Database Management Systems

Database Storage

I/O model of computation

- what’s wrong with memory? It is volatile.
- Typical storage hierarchy:
    - Main memory for currently used data
    - Disk for the main database
    - Tapes for archiving older version of the data
- DBMS assumes the Dominance of I/O cost: The time taken to perform a disk access is much larger than the time likely to be used manipulating that data in main memory. Thus, the number of block accesses (Disk I/O’s) is a good approximation to the time needed by the algorithm and should be minimized.
- In analysis, we also assume the worst case scenario: 100 percent miss rate.

Buffer Management in DBMS

- Similar to paging/buffer management in OS

- Why not let OS manage the buffer?

    - portability issues,

    - extra requirements from DBMS, e.g., to maintain the write ahead log, we may need to force write some pages and be aware which transaction is accessing which page.

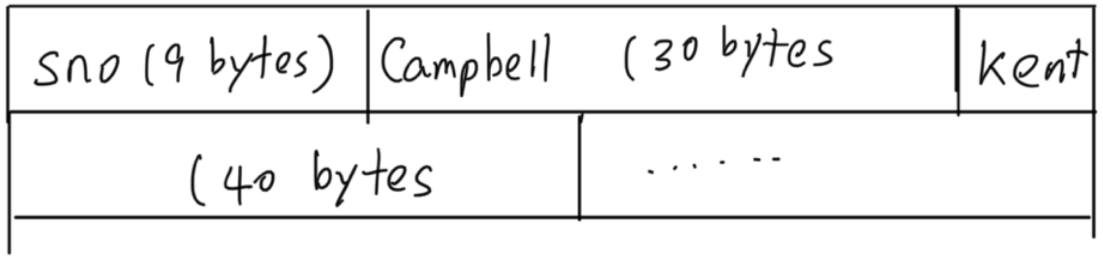
    - we may want to adjust the replacement policy and pre-fetch pages based on access patterns in typical DB operations.

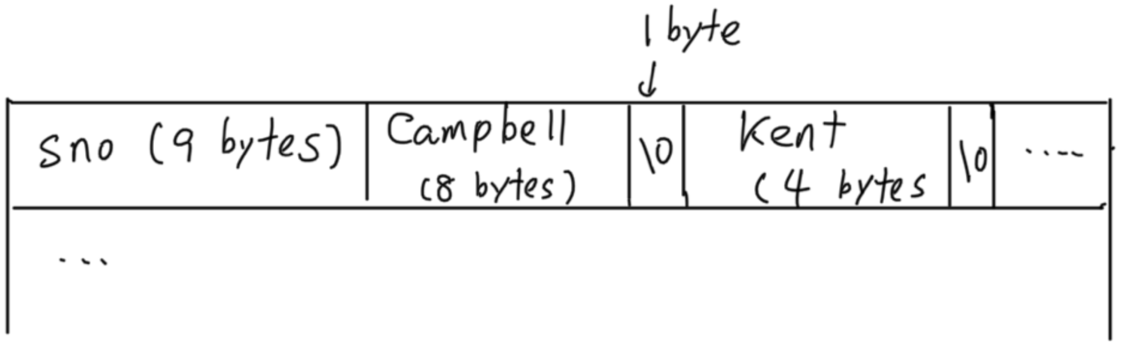
Data on Disk

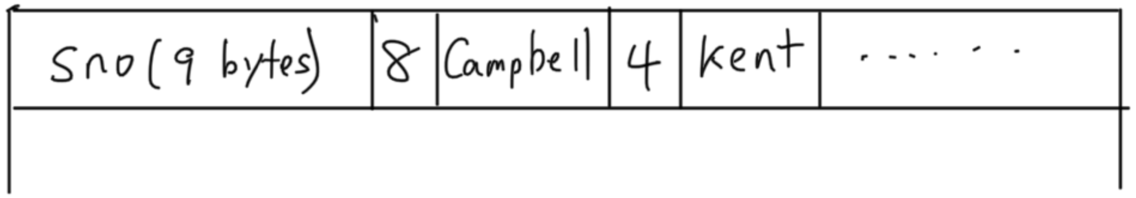
- Record (tuple) format:
    - Fixed Length
    - Variable Length
        - sentinel end-of-record character
        - length/pointer at the beginning of the record
- Page (4k bytes) format
    - as an array of records
    - as an array of pointers
- Table/Relation — Files of Data Pages

Example

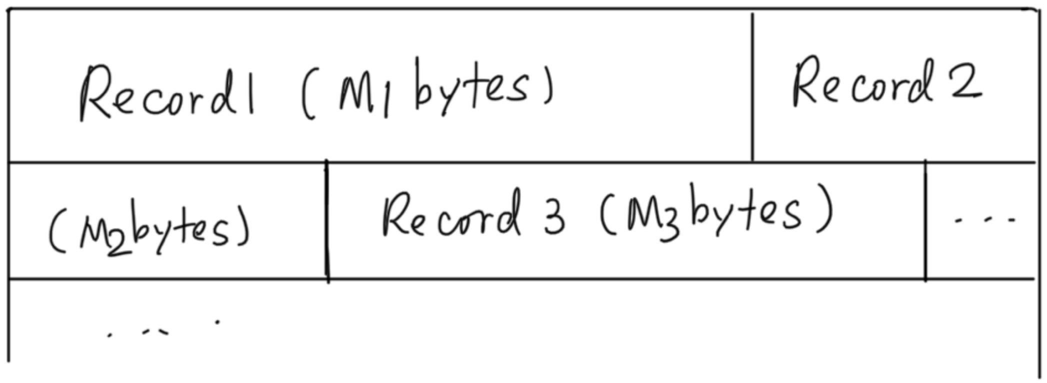
```
Students (
    sno char(9),
    lastName varchar(30),
    firstName varchar(40),
    SIN char(9),
    email varchar(40),
    phone char(10),
    address varchar(80),
    DOB Date
);
```

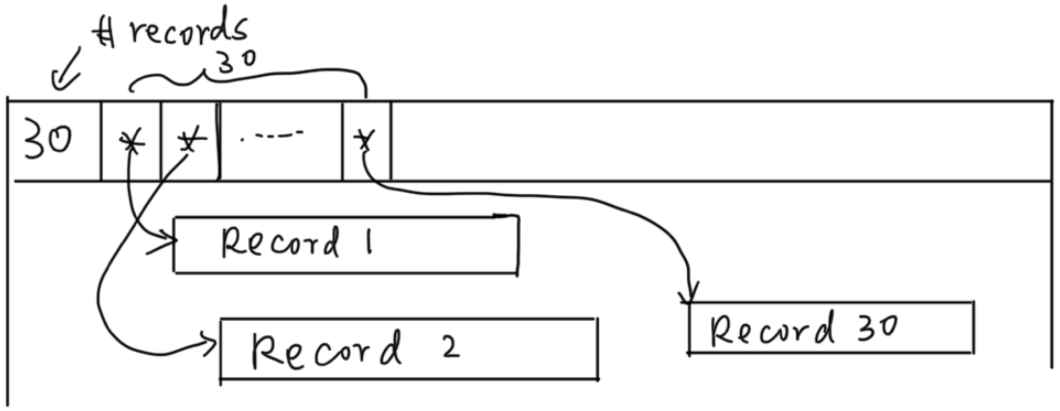


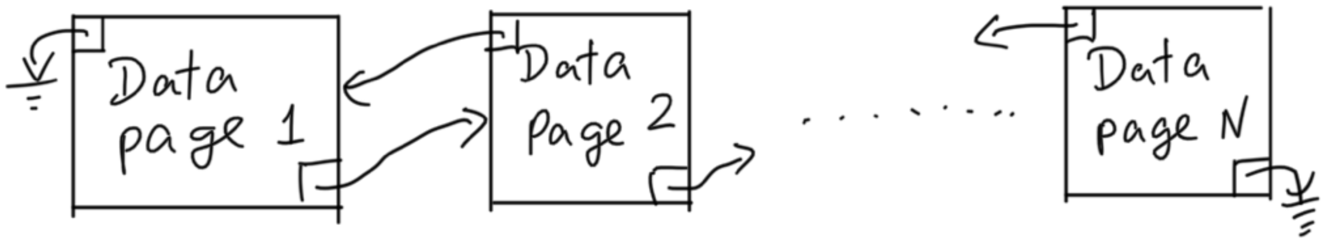




Example


Record1 (M1 bytes)
Record 2
(M2bytes)
Record 3 (M3bytes)
. . .
. .. .


# records
30
30
*
*
.....
*
Record 1
Record 2
Record 30


Data page 1
Data Page 2
. . . .
Data page N

How to access data?

- Table scan

- Using index

- Example:
  10,000 students
  each student record occupies about 250 bytes
  each data page is 4k bytes
  each data page can store about 16 records
  640 data pages to store the 10,000 student records

Why do we need indices?

- For a SQL query:
  select *
  from Students
  where sno = ’123456789’;

- table scan I/O cost: N blocks (640 pages)

- binary search I/O cost: $\log_2 N = \log_2 640$, about 9 to 10 (however, need about 640 nodes to construct the binary search tree)

Index Types

- B+/B tree: widely used, fully dynamic, and support range queries
- Hash Tables
- R-Trees, KD Trees
- ISAM (old, static)
- etc