ICOM 6005 – Database Management Systems Design

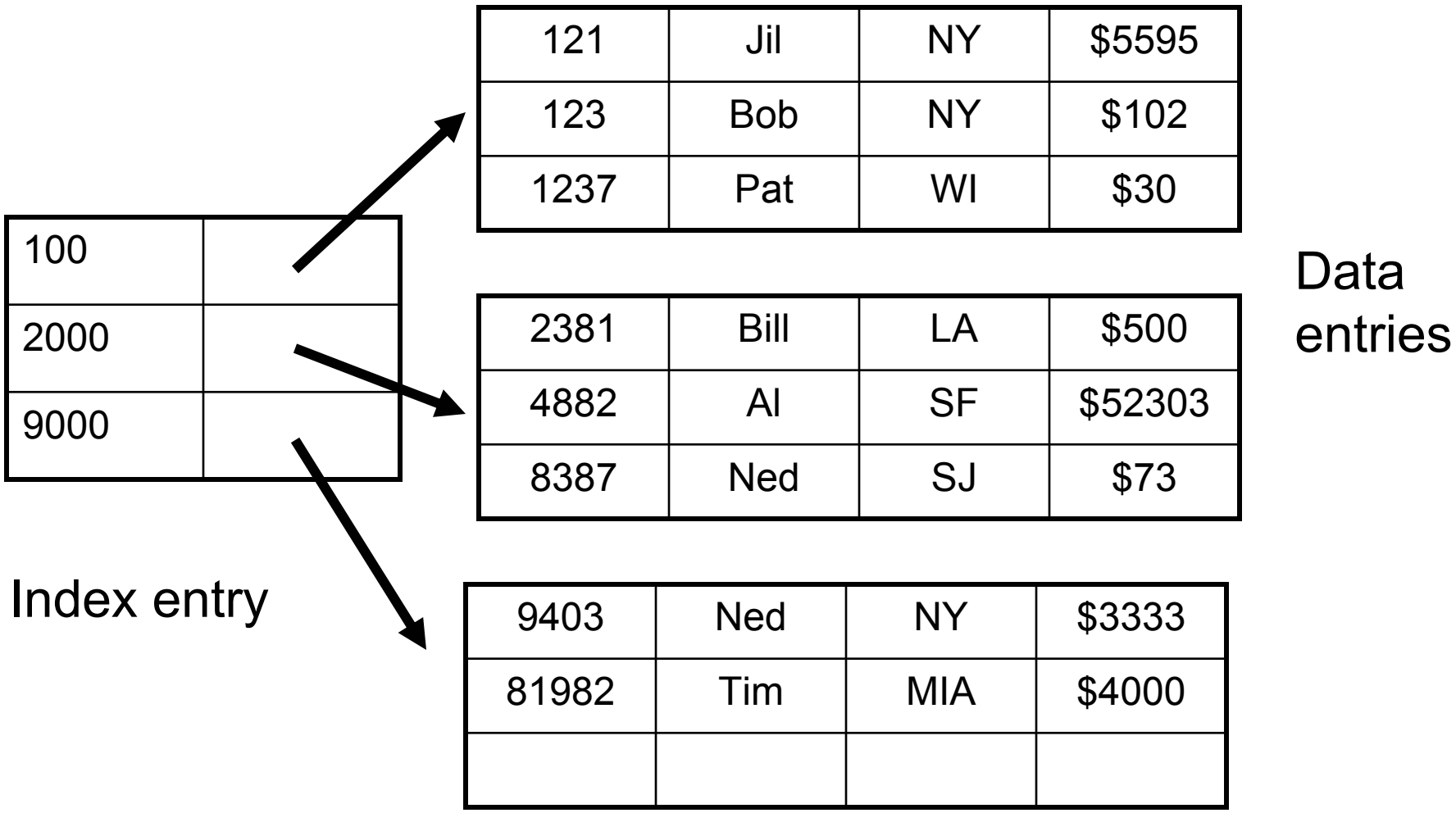
Dr. Manuel Rodríguez-Martínez

Electrical and Computer Engineering Department

Readings

- Read
  - New Book: Chapter 8

Index File


121 Jil NY $5595
123 Bob NY $102
1237 Pat WI $30
2381 Bill LA $500
4882 Al SF $52303
8387 Ned SJ $73
9403 Ned NY $3333
81982 Tim MIA $4000
100
2000
9000
Data entries
Index entry

Index files structure

- Index entries
  - Store search keys
  - Search key – a set of attributes in a tuple can be used to guide a search
    - Ex. Student id
  - Search key do not necessarily have to be candidate keys
    - For example: gpa can be a search key on relation:

      Students(sid, name, login, age, gpa)
- Data entries
  - Store the data records in the index file
  - Data record can have
    - Actual tuples for the table on which index is defined
    - Record identifier for tuples that match a given search key

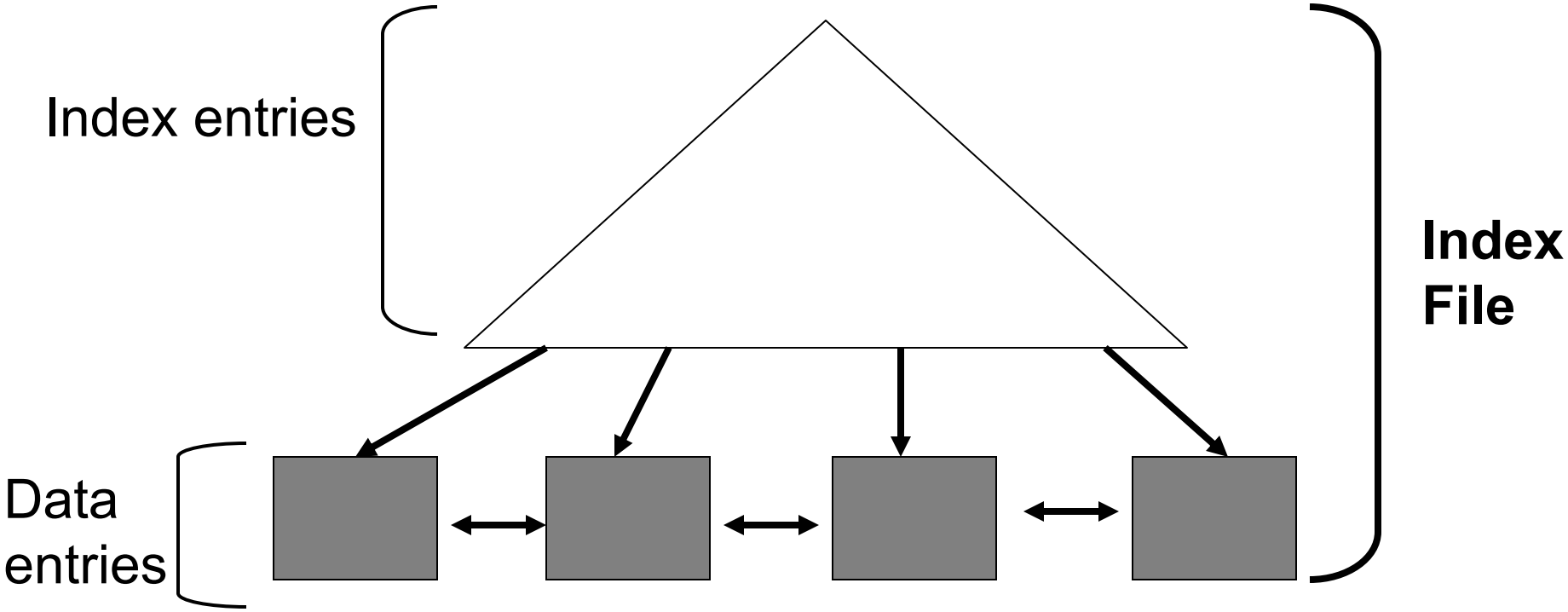
Issues with Index files

- Index files for a relation R can occur in three forms:
  1. Data entries store the actual data for relation R.
     - Index file provides both indexing and storage.
  2. Data entries store pairs <k, rid>:
     - k – value for a search key.
     - rid – rid of record having search key value k.
     - Actual data record is stored somewhere else, perhaps on a heap file or another index file .
  3. Data entries store pairs <k, rid-list>
     - K – value for a search key
     - Rid-list – list of rid for all records having search key value k
     - Actual data record is stored somewhere else, perhaps on a heap file or another index file.

Clustered vs Unclustered Index

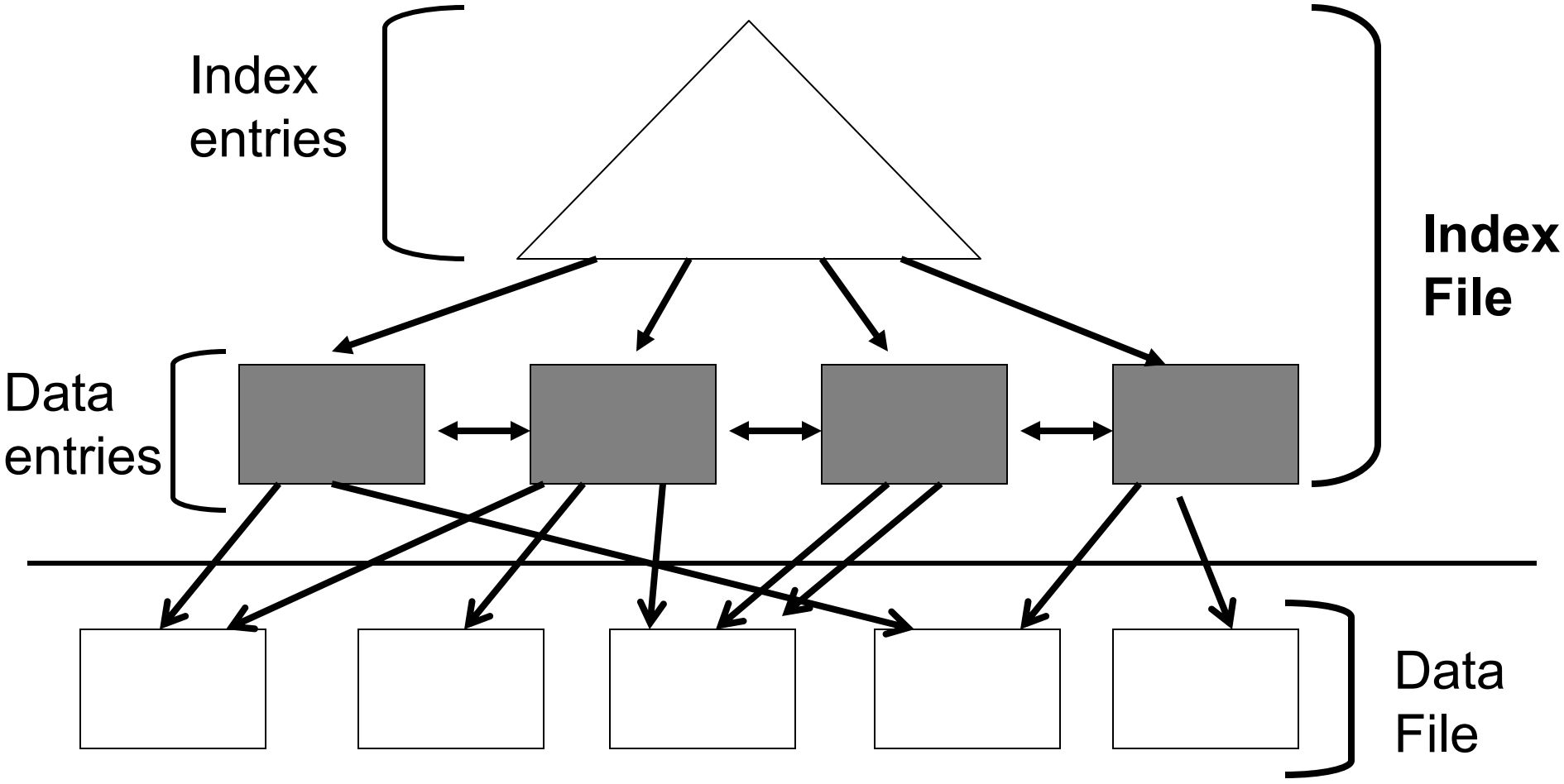
- Index is said to be clustered if
  - Data records in the file are organized as data entries in the index
  - If data is stored in the index, then the index is clustered by definition. This is option (1) from previous slide.
  - Otherwise, data file must be sorted in order to match index organization.
- Un-clustered index
  - Organization on data entries in index is independent from organization of data records.
  - These are options (2) and (3)
- File storing a relation *R* can only have 1 clustered index, but many un-clustered indices
  - Why?

Clustered Index



Records are stored at data entries

Unclustered Index



Some issues

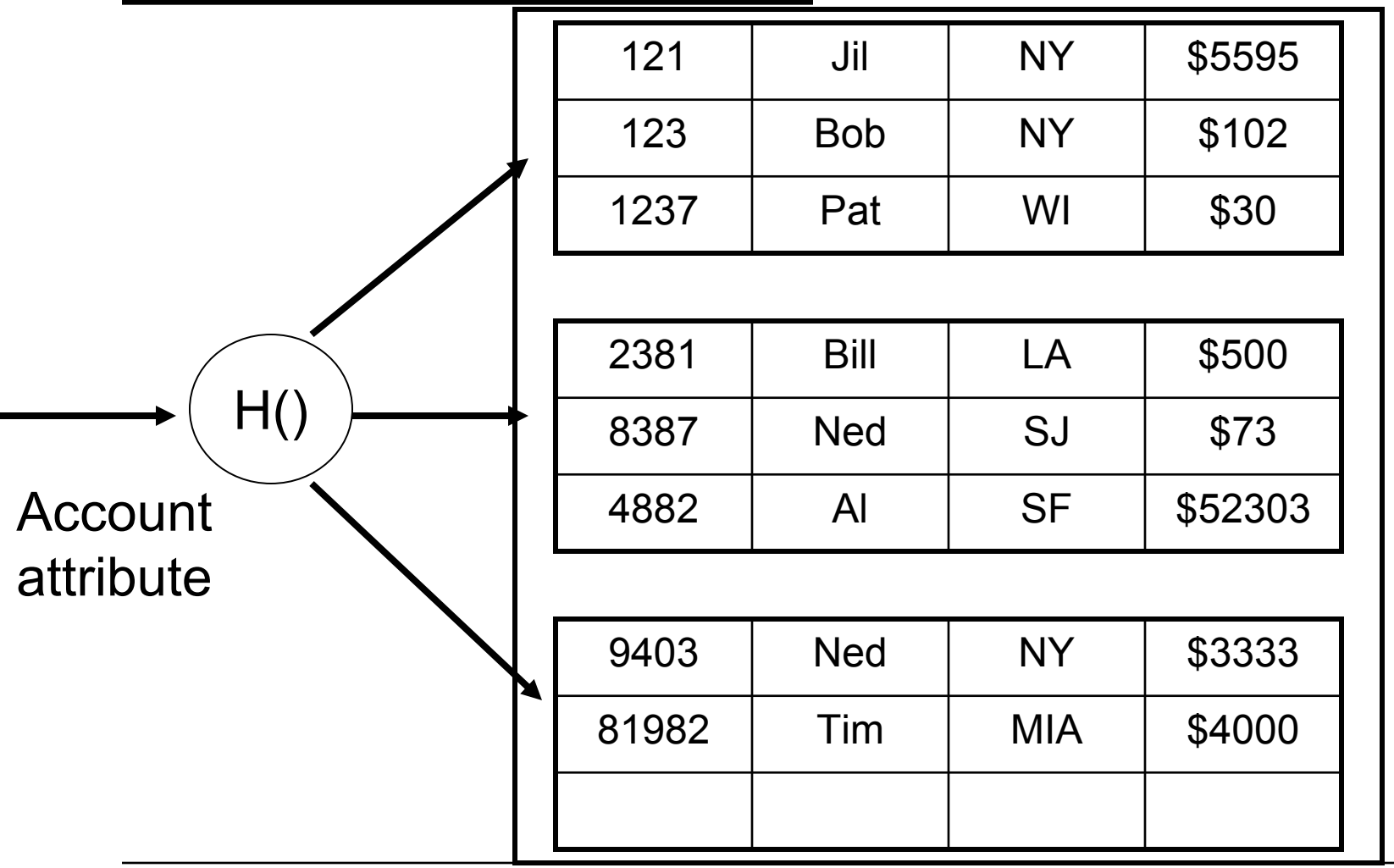
- Primary index
  - Index defined on the primary key of a relation
- Secondary index
  - Index defined on one or more attributes that are not a key
- Other nomenclature
  - Primary access method – access data as stored
    - Primary index
    - Index based on Index organization option (1)
  - Secondary access method – alternative access to data independent from native storage organization
    - Secondary index
    - Other methods such as sorting or hashing data into a temporary file

Hash-Based Indexing

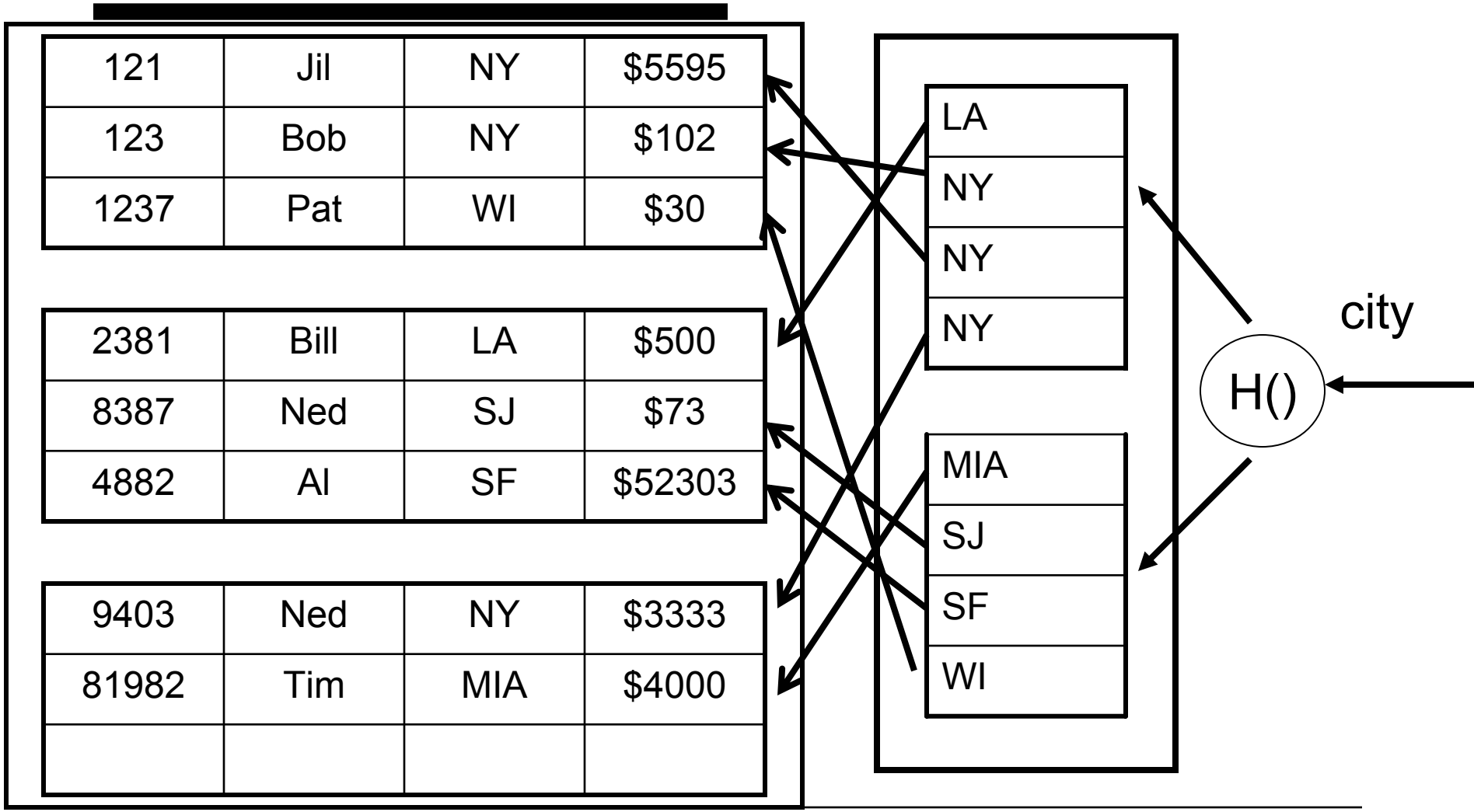
- Hash the records on some attribute(s)
- Accumulate records with same hash into value into same bucket
  - Bucket has a primary page and additional pages are linked in a list
- Hash function maps each record to a bucket
  - Ex. int Hash(char *str, int len) {

```
        int res = 0;
        for (int j =0; j  < len; ++j){
                res+=str[i];
        }
        return res % NUMBER_BUCKETS;
}
```

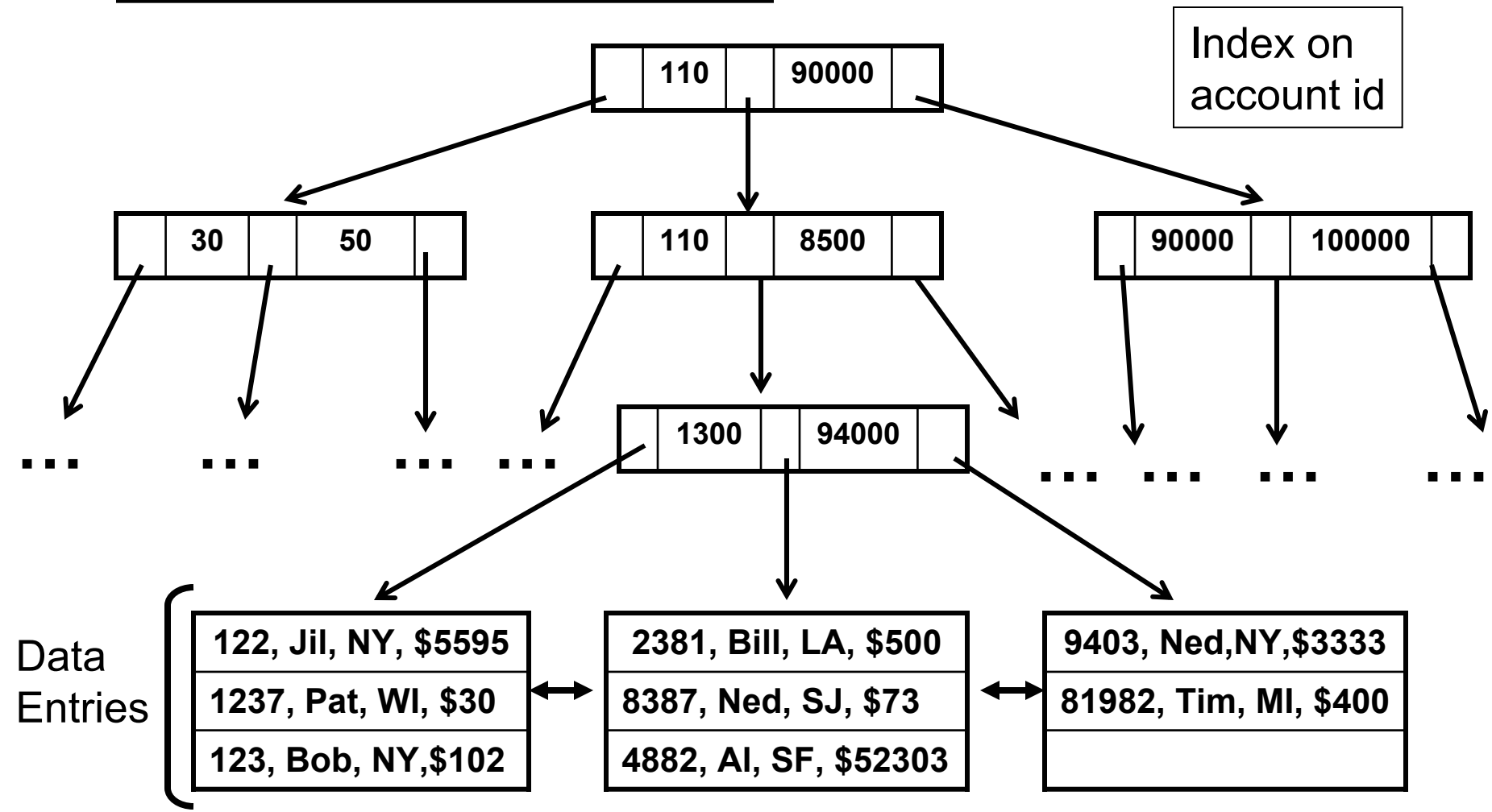
Hash Index (clustered)



Hash Index (Unclustered)



Tree-Structured Index



Some issues

- Data entries are maintained at the leaf level
- Each index entries are stored in disk pages
- We want to keep root page of index in the buffer pool while we are scanning the index
- In practice, finding data with an index will costs
  - N I/Os to read the index entries in the path of the tree.
  - K I/Os to read all the index entries
  - Total N + K I/O operations
- Most DBMS system manage to keep path between 2 and 3!
  - B+ tree
  - Fan-out - number of children in index nodes
    - Bigger means smaller tree height (smaller path to leaves!)

Estimating cost for operations

- The following are the typical operation applied to DBMS files (Heap, sorted, and index files)
  - Scan: fetch all the records in the file
  - Search with equality – find all records that satisfy an equality clause
  - Search with Range – find records all records that satisfy a range condition
    - Range queries
  - Insert a record – add a new record to the file
  - Delete a record – remove a record with a given rid from the file.