

인덱스란?

레코드를 **빠르게** 찾을 수 있도록
도와주는 보조 파일

화일에 대한 또 다른 접근 경로

인덱스 엔트리

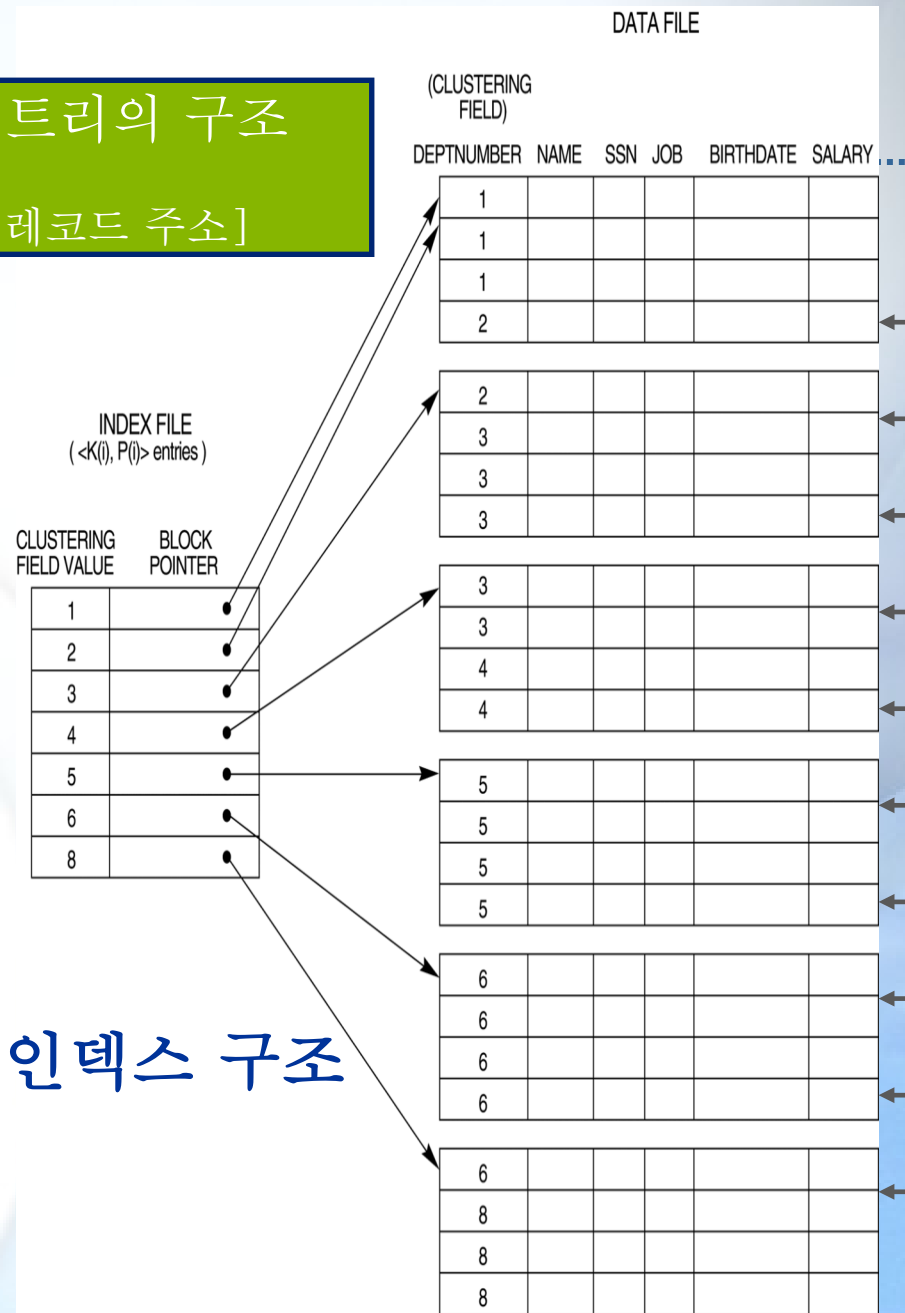
인덱스 화일의 레코드

인덱스 엔트리의 구조

[키 값들, 레코드 주소]

인덱스 엔트리의 구조

[키 값들, 레코드 주소]



일반적인 인덱스 구조

1. 인덱스에서 해당 엔트리를 찾는다
2. 블록포인터가 가리키는 블록부터 원하는 레코드를 찾는다.



검색할 때 인덱스를 사용하는 것이 더 좋은가?

검색 : 순서화일 vs 인덱스

순서화일의 **처음부터** 순차검색 한다.

Search 1

Search 3

Search 8

DEPTNUMBER NAME SSN JOB BIRTHDATE SALARY

1					
1					
1					
2					

2					
3					
3					
3					

3					
3					
4					
4					

5					
5					
5					
5					

6					
6					
6					
6					

6					
8					
8					
8					

1. 인덱스에서 해당 엔트리를 찾는다
2. **블록포인터가 가리키는 블록부터** 순차검색 한다.

DATA FILE

(CLUSTERING
FIELD)

DEPTNUMBER NAME SSN JOB BIRTHDATE SALARY

1					
1					
1					
2					

2					
3					
3					
3					

3					
3					
4					
4					

5					
5					
5					
5					

6					
6					
6					
6					

6					
8					
8					
8					

INDEX FILE
(<K(i), P(i)> entries)

CLUSTERING
FIELD VALUE BLOCK
POINTER

1	
2	
3	
4	
5	
6	
8	

삽입 알고리즘

1. 삽입될 자리를 찾는다
2. 공간이 없으면 공간을 만든다.
3. 삽입한다.
4. 필요하면 인덱스에 반영한다.

3.5
삽입

INDEX FILE
($\langle K(i), P(i) \rangle$ entries)

CLUSTERING
FIELD VALUE BLOCK
POINTER

1	•
2	•
3	•
4	•
5	•
6	•
8	•

DATA FILE

(CLUSTERING FIELD)	DEPTNUMBER	NAME	SSN	JOB	BIRTHDATE	SALARY
	1					
	1					
	1					
	2					
	2					
	3					
	3					
	3					
	3					
	5					
	5					
	5					
	5					
	6					
	6					
	6					
	6					
	6					
	8					
	8					
	8					

3.5를 삽입

4	
4	

일반적인 인덱스 구조

삽입 알고리즘

1. 삽입될 자리를 찾는다
2. 공간이 없으면 공간을 만든다.
3. 삽입한다.
4. 필요하면 인덱스에 반영한다.

인덱스에서의 분할

3.5 삽입

INDEX FILE
($\langle K(i), P(i) \rangle$ entries)

CLUSTERING
FIELD VALUE BLOCK
POINTNER

1	
2	
3	
3.5	

4	
5	
6	
8	

DATA FILE

(CLUSTERING
FIELD)

DEPTNUMBER NAME SSN JOB BIRTHDATE SALARY

1					
1					
1					
2					

2					
3					
3					
3					

3					
3					
3.5					

5					
5					
5					
5					

6					
6					
6					
6					

6					
8					
8					
8					

3.5를 삽입

4		
4		

일반적인 인덱스 구조

삽입 알고리즘

1. 삽입될 자리를 찾는다
2. 공간이 없으면 공간을 만든다.
3. 삽입한다.
4. 필요하면 인덱스에 반영한다.

인덱스에서의 분할

INDEX FILE
($\langle K(i), P(i) \rangle$ entries)

CLUSTERING
FIELD VALUE BLOCK
POINTNER

1	
2	
3	
3.5	

4	
5	
6	
8	

3.5
삽입

DATA FILE

(CLUSTERING
FIELD)

DEPTNUMBER NAME SSN JOB BIRTHDATE SALARY

1					
1					
1					
2					

2					
3					
3					
3					

3					
3					
3.5					

5					
5					
5					
5					

6					
6					
6					
6					

6					
8					
8					
8					

3.5를 삽입

4		
4		

삽입할 때 인덱스를 사용하면 더 좋은
가?

1. 삭제할 레코드를 찾는다
2. 삭제한다.
3. 필요하면 삭제한 공간을 처리한다.
4. 필요하면 인덱스에 반영한다.

CLUSTERING FIELD VALUE	BLOCK POINTER
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99

1	
2	
3	
4	
5	
6	
8	

일반적인 인덱스 구조

DATA FILE

(CLUSTERING FIELD)

DEPTNUMBER	NAME	SSN	JOB	BIRTHDATE	SALARY
1					
1					
1					
2					
2					
3					
3					
3					
3					
3					
3					
4-1					
4-2					
5					
5					
5					
5					
6					
6					
6					
6					
6					
8					
8					
8					
8					

삭제 알고리즘

1. 삭제할 레코드를 찾는다
2. 삭제한다.
3. 필요하면 삭제한 공간을 처리한다.
4. 필요하면 인덱스에 반영한다.

INDEX FILE
(<K(i), P(i)> entries)

Delete 4-1

CLUSTERING FIELD VALUE	BLOCK POINTER
1	•
2	•
3	•
4	•
5	•
6	•
8	•

DATA FILE

(CLUSTERING FIELD)	DEPTNUMBER	NAME	SSN	JOB	BIRTHDATE	SALARY
1						
1						
1						
2						
2						
3						
3						
3						
3						
4-1						
4-2						
5						
5						
5						
5						
6						
6						
6						
6						
6						
8						
8						
8						
8						

일반적인 인덱스 구조

삭제 알고리즘

1. 삭제할 레코드를 찾는다
2. 삭제한다.
3. 필요하면 삭제한 공간을 처리한다.
4. 필요하면 인덱스에 반영한다.

INDEX FILE
(<K(i), P(i)> entries)

Delete 4-1

Delete 4-2

CLUSTERING FIELD VALUE	BLOCK POINTER
1	•
2	•
3	•
4	•
5	•
6	•
8	•

DATA FILE

(CLUSTERING
FIELD)

DEPTNUMBER NAME SSN JOB BIRTHDATE SALARY

1					
1					
1					
2					

2					
3					
3					
3					

3					
3					
4-1					
4-2					

5-1					
5-2					
5					
5					

6					
6					
6					
6					

6					
8					
8					
8					

일반적인 인덱스 구조

삭제 알고리즘

1. 삭제할 레코드를 찾는다
2. 삭제한다.
3. 필요하면 삭제한 공간을 처리한다.
4. 필요하면 인덱스에 반영한다.

INDEX FILE
(<K(i), P(i)> entries)

Delete 4-1

Delete 4-2

Delete 5-1

Delete 5-2

Delete 5-3

CLUSTERING
FIELD VALUE BLOCK
POINTER

1	•
2	•
3	•
4	•
5	•
6	•
8	•

DATA FILE

(CLUSTERING
FIELD)

DEPTNUMBER NAME SSN JOB BIRTHDATE SALARY

1					
1					
1					
2					
2					
3					
3					
3					
4-1					
4-2					
5-1					
5-2					
5-3					
5					
6					
6					
6					
6					
8					
8					
8					
8					

병합

일반적인 인덱스 구조

삭제 알고리즘

1. 삭제할 레코드를 찾는다
2. 삭제한다.
3. 필요하면 삭제한 공간을 처리한다.
4. 필요하면 인덱스에 반영한다.

INDEX FILE
(<K(i), P(i)> entries)

Delete 4-1

Delete 4-2

Delete 5-1

Delete 5-2

Delete 5-3

CLUSTERING
FIELD VALUE BLOCK
POINTNER

1		•
2		•
3		•
4		•
5		•
6		•
8		•

DATA FILE

(CLUSTERING
FIELD)

DEPTNUMBER NAME SSN JOB BIRTHDATE SALARY

1					
1					
1					
2					

2					
3					
3					
3					

3					
3					
4-1					
4-2					

5-1					
5-2					
5-3					
5					

6					
6					
6					
6					

6					
8					
8					
8					

병합

일반적인 인덱스 구조

삭제할 때 인덱스를 사용하는 것이 더 좋은가?

생각해 봅시다.



1. 인덱스 화일의 크기는 어느 정도 일까?
2. 인덱스 엔트리의 개수는 몇 개인가?
3. 하나의 화일에 몇 개의 인덱스를 만들수 있을까?