are stored consecutively, and double buffering is used, the time to read 2500 consecutive blocks
= s+rd+(2500*btt)= 30+12.5+2500*1= 2542.5 msec

ii. If the blocks are scattered over the disk, a seek is needed for each block, so the time is: 2500 * (s + rd + btt) = 2500 * (30 + 12.5 + 1) = 108750 msec = 108.75 sec

(d) Assume the file is ordered by SSN; calculate the time it takes to search for a record given its SSN value by doing a binary search.

For binary search, the time to search for a record is estimated as:
ceiling($\log_2$ b) * (s +rd + btt)

= ceiling($\log_2$ 5000) * (30 + 12.5 + 1) = 13 * 43.5 = 565.5 msec = 0.5655 sec

13.25 Suppose only 80% of the STUDENT records from Exercise 13.24 have a value for PHONE, 85% for MAJORDEPTCODE, 15% for MINORDEPTCODE, and 90% for DEGREEPROGRAM, and we use a variable-length record file. Each record has a 1-byte field type for each field occurring in the record, plus the 1-byte deletion marker and a 1-byte end-of record marker. Suppose we use a spanned record organization, where each block has a 5-byte pointer to the next block (this space is not used for record storage).

(a) Calculate the average record length R in bytes.

Assuming that every field has a 1-byte field type, and that the fields not mentioned above (NAME, SSN, ADDRESS, BIRTHDATE, SEX, CLASSCODE) have values in every record, we need the following number of bytes for these fields in each record, plus 1 byte for the deletion marker, and 1 byte for the end-of-record marker:
R fixed = (30+1) + (9+1) + (40+1) + (8+1) + (1+1) + (4+1) +1+1 = 100 bytes

For the fields (PHONE, MAJORDEPTCODE, MINORDEPTCODE DEGREEPROGRAM), the average number of bytes per record is:

R variable = ((9+1)*0.8)+((4+1)*0.85)+((4+1)*0.15)+((3+1)*0.9)
= 8+4.25+0.75+3.6= 16.6 bytes

The average record size R = R fixed + R variable = 100 + 16.6 = 116.6 bytes
The total bytes needed for the whole file = r * R = 20000 * 116.6 = 2332000 bytes

(b) Calculate the number of blocks needed for the file.

Using a spanned record organization with a 5-byte pointer at the end of each block, the bytes available in each block are (B-5) = (512 - 5) = 507 bytes.
The number of blocks needed for the file are:
b = ceiling((r * R) / (B - 5)) = ceiling(2332000 / 507) = 4600 blocks
(compare this with the 5000 blocks needed for fixed-length, unspanned records in Problem 4.19(b))

13.26 Suppose that a disk unit has the following parameters: seek time s=20 msec; rotational delay rd=10 msec; block transfer time btt=1 msec; block size B=2400 bytes; interblock gap size G=600 bytes. An EMPLOYEE file has the following fields: SSN, 9 bytes; LASTNAME, 20 bytes; FIRSTNAME, 20 bytes; MIDDLE INIT, 1 byte; BIRTHDATE, 10 bytes; ADDRESS, 35 bytes); PHONE, 12 bytes); SUPERVISORSSN, 9 bytes; DEPARTMENT, 4 bytes; JOBCODE, 4 bytes; deletion marker, 1 byte. The EMPLOYEE file has r=30000 STUDENT records, fixed-length format, and unspanned blocking. Write down appropriate formulas and calculate the following values for the above EMPLOYEE file:

(a) The record size R (including the deletion marker), the blocking factor bfr, and the number of disk blocks b.

R = (9 + 20 + 20 + 1 + 10 + 35 + 12 + 9 + 4 + 4) + 1 = 125 bytes

bfr = floor(B / R) = floor(2400 / 125) = 19 records per block
b = ceiling(r / bfr) = ceiling(30000 / 19) = 1579 blocks

(b) Calculate the wasted space in each disk block because of the unspanned organization.

Wasted space per block = B - (R * Bfr) = 2400 - (125 * 19) = 25 bytes

(c) Calculate the transfer rate tr and the bulk transfer rate btr for this disk

Transfer rate tr= B/btt = 2400 / 1 = 2400 bytes/msec
bulk transfer rate btr= tr * (B/(B+G))
= 2400*(2400/(2400+600)) = 1920 bytes/msec

(d) Calculate the average number of block accesses needed to search for an arbitrary record in the file, using linear search.

For linear search we have the following cases:
i. search on key field:
if record is found, half the file blocks are searched on average: b/2= 1579/2 blocks
if record is not found, all file blocks are searched: b = 1579 blocks
ii. search on non-key field:
all file blocks must be searched: b = 1579 blocks

(e) Calculate the average time needed in msec to search for an arbitrary record in the file, using linear search, if the file blocks are stored on consecutive disk blocks and double buffering is used.

If the blocks are stored consecutively, and double buffering is used, the time to read n consecutive blocks= s+rd+(n*btt)
i. if n=b/2: time = 20+10+((1579/2)*1)= 819.5 msec

ii. if n=b: time = s+rd+(n*btt)= 20+10+1579*1= 1609 msec

(f) Calculate the average time needed in msec to search for an arbitrary record in the file, using linear search, if the file blocks are not stored on consecutive disk blocks.

If the blocks are scattered over the disk, a seek is needed for each block, so the time to search n blocks is: n * (s + rd + btt)
i. if n=b/2: time = (1579/2)*(20+10+1)= 24474.5 msec = 24.475 sec
ii. if n=b: time = 1579*(20+10+1)= 48949 msec = 48.949 sec

(g) Assume that the records are ordered via some key field. Calculate the average number of block accesses and the average time needed to search for an arbitrary record in the file, using binary search.

For binary search, the time to search for a record is estimated as:
ceiling($\log_2$ b) * (s +rd + btt)
= ceiling($\log_2$ 1579) * (20+10+1) = 11 * 31 = 341 msec = 0.341 sec

13.27 A PARTS file with Part# as hash key includes records with the following Part# values: 2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, 6975, 4981, 9208. The file uses 8 buckets, numbered 0 to 7. Each bucket is one disk block and holds two records. Load these records into the file in the given order using the hash function h(K)=K mod 8. Calculate the average number of block accesses for a random retrieval on Part#.

Answer:

K h(K) (bucket number)	Bucket	records		overflow
2369 1	0	3760		2428
3760 0		9208		6975
4692 4	1	2369		
4871 7		9209		
5659 3	2	1074		
1821 5				
1074 2	3	5659		
7115 3		7115		
1620 4	4	4692		
2428 4 Overflow		1620		
3943 7	5	1821		
4750 6		4981		
6975 7 Overflow	6	4750		
4981 5				
9208 0	7	4871		
9209 1		3943		

Two records out of 15 are in overflow, which will require an additional block access. The other records require only one block access. Hence, the average time to retrieve a random record is:
(1 * (13/15)) + (2 * (2/15)) = 0.867 + 0.266 = 1.133 block accesses

13.28 Load the records of Exercise 13.27 into expandable hash files based on extendible hashing. Show the structure of the directory at each step. Show the directory at each step, and the global and local depths. Use the has function
h(k) = K mod 32.

Answer:

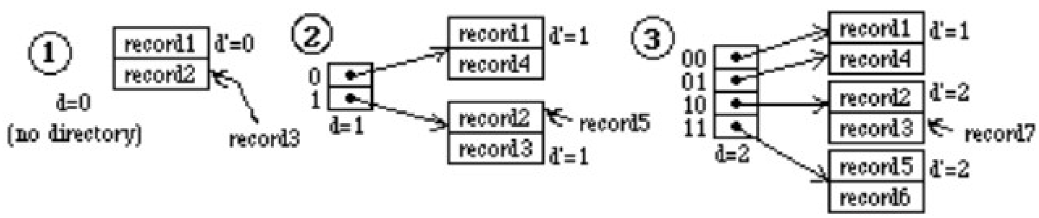
Hashing the records gives the following result:

Sheet01: Chapter 13: Disk Storage, Basic File Structures, and Hashing

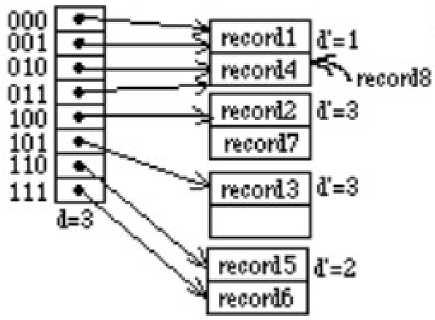
	K	h(K) (bucket number)	binary h(K)
record1	2369	1	00001
record2	3760	16	10000
record3	4692	20	10100
record4	4871	7	00111
record5	5659	27	11011
record6	1821	29	11101
record7	1074	18	10010

record8	7115	11	01011
record9	1620	20	10100
record10	2428	28	11100
record11	3943	7	00111
record12	4750	14	01110
record13	6975	31	11111
record14	4981	21	10101
record15	9208	24	11000

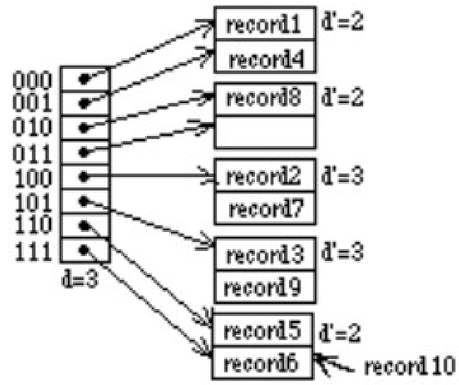
Extendible hashing:



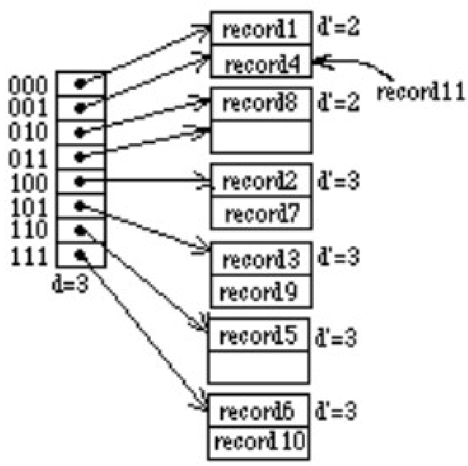
4


000
001
010
011
100
101
110
111
d=3
record1
record4
d'=1
record8
record2
record7
d'=3
record3
d'=3
record5
record6
d'=2

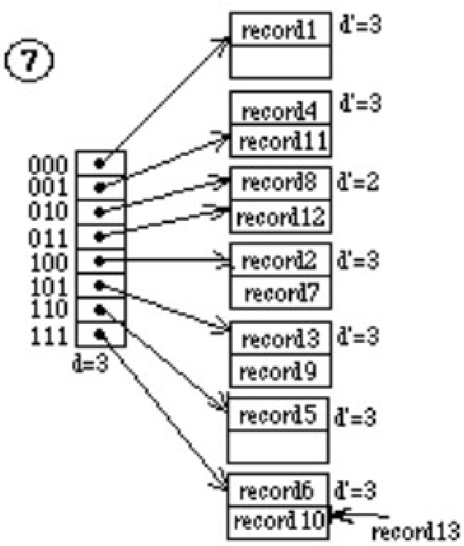
5


000
001
010
011
100
101
110
111
d=3
record1
record4
d'=2
record8
d'=2
record2
record7
d'=3
record3
record9
d'=3
record5
record6
d'=2
record10

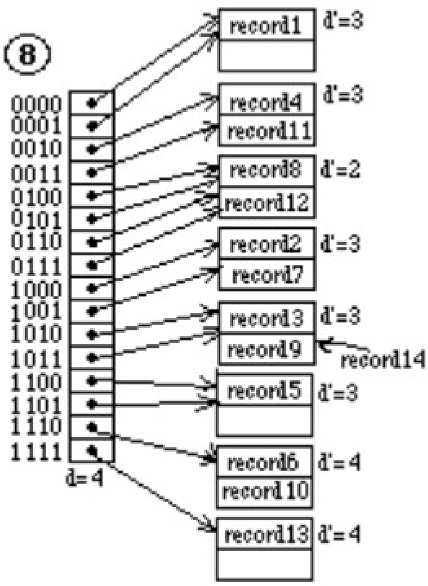
6


000
001
010
011
100
101
110
111
d=3
record1
record4
d'=2
record11
record8
d'=2
record2
record7
d'=3
record3
record9
d'=3
record5
d'=3
record6
record10
d'=3

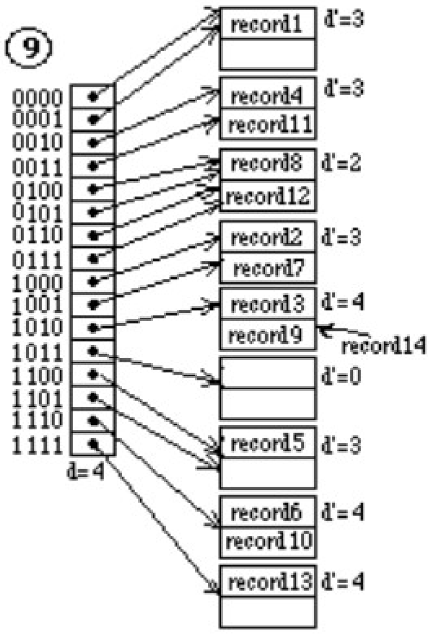
7


000
001
010
011
100
101
110
111
d=3
record1
d'=3
record4
record11
d'=3
record8
record12
d'=2
record2
record7
d'=3
record3
record9
d'=3
record5
d'=3
record6
record10
d'=3
record13

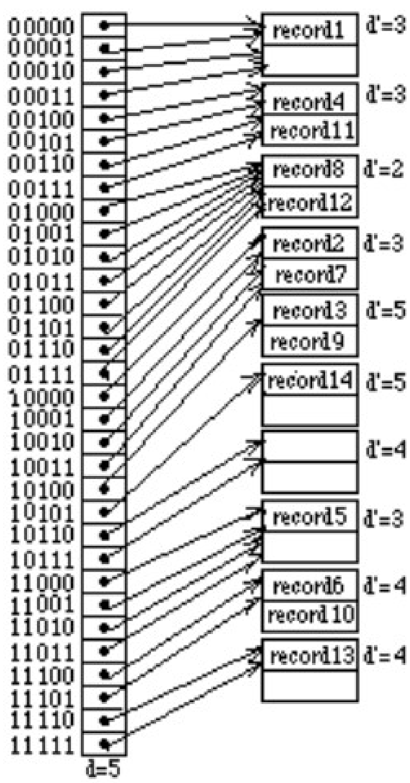
8


0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
d= 4
record1 d'=3
record4 d'=3
record11
record8 d'=2
record12
record2 d'=3
record7
record3 d'=3
record9
record14
record5 d'=3
record6 d'= 4
record 10
record13 d'= 4

9


0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
d=4
record1 d'=3
record4 d'=3
record11
record8 d'=2
record12
record2 d'=3
record7
record3 d'= 4
record9
record14
d'=0
record5 d'=3
record6 d'= 4
record 10
record13 d'= 4

10


00000
00001
00010
00011
00100
00101
00110
00111
01000
01001
01010
01011
01100
01101
01110
01111
10000
10001
10010
10011
10100
10101
10110
10111
11000
11001
11010
11011
11100
11101
11110
11111
d=5
record1 d'=3
record4 d'=3
record11
record8 d'=2
record12
record2 d'=3
record7
record3 d'=5
record9
record14 d'=5
d'=4
record5 d'=3
record6 d'=4
record 10
record13 d'=4