数据组织与存储

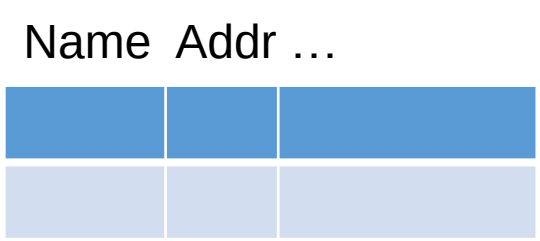
徐辰
华东师范大学
数据科学与工程学院
cxu@dase.ecnu.edu.cn

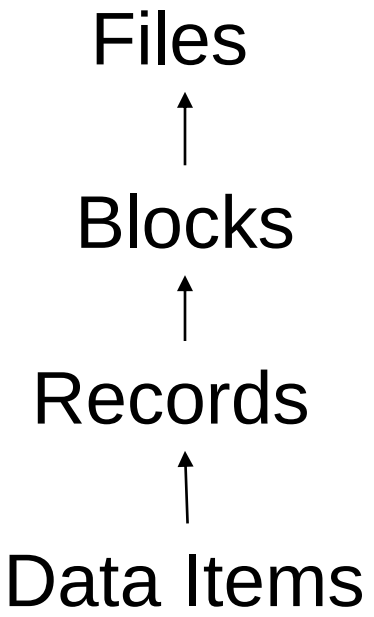
Outline

- How to store (data) record on disk
- How the DBMS locate records
- How to store record with variable size
- How to modify record

Overview

Structured data has a structure in the storage format





Each row is a record

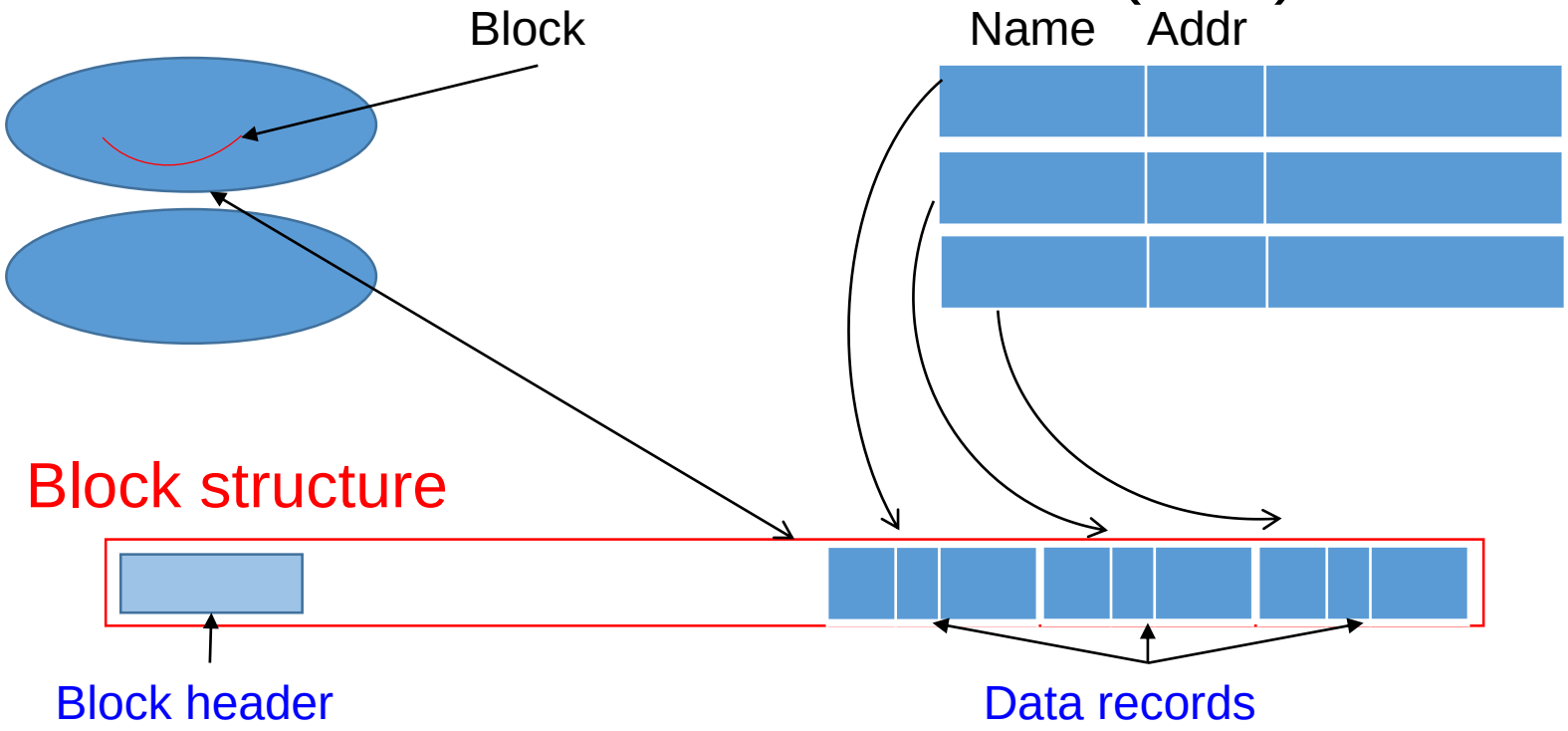
Records ars **stored** in a file

A file consists of one/more disk blocks

How do we store records in disk blocks?

Block Structure

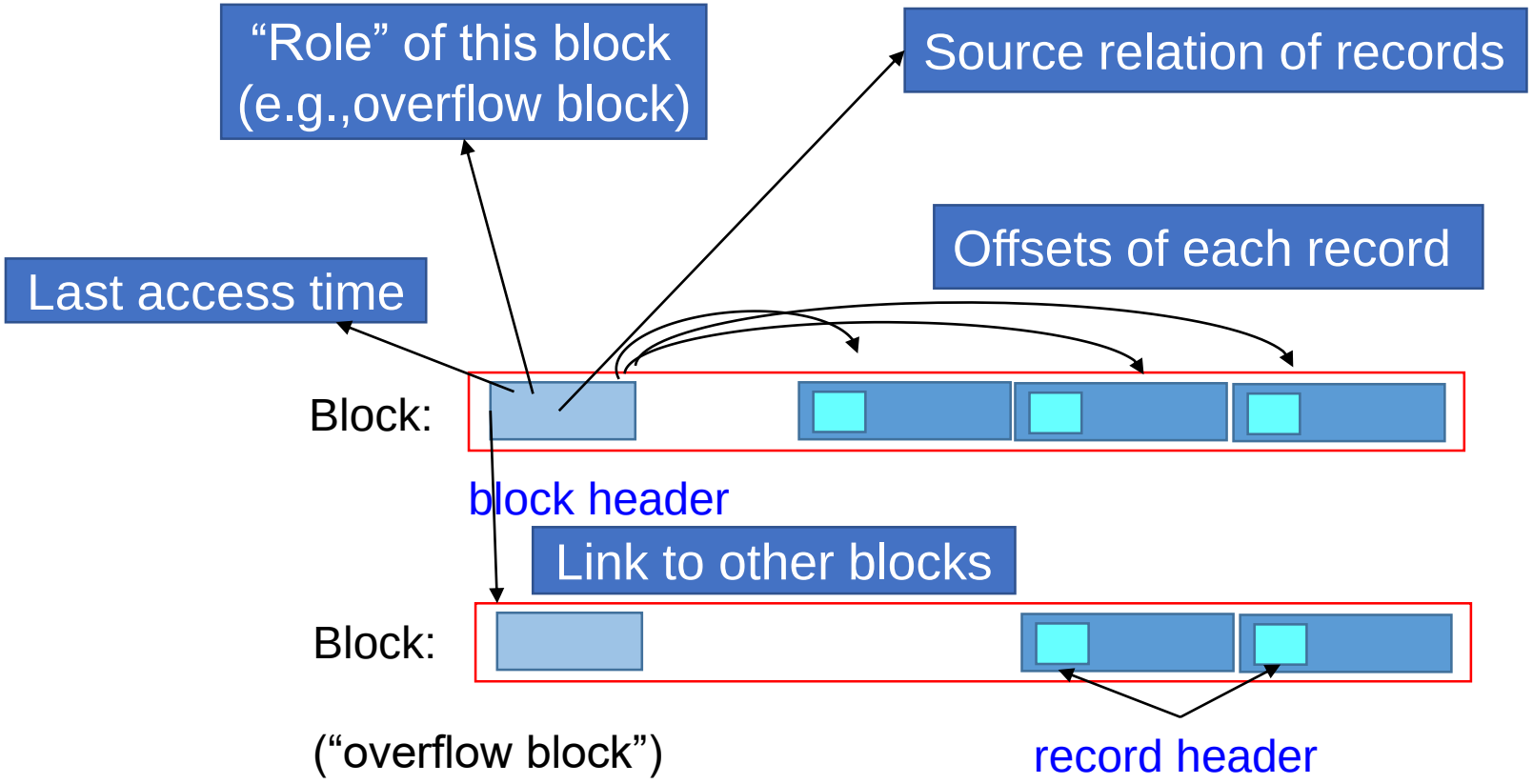
There is a block structure (organization) in a disk block when records are stored in a (disk) block



Block structure = block header + records

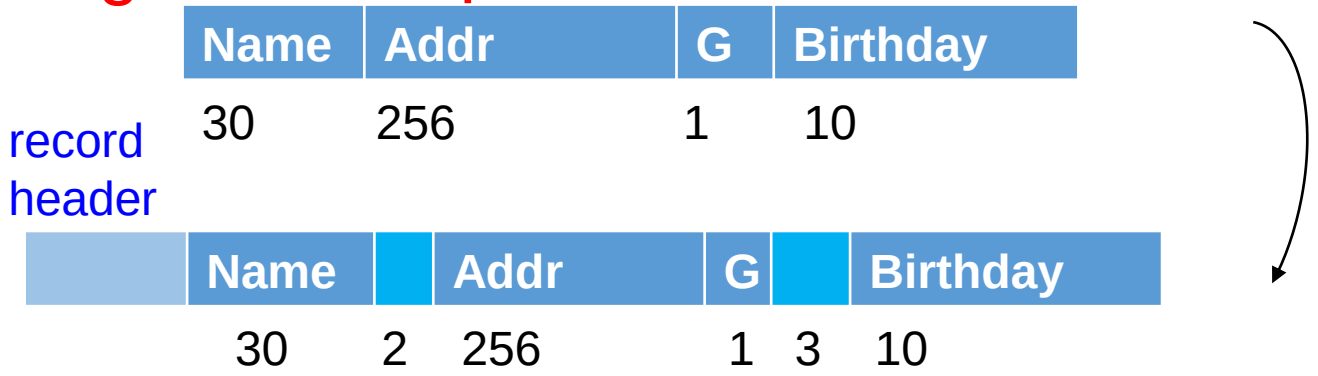
Block Header

Contain info on the records stored in this block



Record Format

- A record consist of
  - Record header
    - Pointer to record scheme
    - Length of record
    - Timestamp that the record was last updated
  - Record fields
- A record will contain "**padding**" bytes to satify alignment requirement



Outline

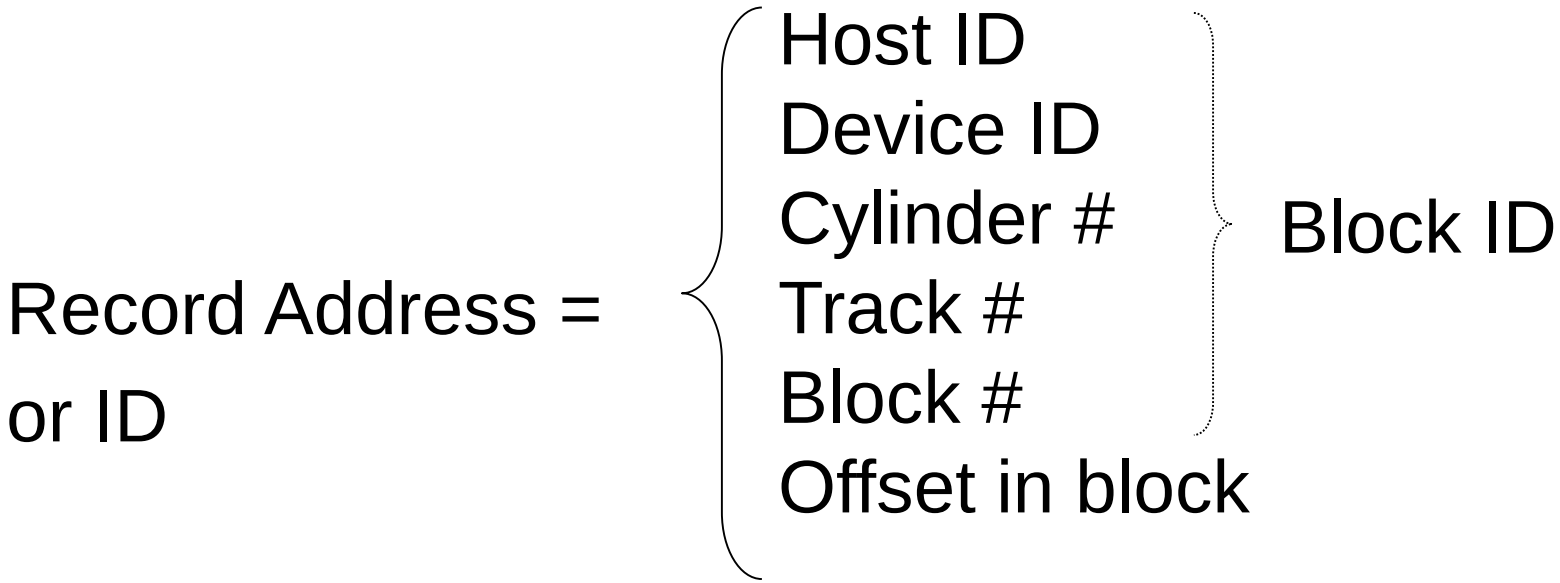
- How to store (data) record on disk
- How the DBMS locate records
- How to store record with variable size
- How to modify record

Blocks/records Address

- Two types of address
  - Database address: data stored **on disk**
    - Physical address
    - Logical address
  - (Virtual) memory address: data stored **in memory**

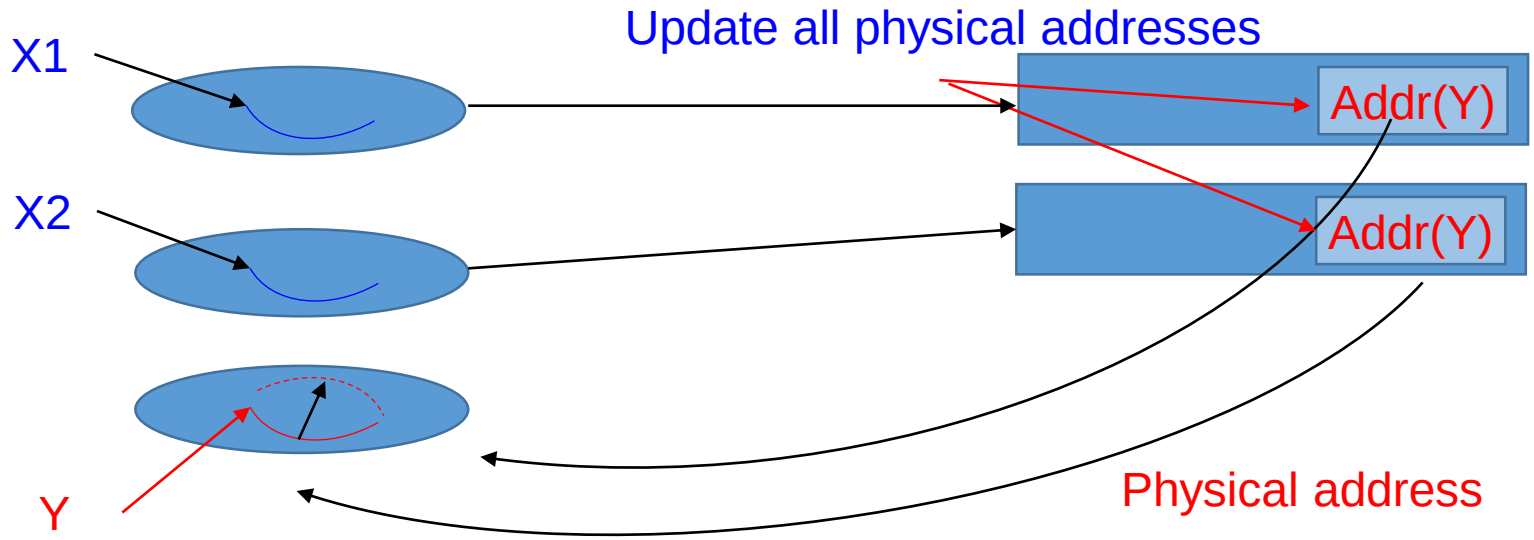
Physical Address

- Direct addressing format
- Byte strings determine the place of block or record in the secondary storage.



Problem of Physical Address

2 records X1 and X2 are referencing the record Y using physical database address.

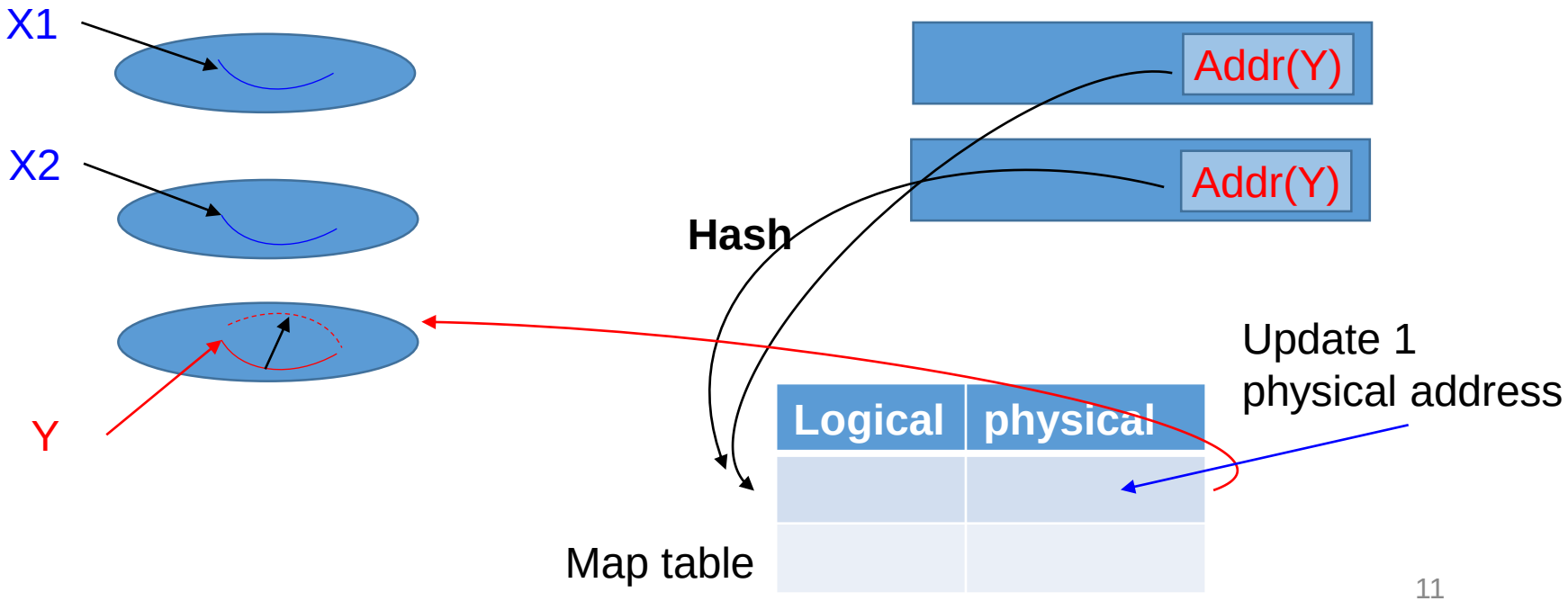


Problem: Move Y to different block on the disk, we must updata multiple physical addresses.

Logical Address

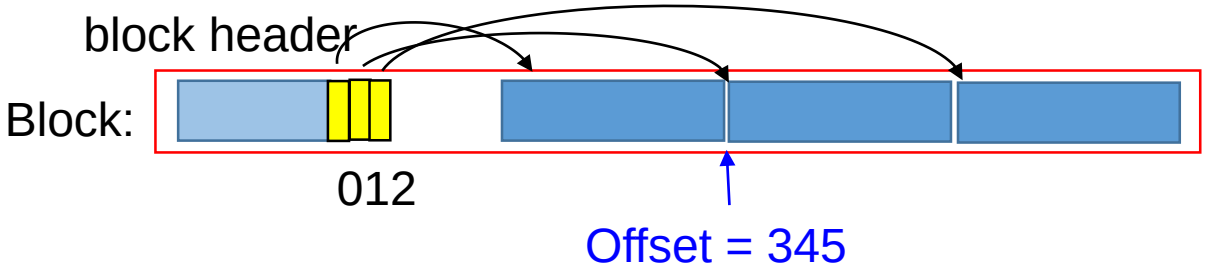
- Indirect addressing format: map table (on disk)
- Record ID is arbitrary bit string

Only update physical address in the map table



Structured Address

- Combination
  - Physical address of block + offset of record
  - 2 way represent offset
    - Direct: (actual) offset in the block (e.g., offset = 345)
    - Indirect: index in the offset table (e.g., offset = 1)

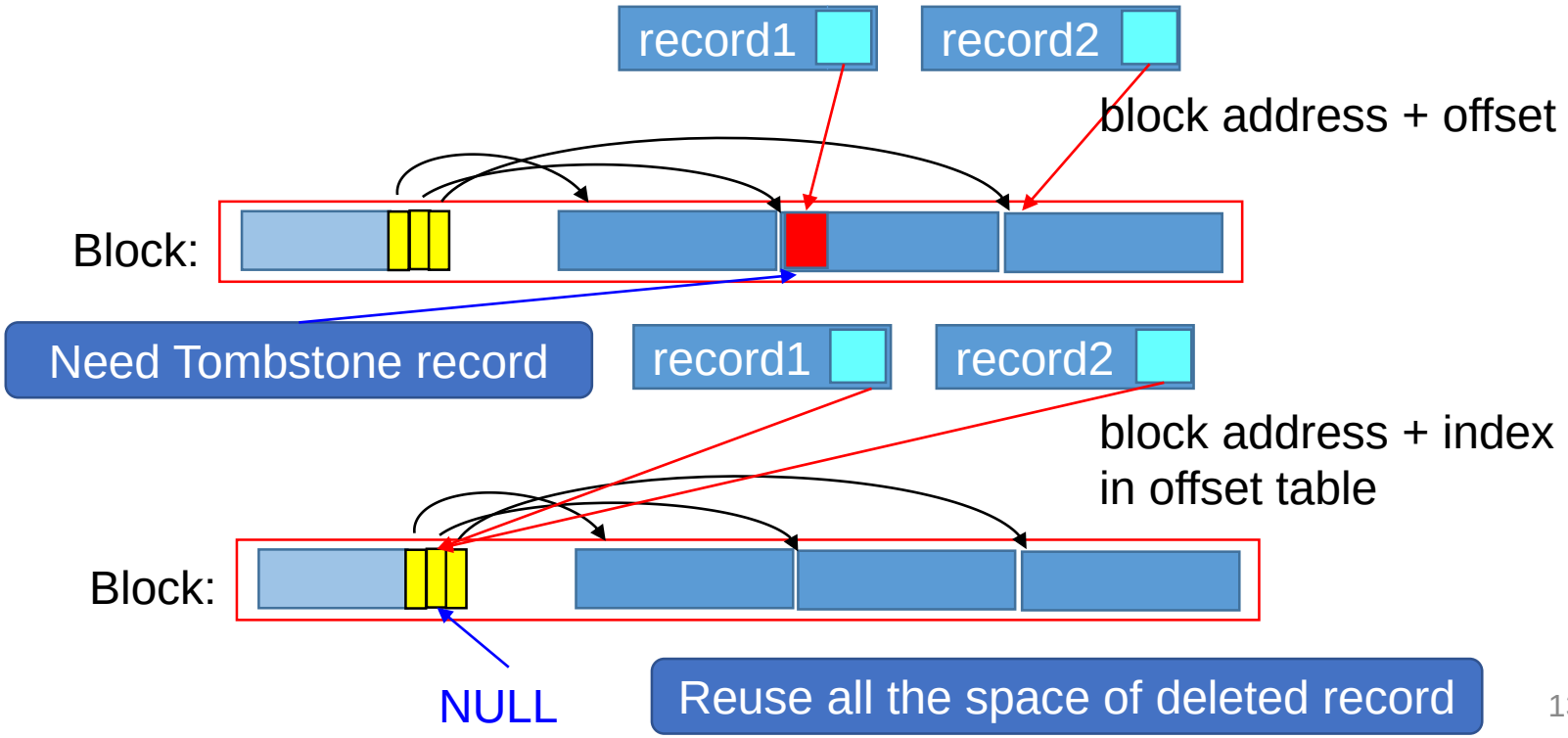


- Addvantage
  - No need a global of map table

Direct Offset vs Indirect Offset

Consider the deletion of a record

- Direct offset: need a tombstone record
- Indirect offset: set offset to NULL



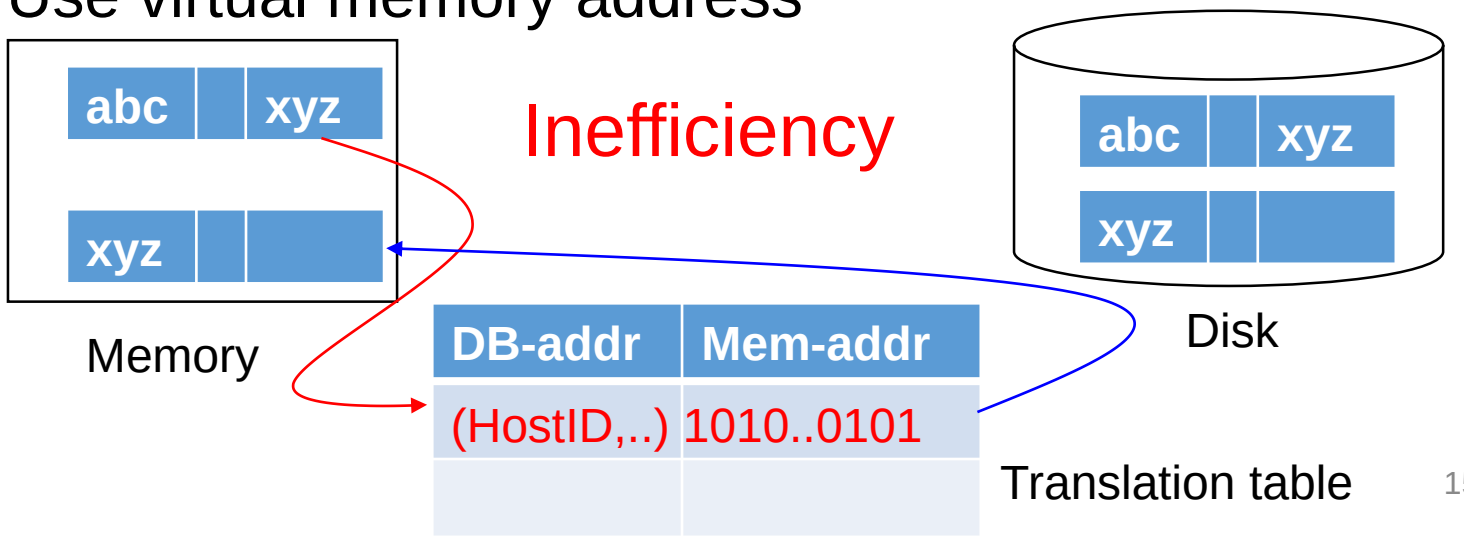
Blocks/records in Memory

- Block/record is read into main memory
  - **Must** use virtual memory address
- Problem
  - Where is the block right now?
  - How to find the memory address of a block?
- Solution for both problems
  - Use a translation table
    - Map : Database address - >Virtual memory address

How to Use Translation Table

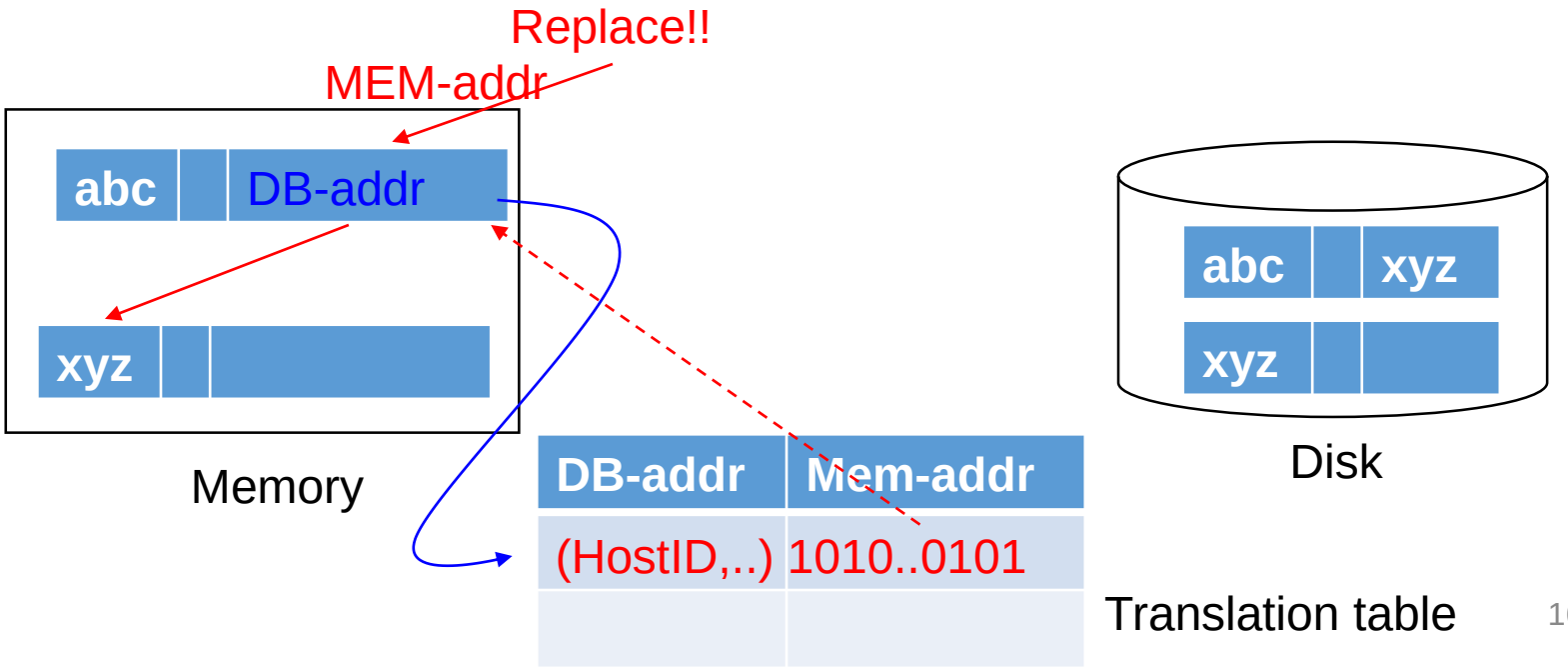
Hash database address x

- x is **not found** in translation table
  - Read the database object into memory (update the translation table)
- x is **found** in translation table
  - Use virtual memory address



Pointer Swizzling

Replace database address in a record by virtual memory address when the referenced data block/record resides in memory.

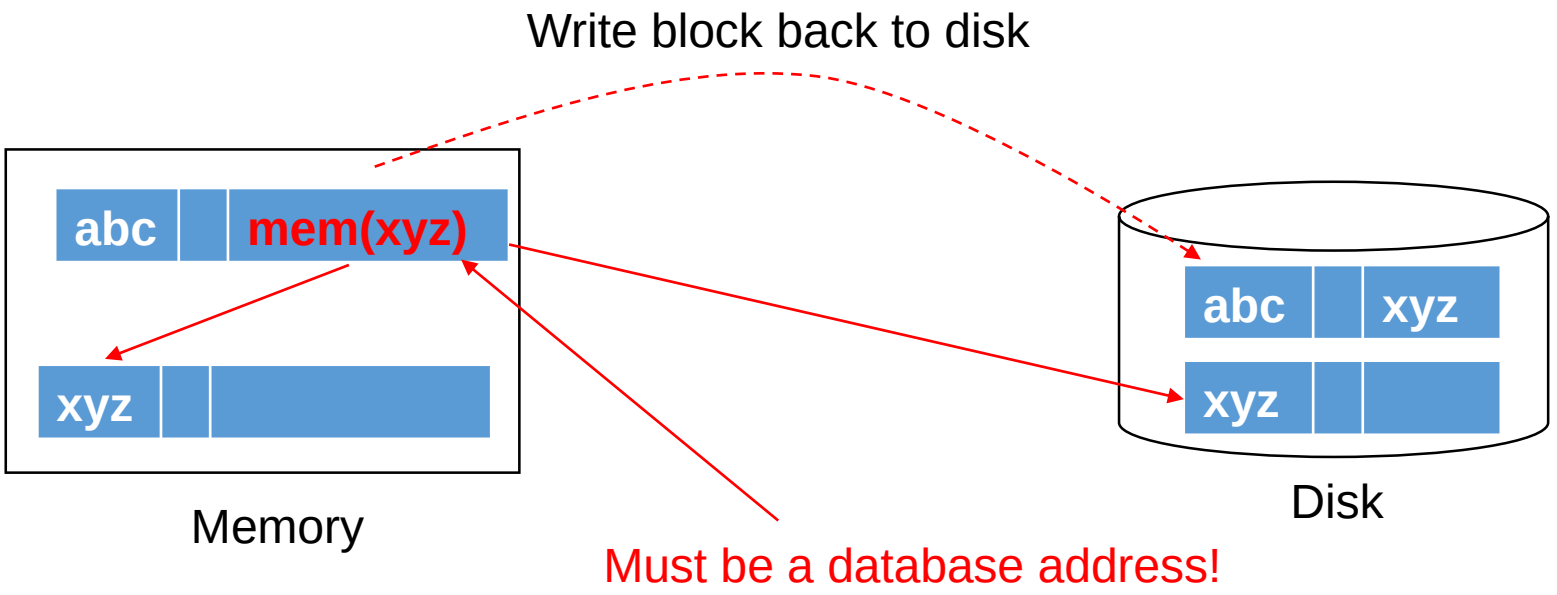


Swizzling Policies

- Automatic swizzling
  - Replace all database address in the records stored in the block when a block is read into memory.
- On-demand swizzling
  - Swizzle the database address when we access a record that has not been swizzled.
- No swizzling
  - Never to swizzle pointers
- Programmer control

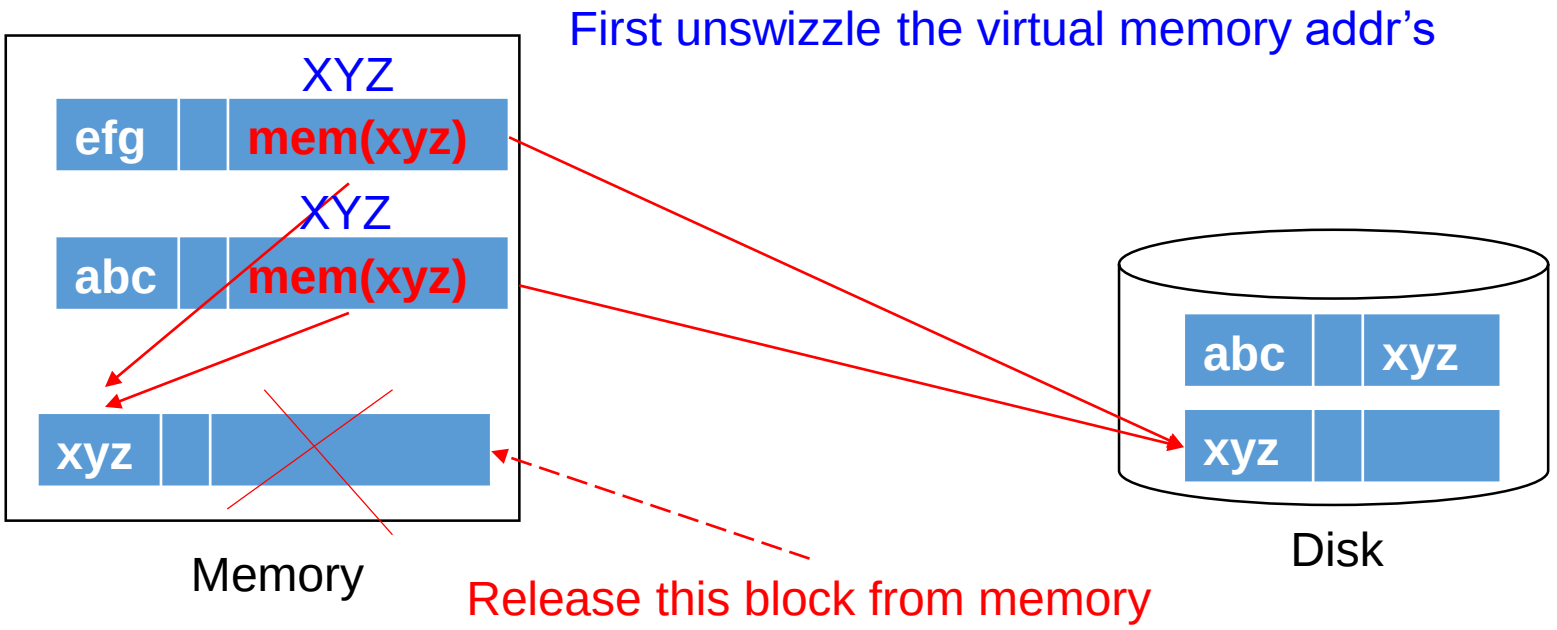
Problems caused by Pointer Swizzling

- Problem 1
  - Unswizzle all virtual addresses before we write a block back to disk.



Problems caused by Pointer Swizzling

- Problem 2
  - Release memory used by a disk block can result in invalid record references in other records.



Outline

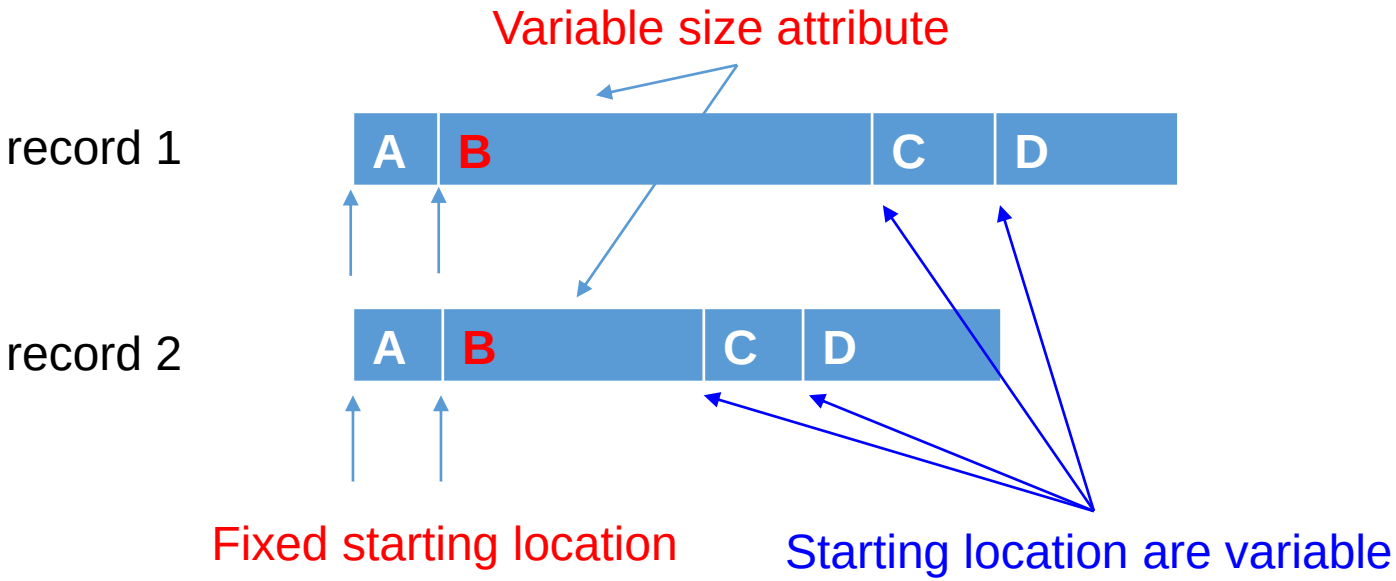
- How to store (data) record on disk
- How the DBMS locate records
- How to store record with variable size
- How to modify record

Need for Variable-Length Record

- Variable size record
  - String typed attribute
  - Repeating field
- Variable format
  - XML element
- Large size field

Problem of Storing Variable Size Record

Must store the starting location of every field in the record

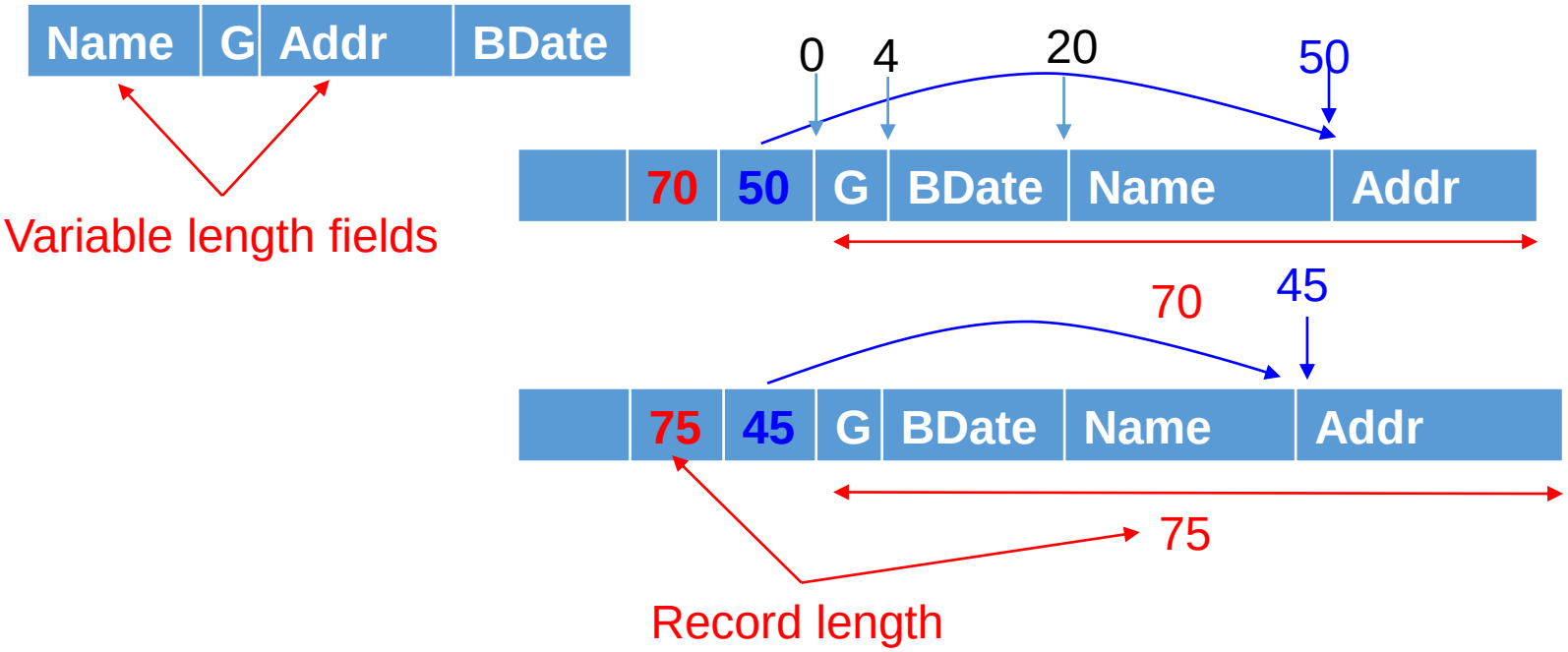


How to store records with variable length field that uses minimal number of starting location information ?

Solution to Store Variable Size

Solution 1:

Store all fixed length fields first, then store the variable size attributes

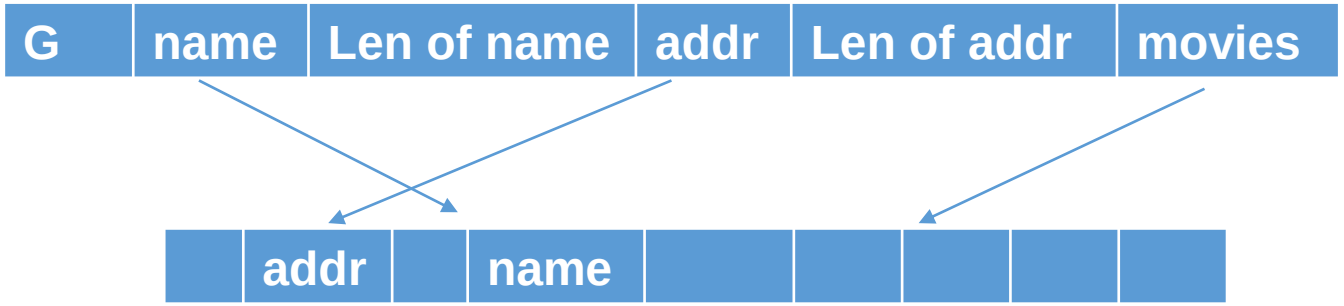


Solution to Store Variable Size

Solution 2:

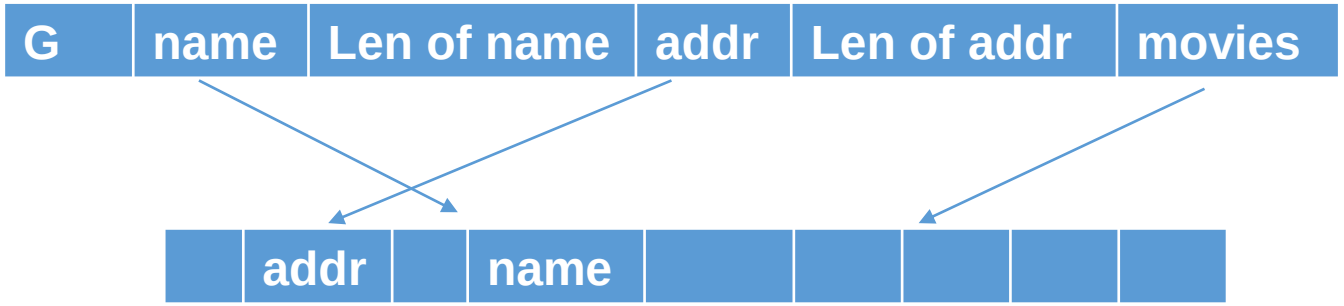
Use 2 separate block

- A fixed length record with data + pointer to variable length fields
- The variable length fields are stored separately



Advantages and Disadvantages of Using 2 Separate Block

- Advantage
  - Search more efficiently
  - Minimize the overhead in block header
  - Easy to move record within or among blocks
- Disadvantage
  - Increase the disk IO (need 2 disk access)



Storing Variable Format Record

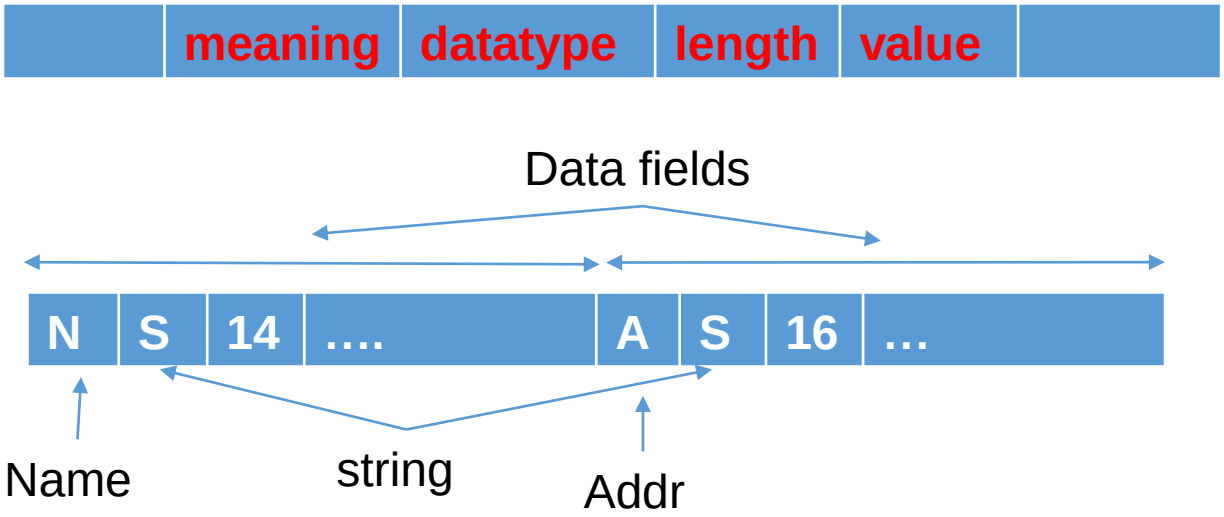
Problem:

- Record structure is unknown or variable

Solution:

- First define a encoding scheme

Example

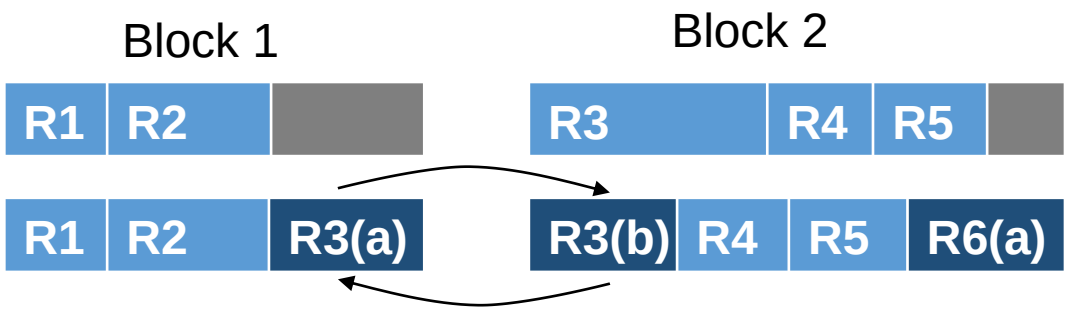


Spanned and Unspanned Record

- Problem
  - Packing whole record into blocks wastes space
  - Example
    - If record are just slightly larger than half a block, it waste space can approach 50%
- Solution
  - Spanned record
- Example

Unspanned

Spanned

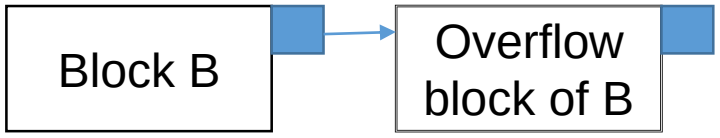


Outline

- How to store (data) record on disk
- How the DBMS locate records
- How to store record with variable size
- How to modify record

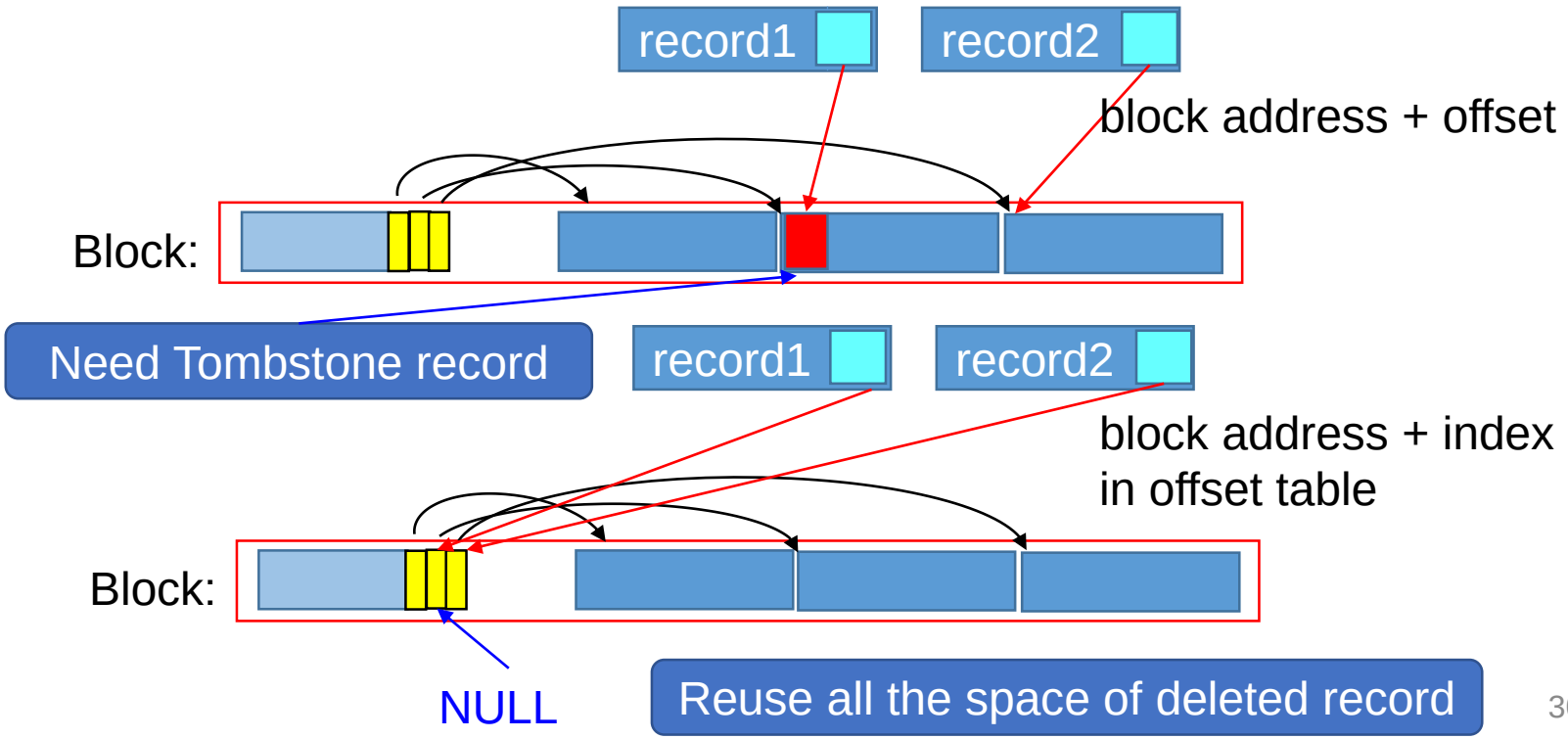
Insertion

- Easy case: records not ordered
  - Insert new record at end of file or in a deleted slot
- Hard case: records are ordered
  - Has room within the block
    - Slide records and insert
  - No room within the block
    - Find room outside the block
      - Nearby block
      - Overflow block



Deletion

- Need a tombstone record
- Set offset to NULL



Thank you!
Q & A