



13. File organization

*Database는 OS 위에서 하버의 스위치 돌아감 → 다른 program과 똑같이 file은 저장된다!
↳ 구조를 알아보자.

▼ File organization

① database → file의 집합으로 저장됨

② 각 file → record의 sequence
↳ record sequence

• file 구성 방법 가정 → 간단/단일화된 방법 알아보자 → 이후, variable length도 알아보자

1. fixed record size → 쉬운 고정
2. 한 가지 type의 file만 존재
3. 다른 file은 다른 relation 조작 가능
4. record는 disk block보다 작음 → 한 page에 모두 기록 가능

▼ Fixed-Length Records

• simple approach : 순차적으로 기록

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

◦ record size = n일 때

▪ i번째 record는 $n \cdot (i-1)$ byte에 저장됨

◦ record 접근은 간단하지만 block 여기 저기에 저장될 수 있음

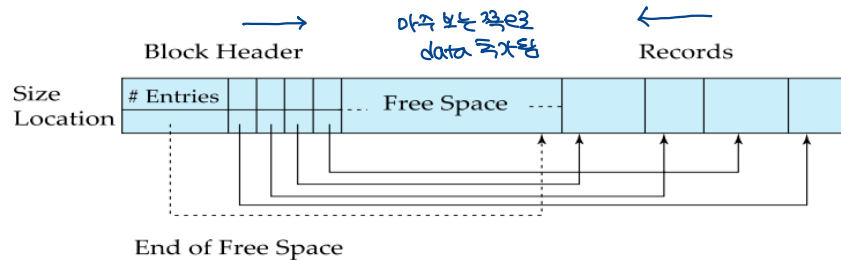
↳ boundary 넘어가지 않게 관리해야함
↳ SSD, HDD I/O 시간 비교 결정

1. Deletion of i

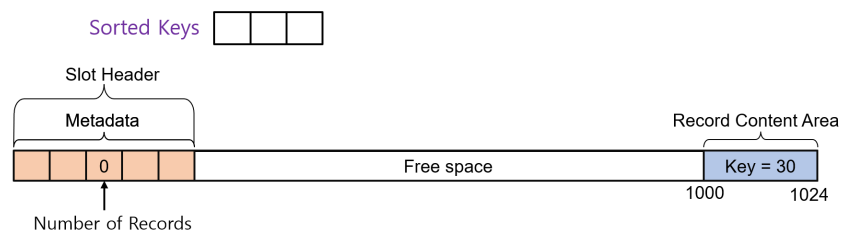
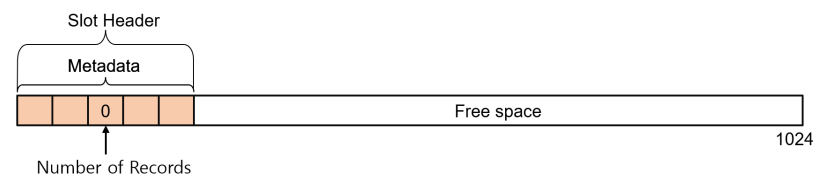
i. 하나씩 shift

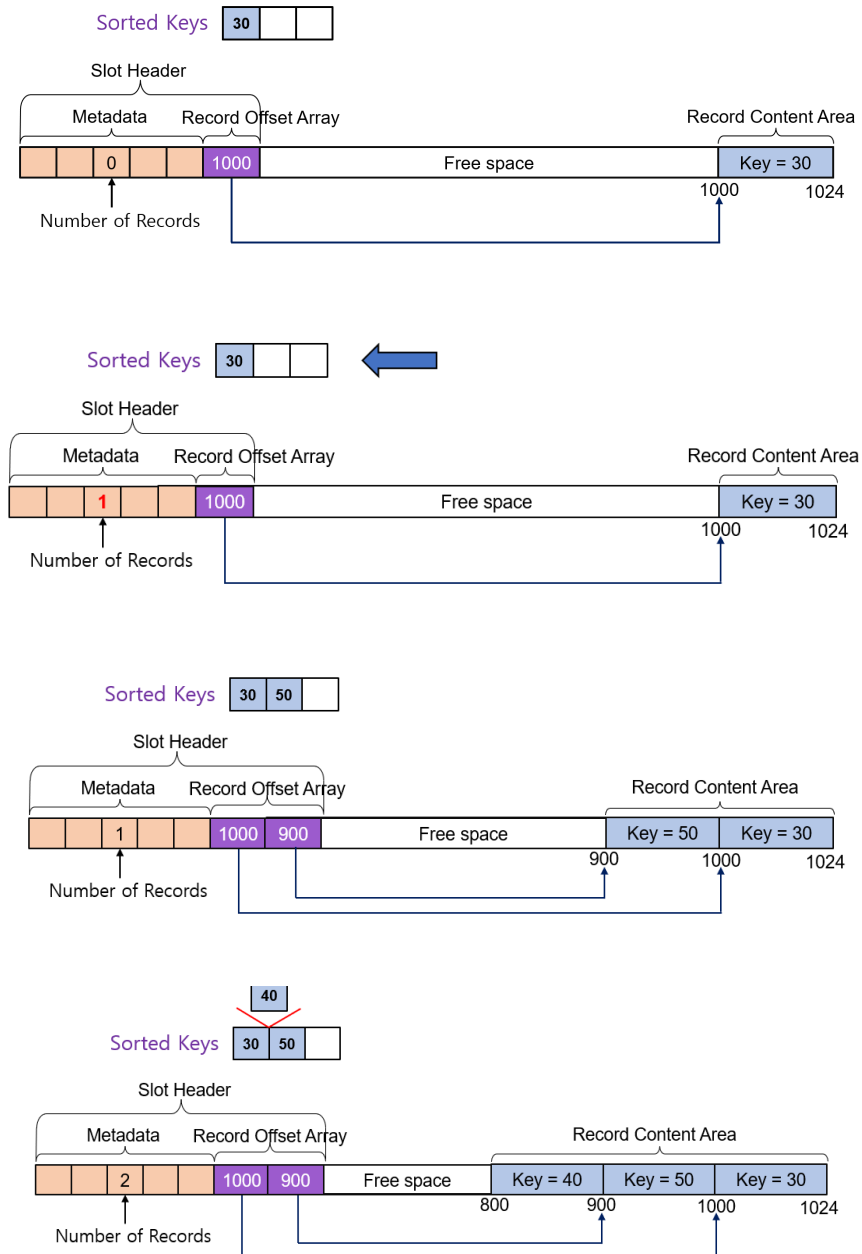
- attribute : 정렬된 상태로 저장됨
 - <offset, length> pair로 저장
 - null : null-value bitmap 나타냄

▼ Slotted page structure



- 구조
 - Slotted page header
 - record 개수
 - free space의 끝
 - 각 record의 location, size
 - Record
 - page 내에서 이동하여 빈 공간 없이 연속적으로 유지 가능
 - header의 항목을 update해야 함
 - Pointer
 - header 안에 있는 record를 point
- insertion example





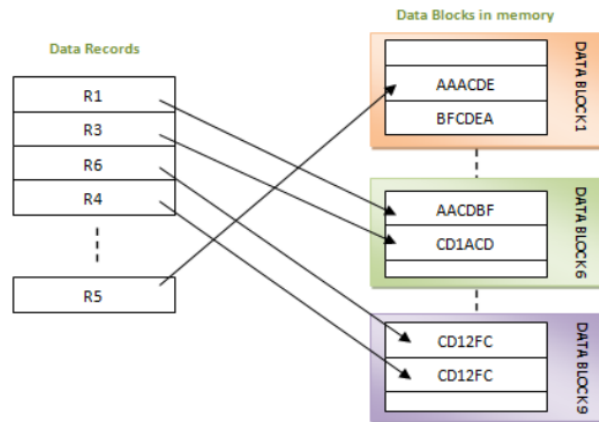
▼ Storing Large Objects

blob/clob type 같은 것이 있음

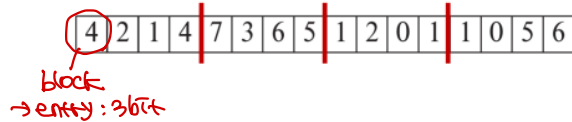
- 원래 우리의 가정 → record는 무조건 page보다 작다
- 대안책을 위한 가정
 1. file은 file system 내부에 저장
 2. database에 의해 관리되는 file을 저장
 3. 조각조각 나누어 분리된 relation 내에 여러 tuple 저장

▼ Organization of Records in Files

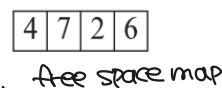
① Heap



- record는 어디에나 저장 가능 → 한 번 할당되면 움직이지 x
- free한 공간이 어디에 있는지 찾는 것이 중요
 - Free-space map → 영역별로 나누어서 표기
 - block 당 bit 단위인 entry 1개인 array
 - record: free한 block의 fraction
 - example



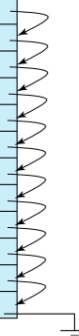
- 한 block : 3bit entry를 가짐
- 8로 나뉜 value → free한 block의 fraction 나타냄 ⇒ record를 나타냄!
- second-level free-space map
 - example



- 각 entry : first-level의 free space map의 최대 4개 entry를 저장
- ⇒ 병렬적으로 작성됨 → 몇개의 entry 잘못되어도 괜찮음(나중에 고쳐짐)

② sequential


10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	



- 전체 file의 sequential한 processing을 필요할 때 적합
- file 안의 record들은 search-key로 정렬됨 → operation 적용 후 reorganize 필요
- operation

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

32222	Verdi	Music	48000	
-------	-------	-------	-------	--



- deletion → pointer chain 사용
- insetion → record가 삽입될 위치 정함

1. free space O → 그곳에 삽입
2. free space X → overflow block에 record 삽입

⇒ pointer chain이 update 되어야 함 → 만약 두 경우가 다 아니라면 restore

③ multitable clustering file organization

department

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Comp. Sci.	Taylor	100000
Physics	Watson	70000

instructor

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

multitable clustering
of *department* and
instructor

Comp. Sci.	Taylor	100000	
10101	Srinivasan	Comp. Sci.	65000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
Physics	Watson	70000	
33456	Gold	Physics	87000

- 여러 relation을 한 file에 저장할 때 사용
- variable size record 사용
- 특정 relation의 record를 연결하기 위해 pointer chain 추가 가능

③ B+tree file organization

④ Hashing

} → 다음 chapter

▼ Colum-oriented storage

(= columnar representation)

- relation의 각 attribute를 독립적으로 저장

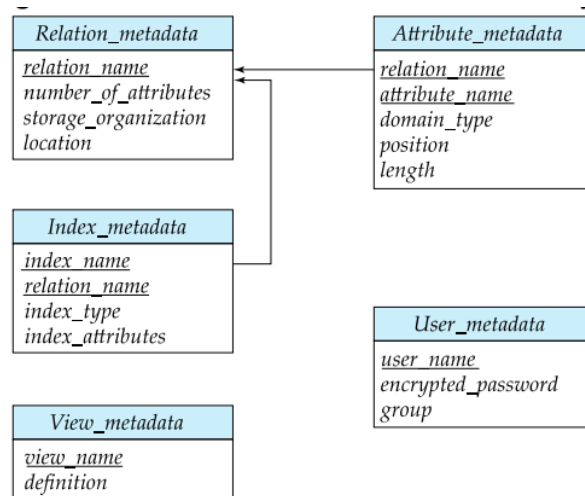
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

▼ Data directory

(= system catalog)

data directory → stores metadata

- metadata



- information about relations
- password를 포함한 user & accounting information
- statistical and descriptive data
- physical file organization
- index들에 대한 정보