Identifying blocks/records on disks and in memory

- **Identifying a block/record on *disk*: Block *Address* and record *address***
  - **Fact:**

    - A **block** is **identified** by a **block address**

    - A **record** is **identified** by a **record address**

    (A **record address** is an **extension** of a **block address**, so in this discussion, I will often use: **block/record address** and handle **both topics** at the same time)

- **Identifying a block/record in memory**
  - **Fact:**

    - A **block/record** can be **read** into the **computer memory**

  - **Identifying** a **block/record** when it is **in memory**:

    - Use a ***virtual*** **memory address** (**paging**)

- **Overview: Types of addresses to identify blocks/records**
  - There are **2 types** of **address** to **identify** a **block/record** in **use**:

    1. **Database Address**: used to identifies **data (block or record)** stored on **disk**

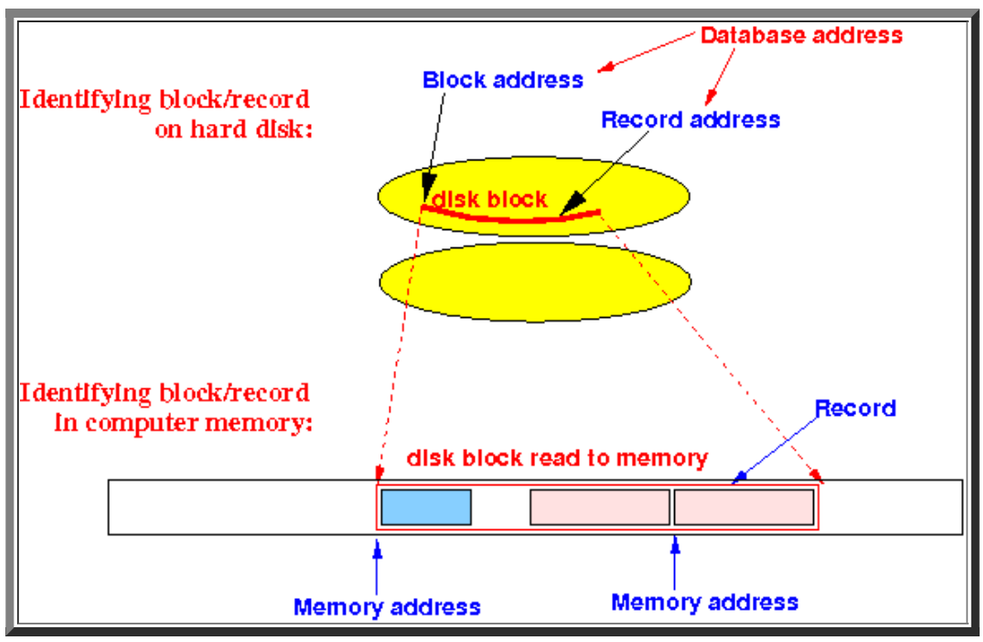
       There are **2 kinds** of **database addresses**:

       - ***Physical*** **address**
       - ***Logical*** **address**

       ---

    2. **Virtual memory address**: used to identify **data (block or record)** stored in **(virtual) memory**

    Graphically:



- **Database address = address used to identify a block (data) on *disk***
  - **Database Address:**

    - **Database Address** = an **address** used to identify **data (block/record)** stored on **disk**

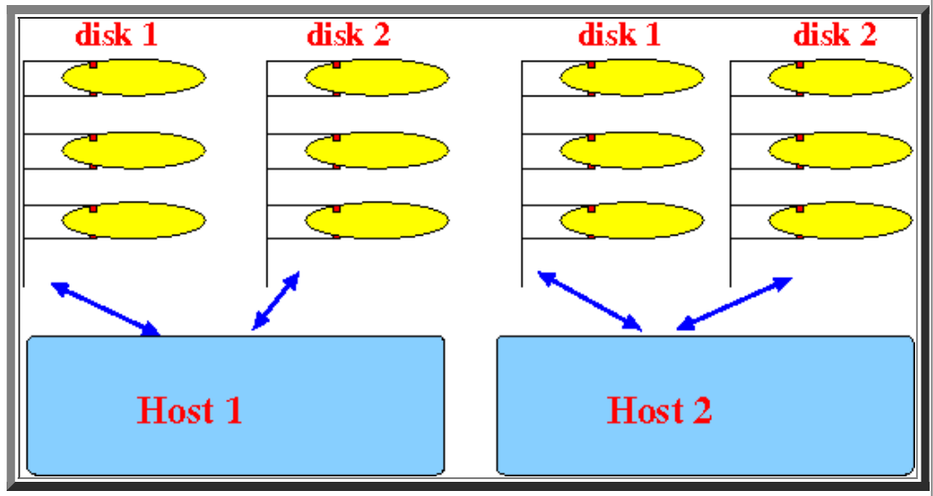
- The are **2 common techniques** to **identify** a **block/record** stored on **disk**:

  - ***Physical* address**
  - ***Logical* address**

- ***Physical* Address: a *direct* addressing format for identify block/record on a disk**
  - **Physical Address** of a **block**:

    - **Physical Address** of a **block** = **address** used to identify **block** on a **disk**

    - **Components** of a **block physical address**:



      - **Host ID** that contains the **disk**
      - **Disk ID** the **identifies** the disk on the **host**
      - **Block (cylinder/track/section) number** on the **disk**

  - **Physical Address** of a **record**:

    - **Physical Address** of a **record** = **address** used to identify **record** on a **disk**

    - **Components** of a **record physical address**:

      - **Physical address** of a **block**
      - **Offset** of the **record** in the **block**

- ***Logical* Address: an *indirect* addressing format for identify block/record on a disk**
  - **Logical block address:**

    - Each **block/address** is assigned a unique ***logical* address**

    - **Logical address** = an **arbitrary string** of **fixed length bits**

      (Can be generated **automatically** using some sequence generator or keep adding 1 to a counter)

  - **DBMS** uses a **map table** to **translate**:

    - **Map table**:

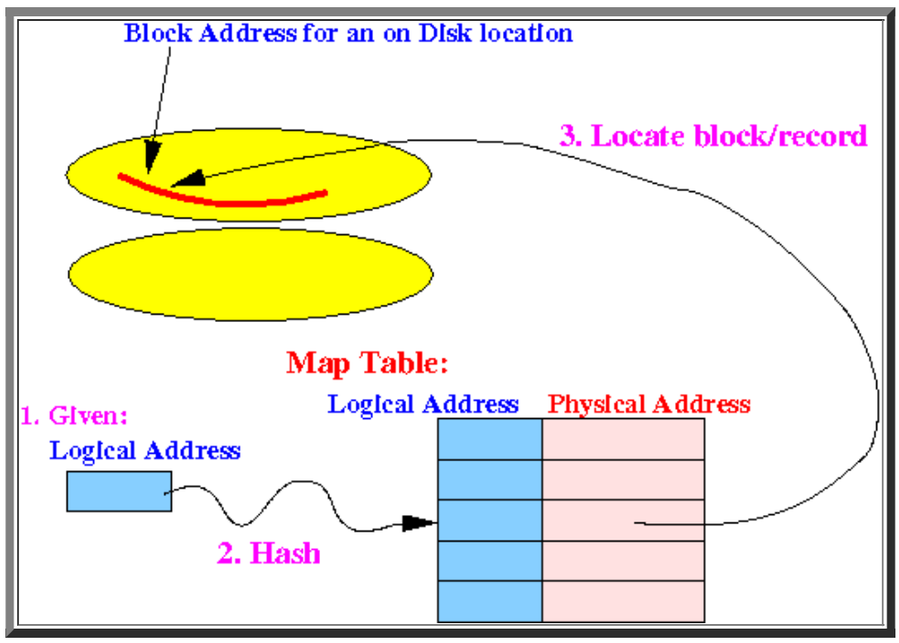
```
    Logical Address     Physical Address
-------------------+-----------------------------------
  101001.......10  |  (HostID, DiskID, ..., BlockID)
  010101.......01  |  (HostID, DiskID, ..., BlockID)
  ...
```

    - To **speed up** **access**, the **Map Table** is organized as:

- a **hash table**.

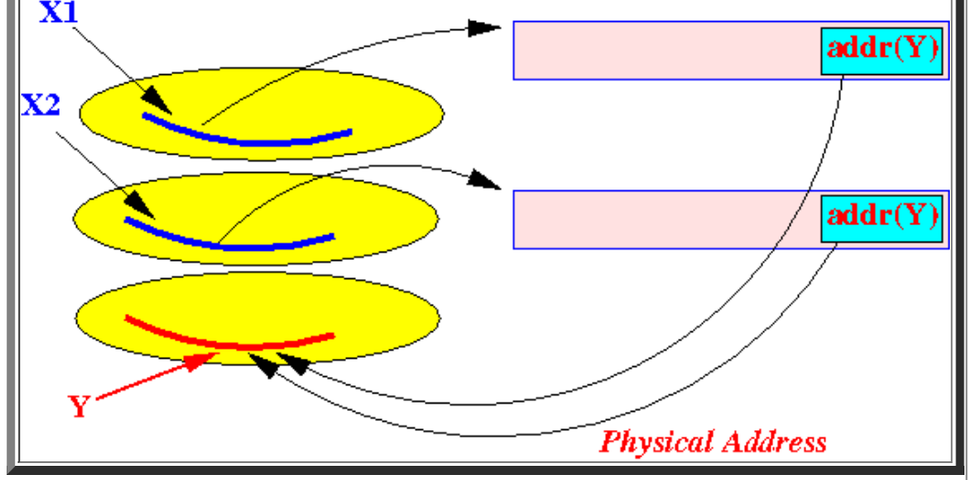
- The **map table** is **stored** on **disk** (in a **well-known location**)

- **How** to use the **map table**:



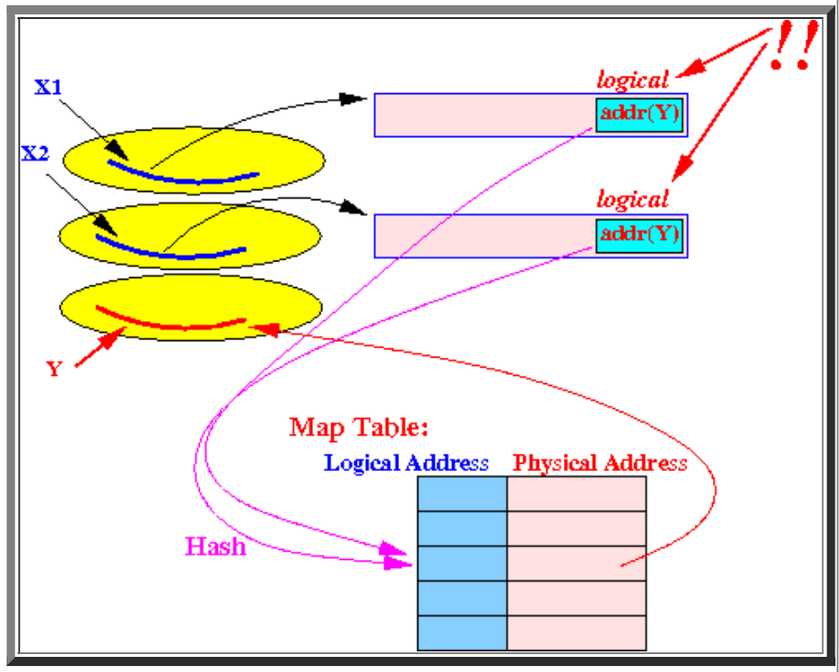
- ***Usage* of (physical/logical) database addresses**
  - **How** a **(physical/logical) database address** is **used**:
    - When some **record *x*** need to **refer** to ***another* record *y***, we can **store**:
      - the **(physical/logical) database address** of ***y*** inside ***x***
  - **Intro: Techniques** to **referrence (point)** to a **record**:
    - **Primary key/foreign key**:
      - Each **record x** contains a ***unique* primary key**
      - **Another record y** that **references** the **record x** will store the **primary key** of **x**

        (The **primary key** of **x** stored in **record y** is called a ***foreign* key**)
    - The ***direct* approach**:
      - The **record y** stores the ***physical/logical* address** of the **record x**
  - **Problem** with **referencing** another **record** using a ***physical* address**:
    - **Example:** 2 records references the **record Y**



- **Problem**: If **record Y** is ***moved*** to a ***different*** **part** of the **disk**:
  - We must **update** ***many*** addresses !!!

- **Solution** using a ***logical*** **address** as **reference**:
  - **Example:** 2 records references the **record Y**



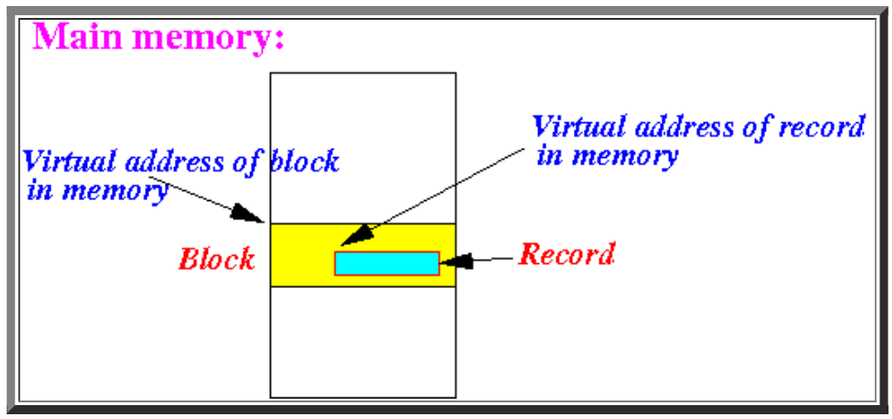
  - If we **move** the **record Y**, we ***only*** need to:
    - **Update** the **physical address** in the **map table** !!!

- **Caveat: block in memory**
  - **Fact:**
    - A **block/record** is **stored** on ***disk*** (this is ***not*** the **problem**)
  - **Caveat/Problem:**
    - A **block/record** must be **stored** in ***main memory*** in order to **perform** **read/write operations** on **data** in the **block**

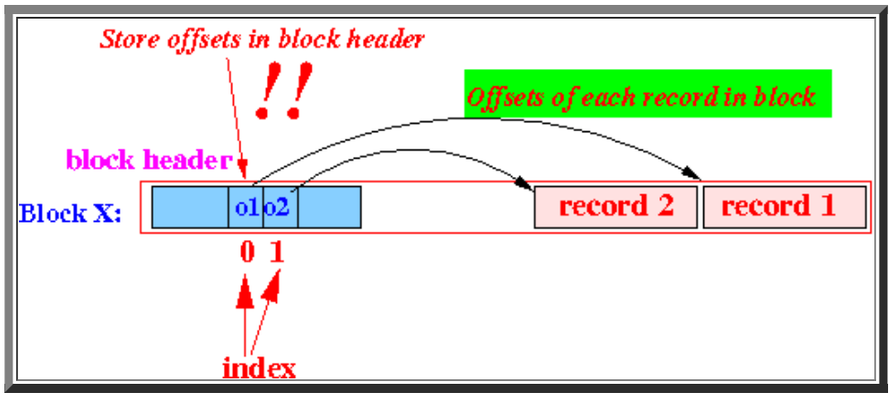
- ***Virtual* Address: address used to identify block/record in main memory**
  - **Main memory:**

    - **Each data item** stored in **(main) memory** is identified by its ***virtual* memory address**

    - **While** a **block/record** is in **memory**, the **DBMS** must use the ***virtual* address** to **identify** the **block/record**:



- ***Common* ways to identify a *record* inside a block**
  - **Consider** the **records** stored in a **block**:



Clearly:

- **Address** of a **record** = **address** of the **block** that contains the **record** + some ***offset*** information

---

  - There are **2 common ways** to **identify** a **record**:

1. (**DatabaseAddress(Block X)** + the ***(absolute) offset* in block**)

   Example:

   - **record 1**: (**DatabaseAddress(Block X)**, **234 (offset in bytes)**)
   - **record 2**: (**DatabaseAddress(Block X)**, **456 (offset in bytes)**)

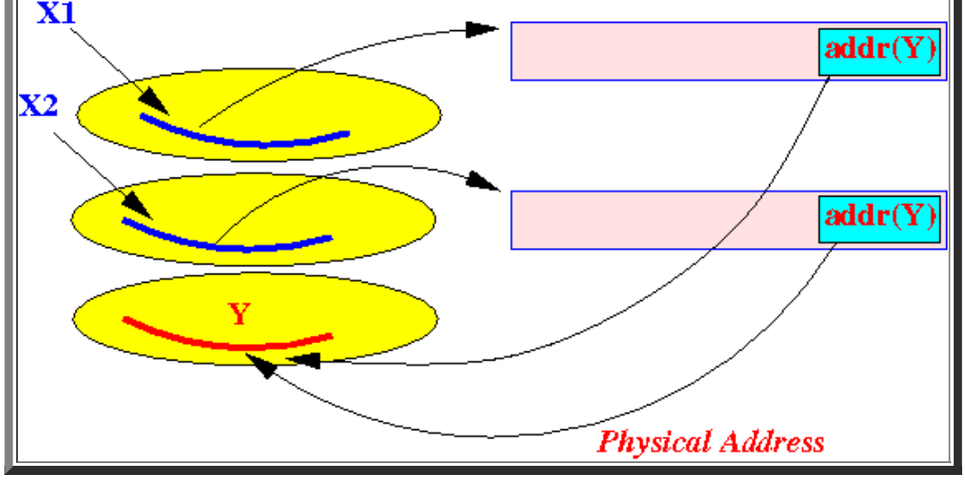
---

2. (**DatabaseAddress(Block X)** + the ***index*** of the **offset table** (in the block header)

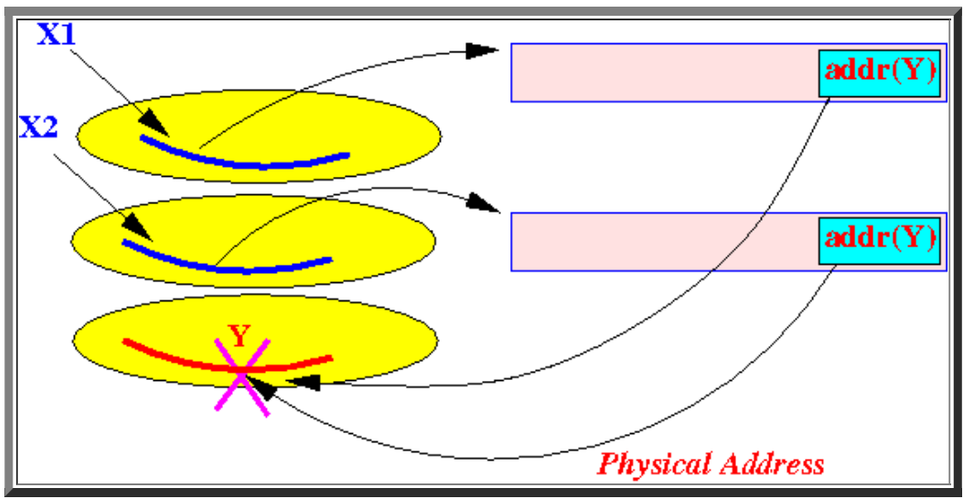
   Example:

   - **record 1**: (**DatabaseAddress(Block X)**, **0**)
   - **record 2**: (**DatabaseAddress(Block X)**, **1**)

- **A caveat when using *physical* addresses to reference a block/record**
  - **Illustrative example:** 2 block with **records** referening a **record Y**:

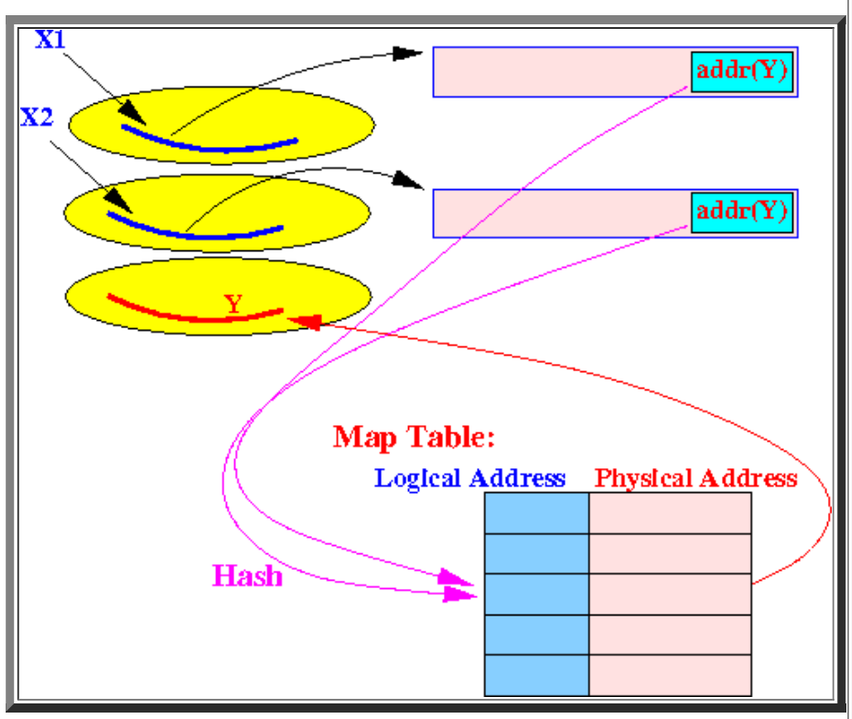


- When the **record Y** is **deleted** :

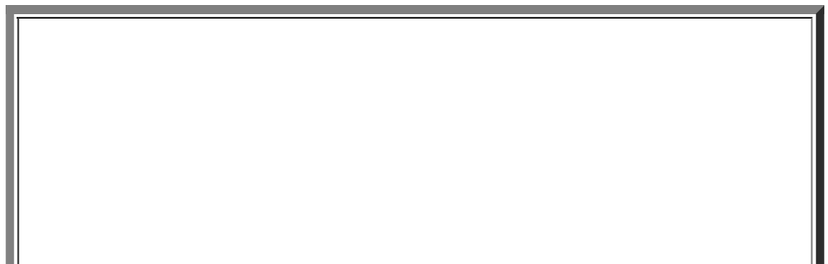


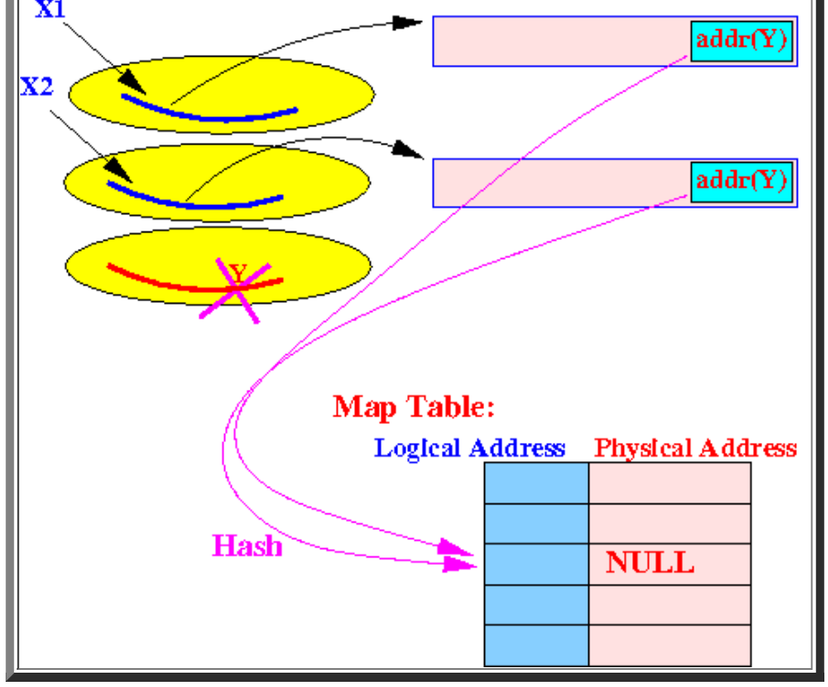
the ***physical* addresses** will reference an **incorrect record** !!!

- **Techniques to handle record *deletion***
  - Using ***logical* addresses** is **easy**:
    - **Before** deleting **record Y**:



After deleting **record Y**:





(The **logical address** used by **record Y** needs to store in the ***map table*** !!!)

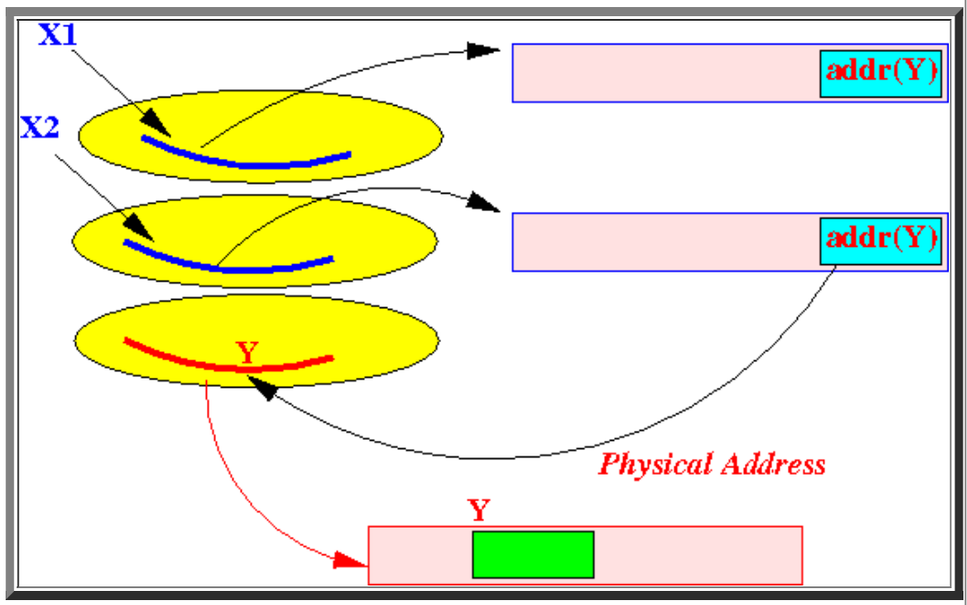
- **Deleting** a **record** using ***physical*** **address**: use a **tomb stone record**

  - **Tomb stone record**: a (very small) ***special purpose*** **record** used to indicate a **deleted record**

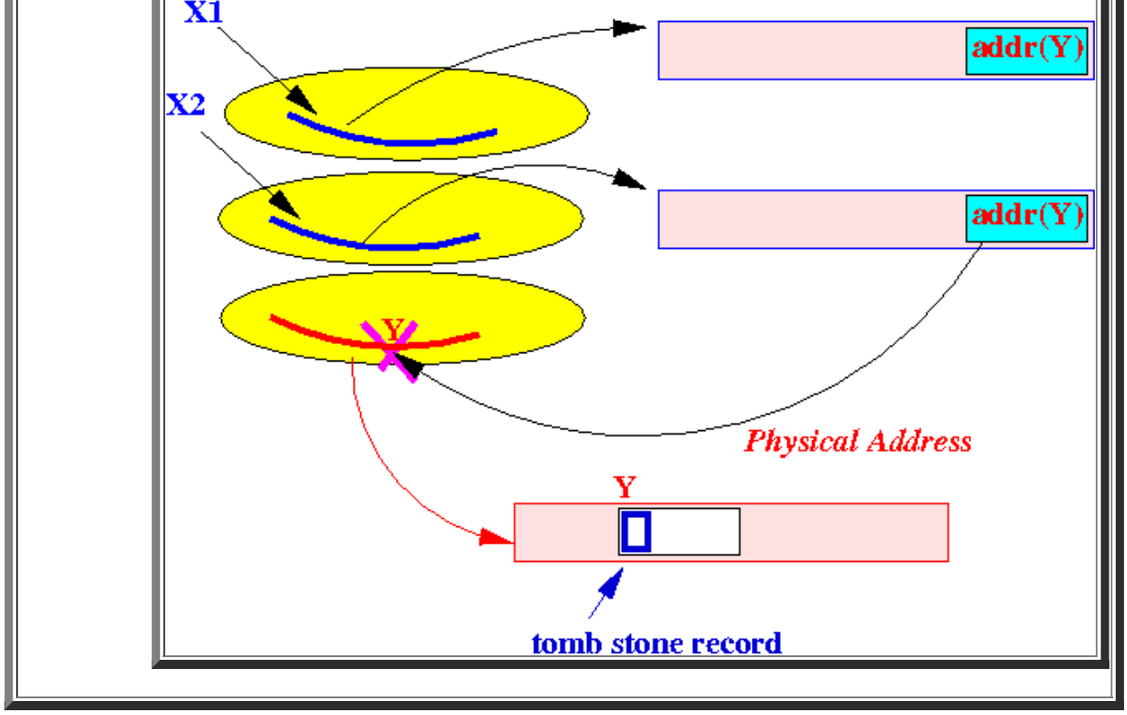
  - When a **record** is **delete**, it is **replaved** by the **tomb stone record**

  Example:

  - **Before** deleting **record Y**:



  - **After** deleting **record Y**:



- **Caveat using a tomb stone record**
  - **Insertion**:

    - When you **insert** a new **record**:

      - You *cannot* use the **space** of a **tomb stone record** !!!

      Because:

      - **Existing record references** to the **deleted record** will *then* references to the *newly* **inserted trecord**: