UCLA

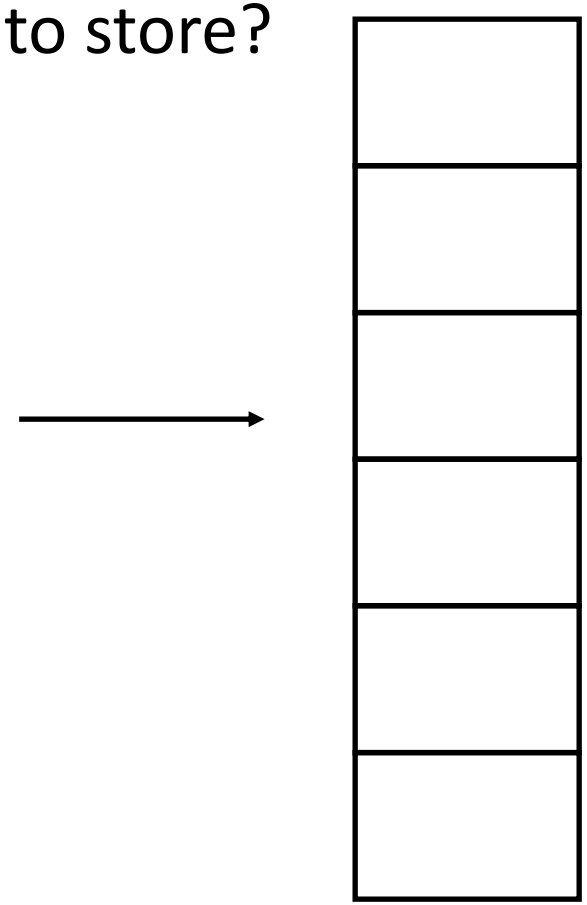
CS143: Files

Professor Junghoo “John” Cho

Files: Main Problem

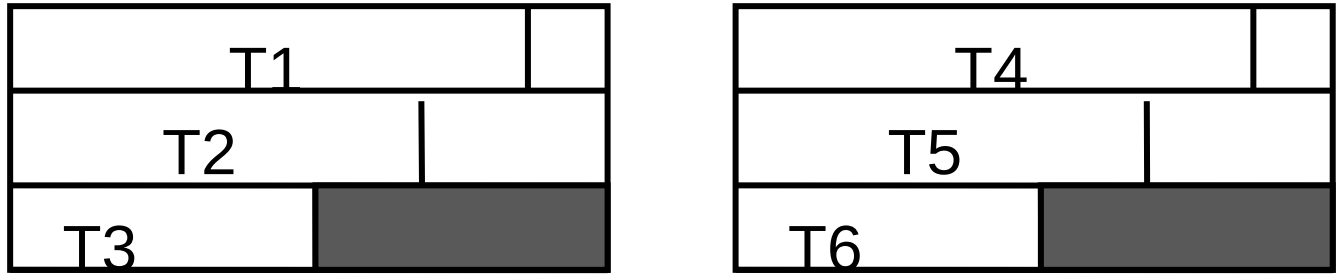
- How to store tables into disks?
- Q: 512Byte block. 80Byte tuple. How to store?

Jane	CS	3.7
Susan	ME	1.8
June	EE	2.6
Tony	CS	3.1

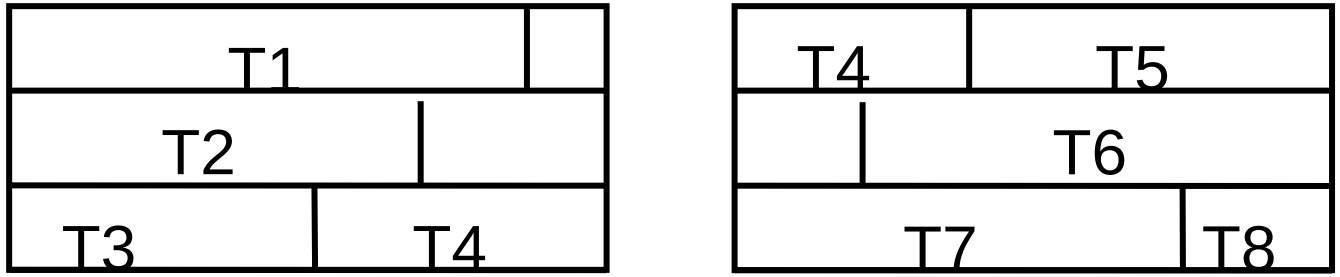


Spanned vs Unspanned

- Unspanned



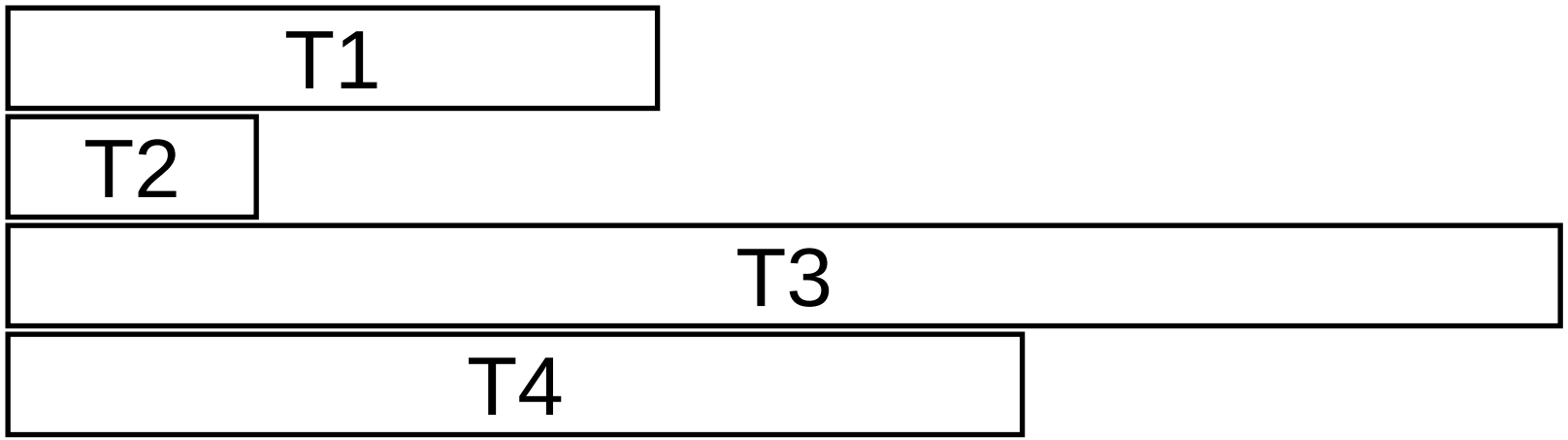
- Spanned



- Q: Maximum space waste for unspanned?

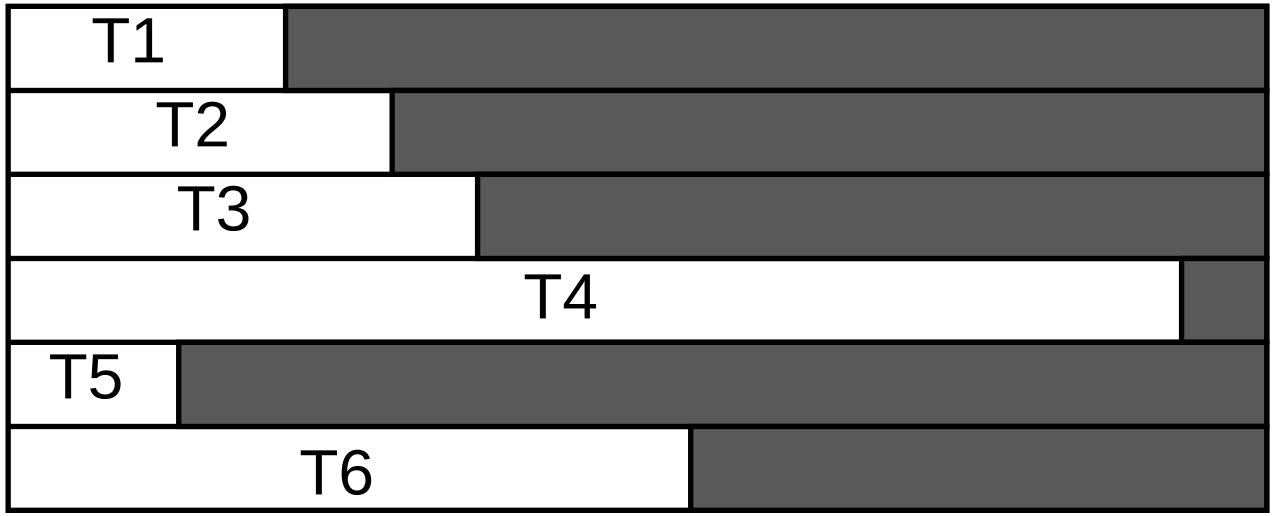
Variable-Length Tuples

- How do we store them?



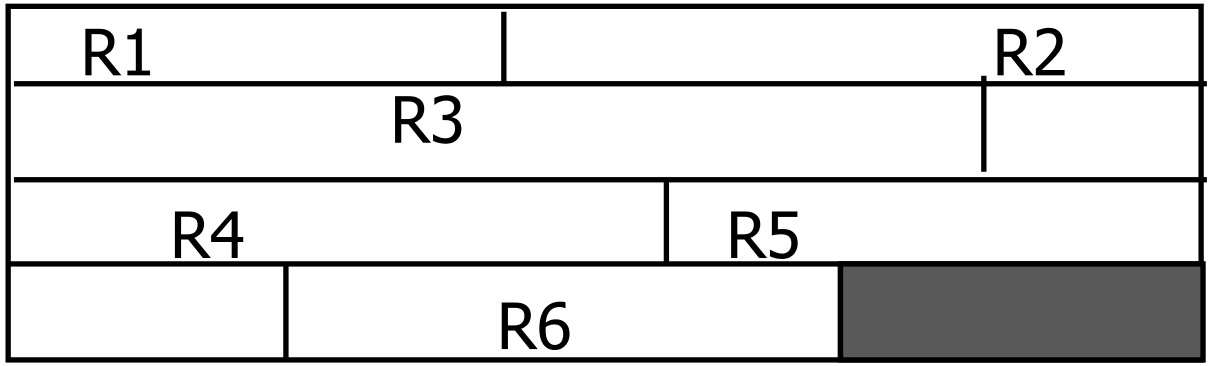
Reserved Space

- Reserve the maximum space for each tuple



- Q: Any problem?

Variable-Length Space

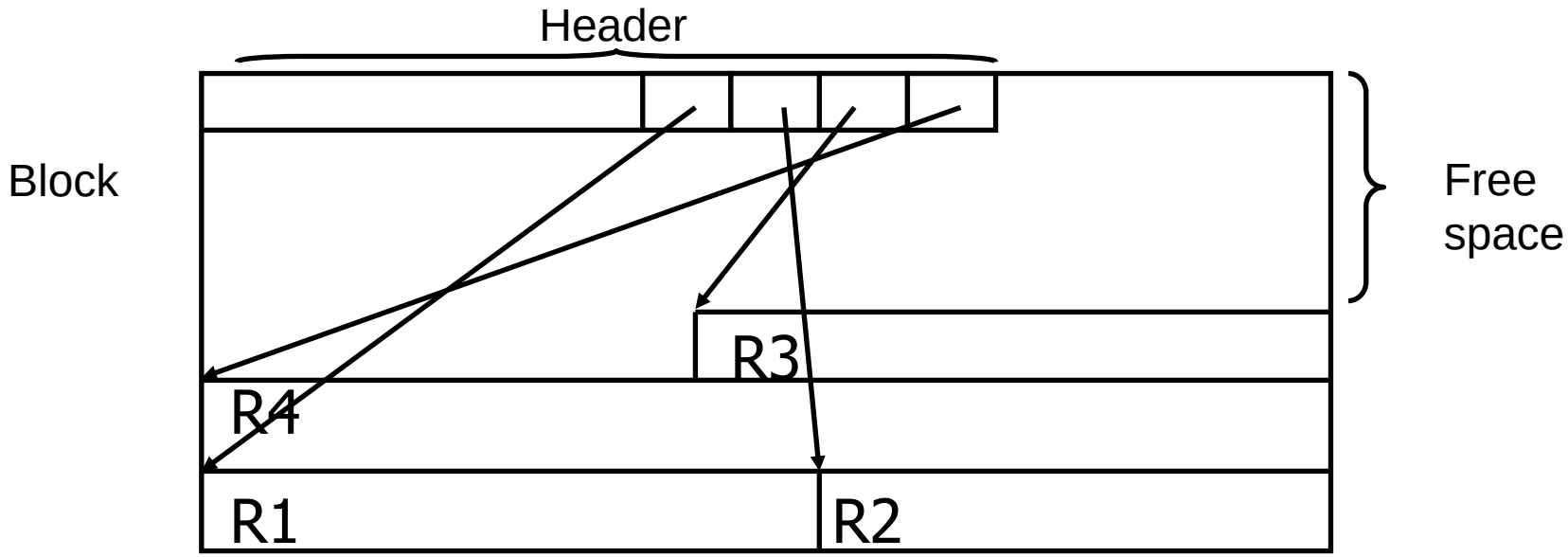


- Pack tuples tightly
- Q: How do we know the end of a tuple?

- Q: What to do for delete/update?

- Q: How can we “point to” to a tuple?

Slotted Page



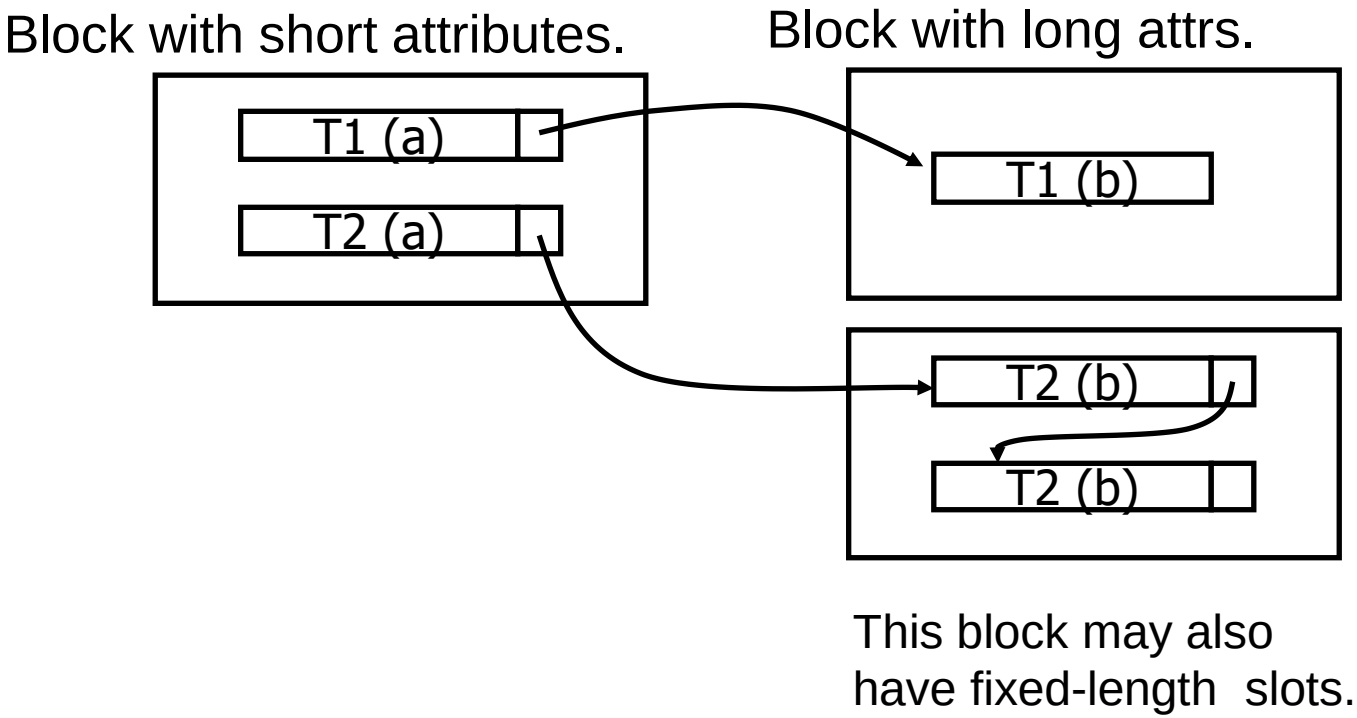
Q: How can we point to a tuple?

Long Tuples

- ProductReview(
  pid INT,
  reviewer VARCHAR(50),
  date DATE,
  rating INT,
  comments VARCHAR(4000))
- Block size 512B
- How should we store it?

Long Tuples

- Splitting tuples
  - Long attributes are stored separately (often as a separate file)



Column-Oriented Storage

- SELECT name FROM Students WHERE GPA > 3.7
- For analytical queries, reading the entire row of a tuple may not be needed
  - Row-oriented storage forces us to read the entire row even if most columns are not needed for query processing

Elaine	1 Le Conte	3.7
James	3 Mississippi	2.8
John	12 Wilshire	1.8
Peter	4 Olympic	3.9
Susan	7 Pico	1.0
Tony	12 Sunset	2.4

Column-Oriented Storage

- Store by column, not by row
- Unneeded Columns can be skipped for query processing
  - Better compression and caching behavior
- But
  - Column values of matching rows must be “joined”
  - Insertion/update of a row is more expensive (multiple IOs per row)

Elaine
James
John
Peter
Susan
Tony

1 Le Conte
3 Mississippi
12 Wilshire
4 Olympic
7 Pico
12 Sunset

3.7
2.8
1.8
3.9
1.0
2.4

Sequential File

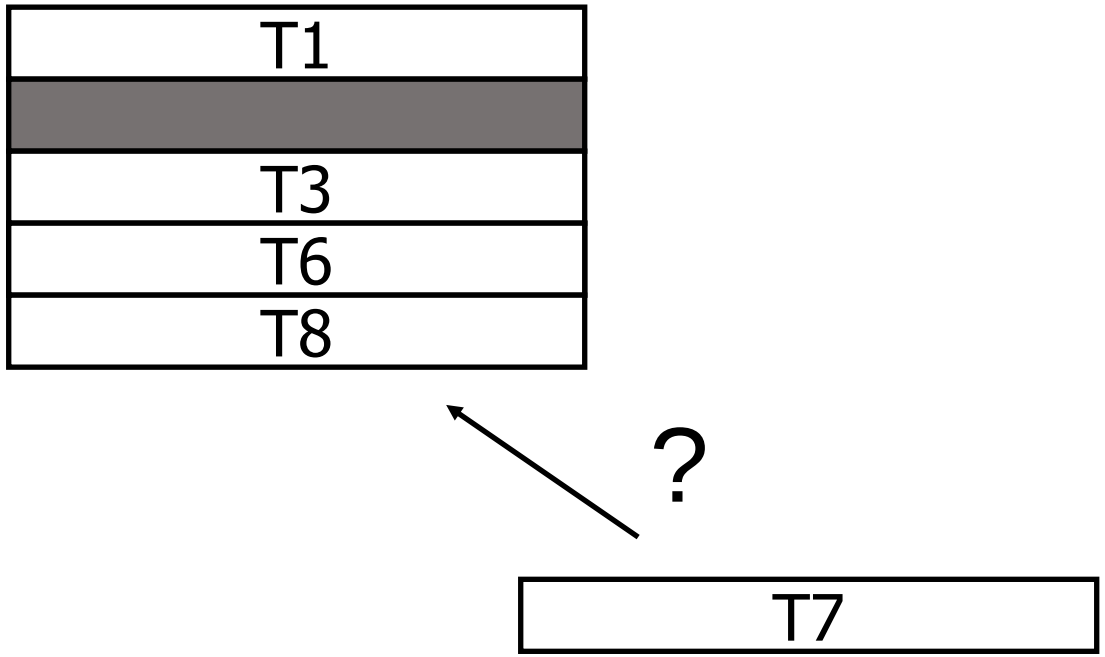
- Tuples are ordered by certain attribute(s) (search key)

Elaine	1 Le Conte	3.7
James	3 Mississippi	2.8
John	12 Wilshire	1.8
Peter	4 Olympic	3.9
Susan	7 Pico	1.0
Tony	12 Sunset	2.4

Search key: Name

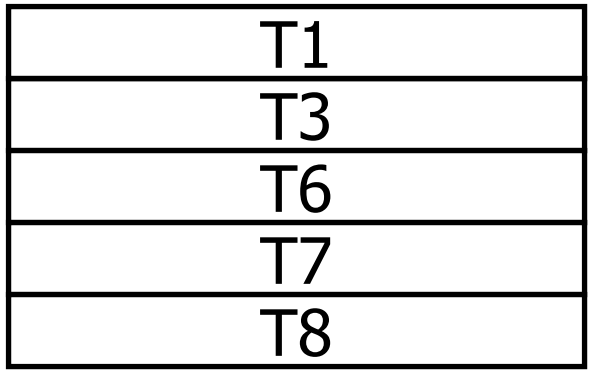
Sequencing Tuples

- Inserting a new tuple
  - Easy case

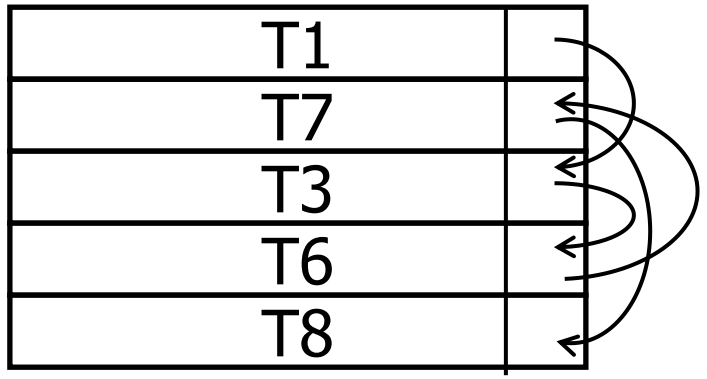


Two Options

1) Rearrange

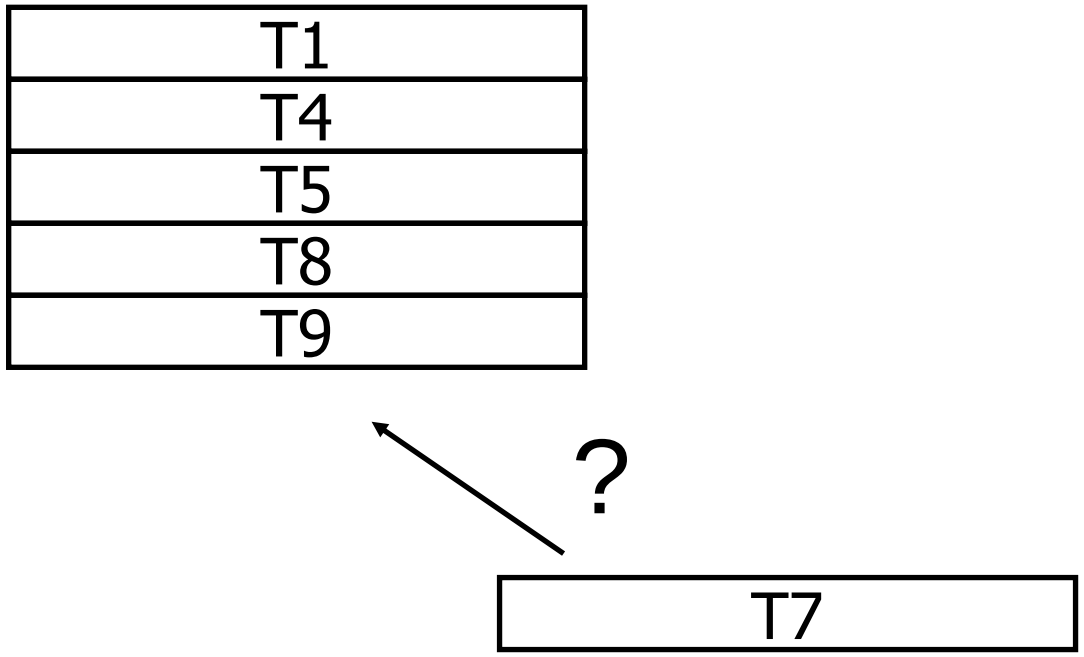


2) Linked list



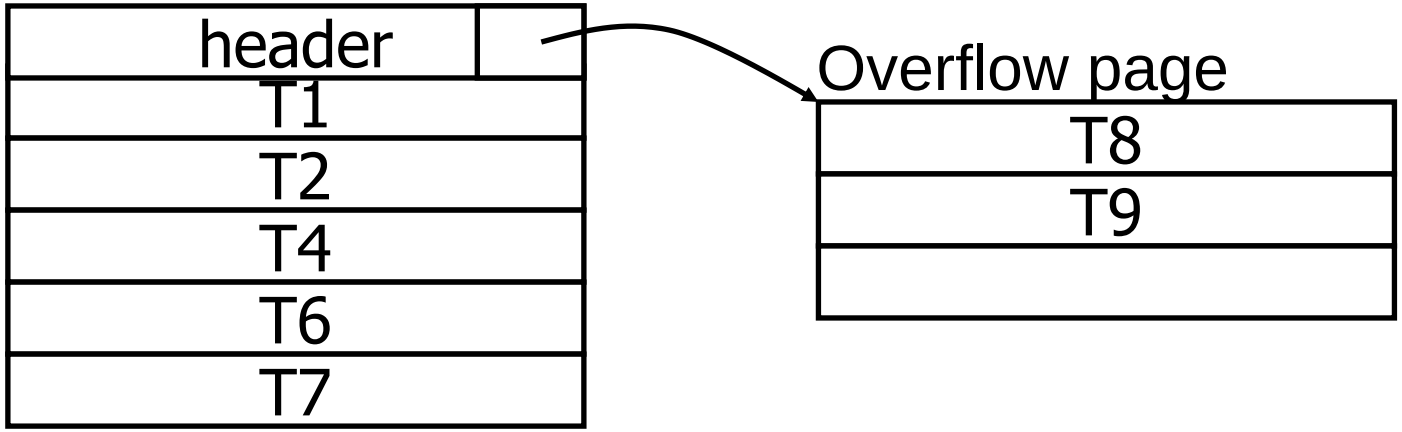
Sequencing Tuples

- Inserting a new tuple
  - Difficult case



Sequencing Tuples

- Overflow page



- Reserving free space to avoid overflow
  - PCTFREE in DBMS

    CREATE TABLE R(a int) PCTFREE 40

Things to Remember

- Spanned/unspanned tuples
- Variable-length tuples (slotted page)
- Long tuples
- Row-oriented vs column-oriented storage
- Sequential file and search key
  - Problems with insertion (overflow page)
  - PCTFREE

UCLA

CS143: Index

Professor Junghoo “John” Cho

Topics to Learn

- Index
- Dense index vs. sparse index
- Primary index vs. secondary index
  (= clustering index vs. non-clustering index)
- Multi-level index
- Indexed Sequential Access Method (ISAM)

Basic Problem

- SELECT *
  FROM Student
  WHERE sid = 30

sid	name	GPA
20	Susan	3.5
60	James	1.7
70	Peter	2.6
40	Elaine	3.9
30	Christy	2.9

- How can we answer the query?

Random-Order File

- How do we find sid=30?

sid	name	GPA
20	Susan	3.5
60	James	1.7
70	Peter	2.6
40	Elaine	3.9
30	Christy	2.9

Sequential File

- Table sequenced by sid. Find sid=30?

sid	name	GPA
20	Susan	3.5
30	James	1.7
40	Peter	2.6
50	Elaine	3.9
60	Christy	2.9

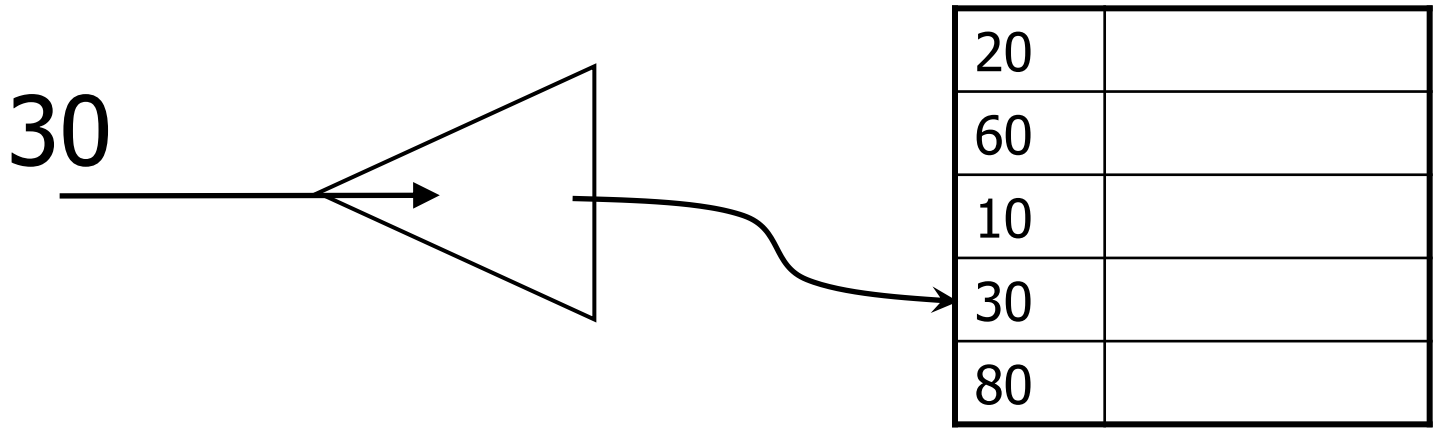
Binary Search

- 100,000 tuples
- Q: How many blocks to read?

- Any better way?

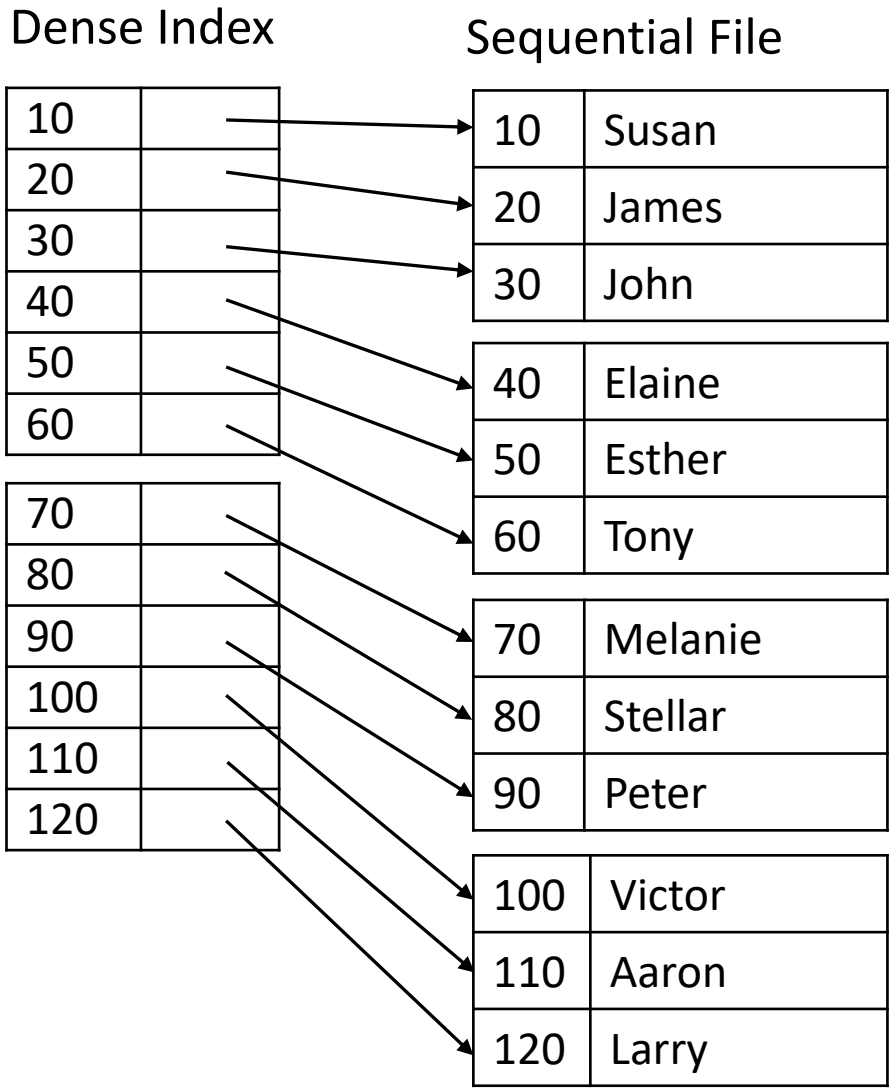
Index: Basic Idea

- Build an “index” on the table
  - An auxiliary structure to help us quickly locate a tuple given a “search key”



Dense, Primary Index

- Primary index (=clustering index)
  - Underlying table is sequenced by a key
  - Index is built on on the same key (= search key)
- Dense index
  - One (key, pointer) index entry per every tuple
- Search algorithm
  - Find the key from index and follow pointer
  - Maybe through binary search
- Q: Why dense index?
  - Isn't binary search on the file the same?



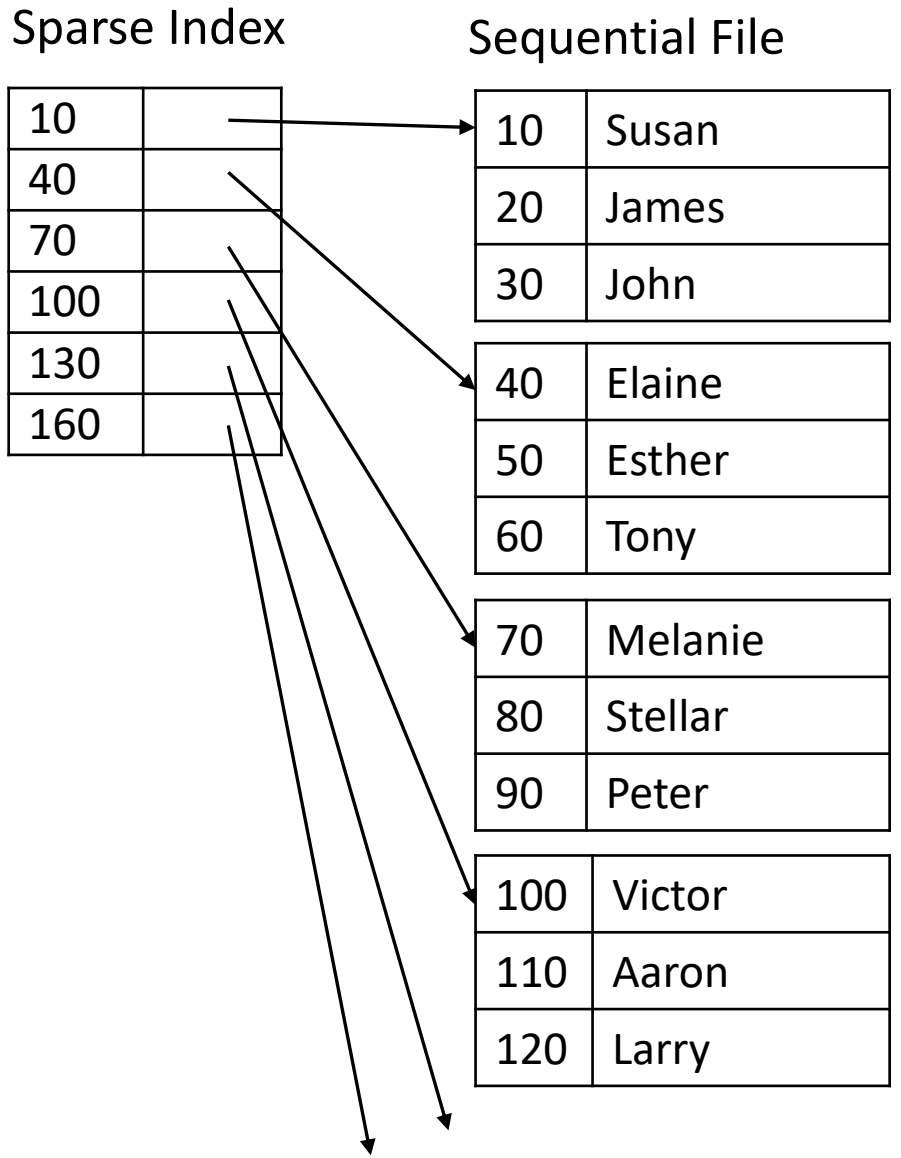
Why Dense Index?

- Example
  - 100,000,000 tuples (900-bytes/tuple)
  - 4-byte search key, 4-byte pointer
  - 4096-byte block. Unspanned tuples
- Q: How many blocks for table (how big)?

- Q: How many blocks for index (how big)?

Sparse, Primary Index

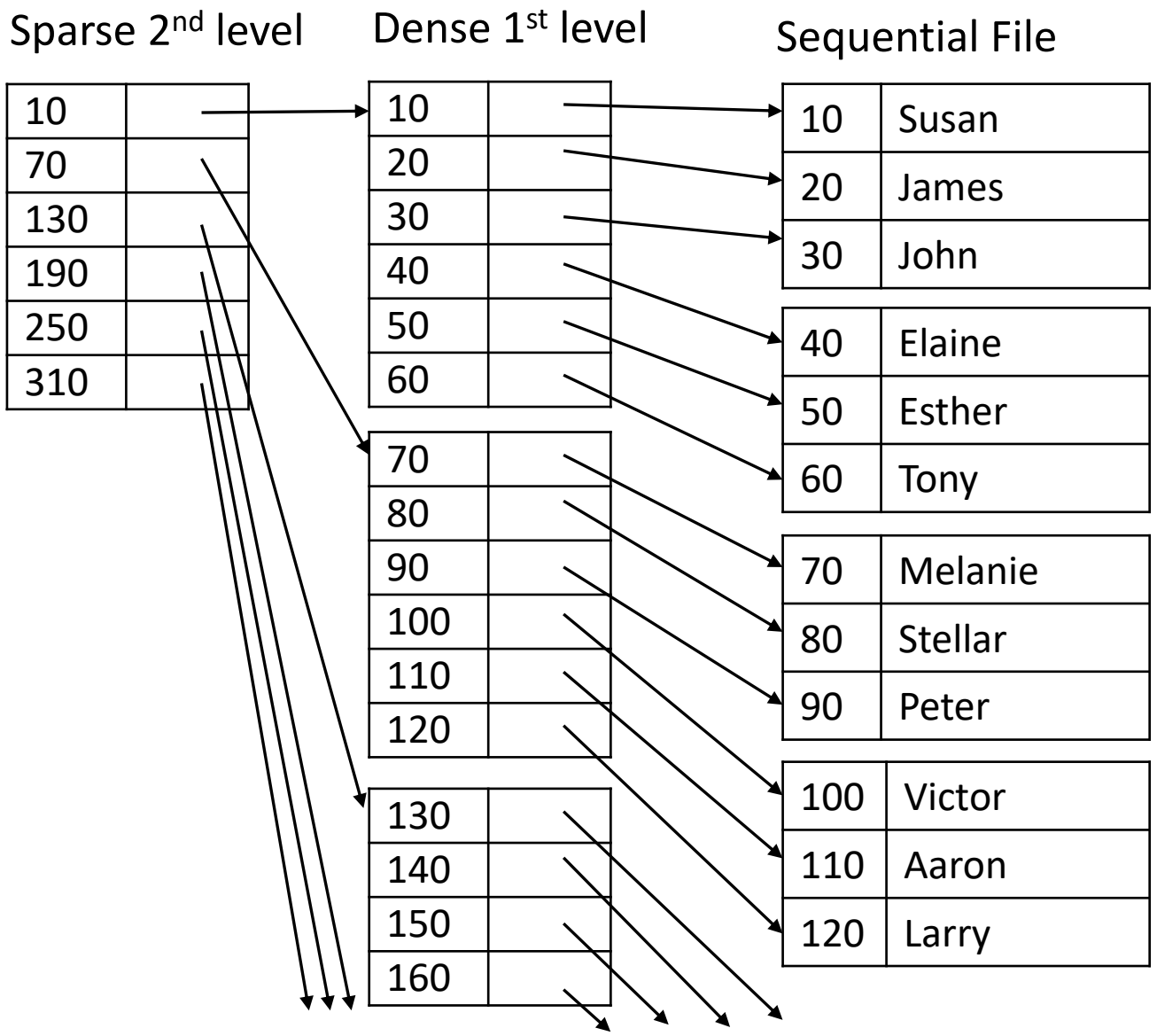
- Primary Index
  - Index is built on on the same search key as the underlying sequential file
- Sparse index
  - (key, pointer) pair per every “block”
  - (key, pointer) pair points to the first tuple in the block
- Q: How can we find 80?



Multi-level index

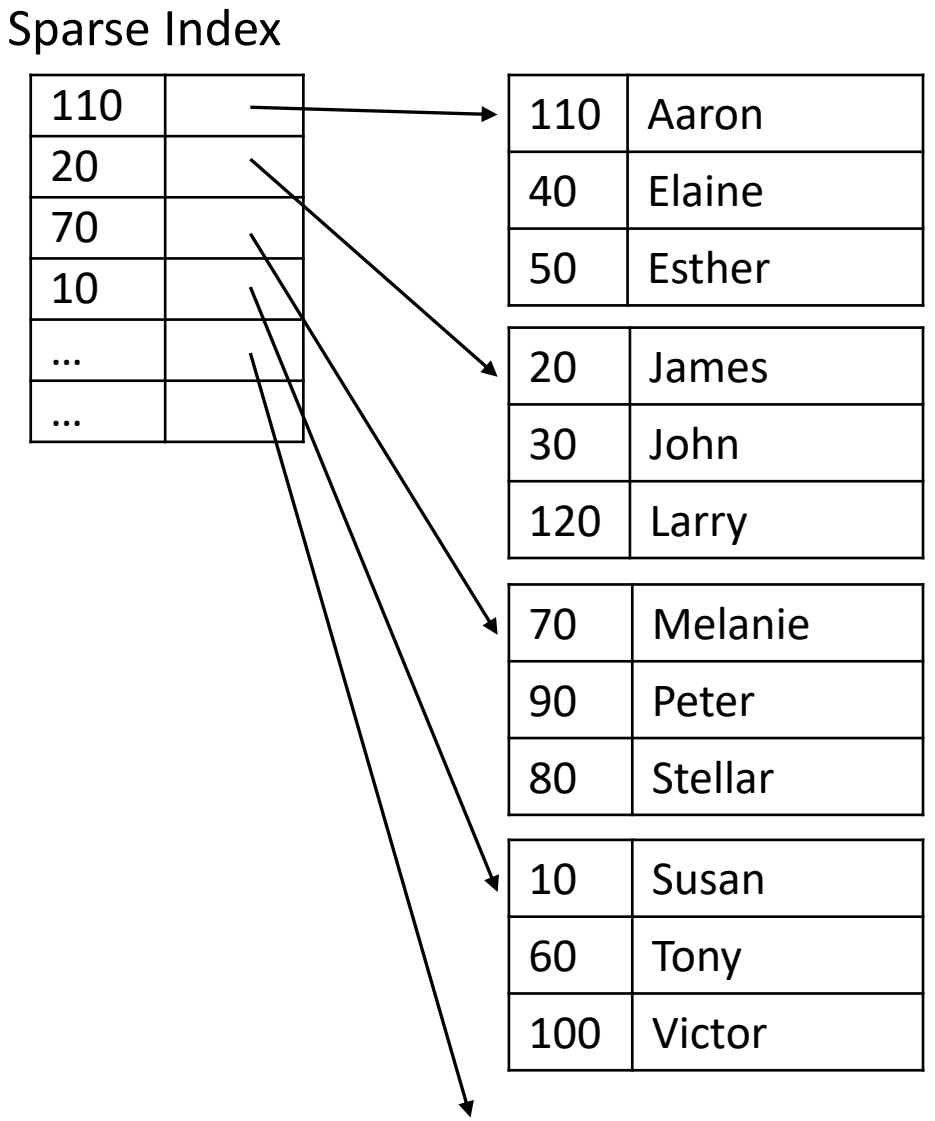
Q: Why multi-level index?

Q: Does dense, 2nd level index make sense?



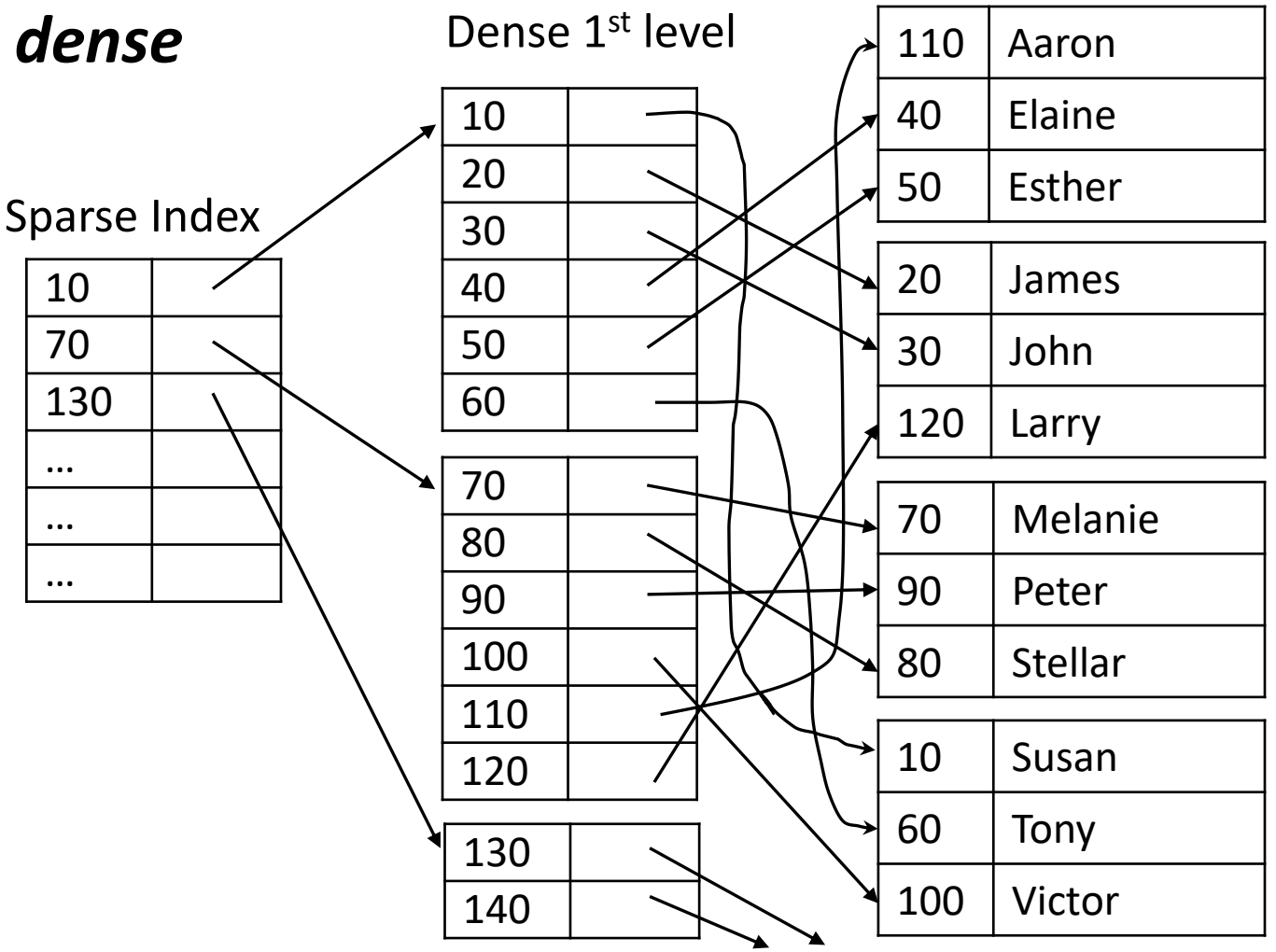
Secondary (non-clustering) Index

- Secondary (non-clustering) index
  - When tuples in the table are not ordered by the index search key
    - Index on a non-search-key for sequential file
    - Unordered file
- Q: What index?
  - Does sparse index make sense?



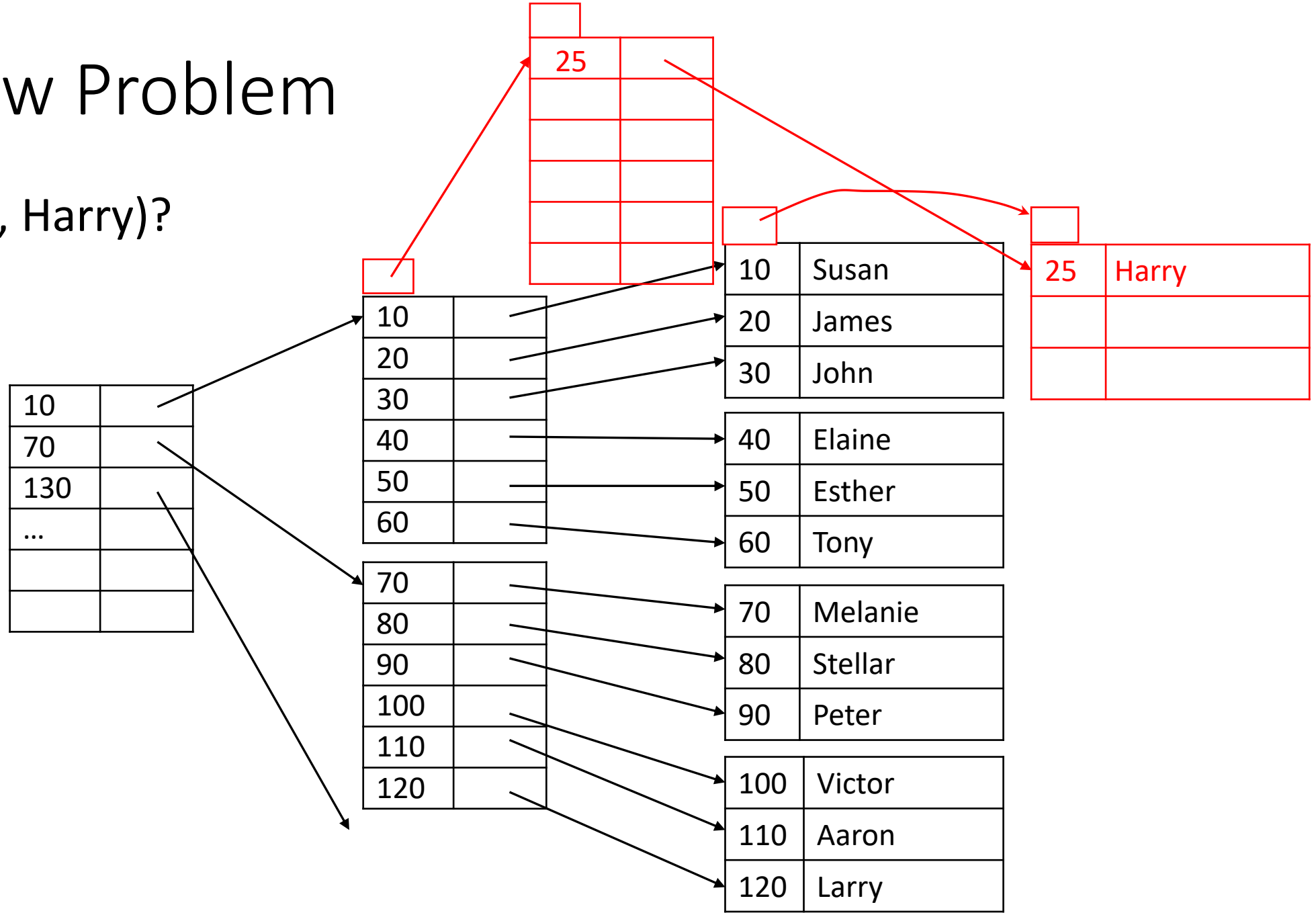
Secondary index

- First level must be ***always dense***
- Sparse from second level



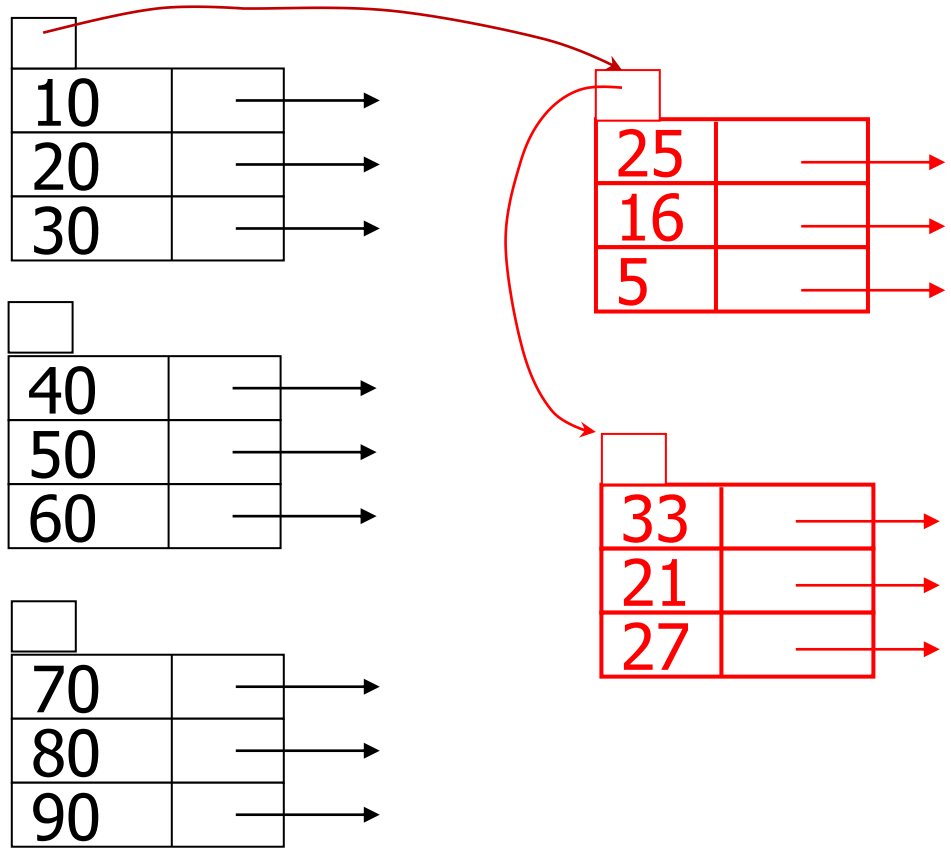
Overflow Problem

Q: Insert (25, Harry)?



Performance Problem after many insertions

- After many insertions, long chain of overflow pages



Indexed Sequential Access Method (ISAM)

- Advantage
  - Simple
  - Sequential blocks
- Disadvantage
  - Not suitable for updates
  - Becomes ugly (loses sequentiality and balance) over time

Index Creation in SQL

- CREATE INDEX <indexname> ON <table>(<attr>,<attr>,…)
- Example
  - CREATE INDEX sid_idx ON Student(sid)
    - Creates a B+tree on the attributes
    - Speeds up lookup on sid

Primary (Clustering) Index

- MySQL:
  - Primary key becomes the clustering index
- DB2:
  - CREATE INDEX idx ON Student(sid) CLUSTER
  - Tuples in the table are sequenced by sid
- Oracle: Index-Organized Table (IOT)
  - CREATE TABLE T (
    ...
) ORGANIZATION INDEX
  - B+tree on primary key
  - Tuples are stored at the leaf nodes of B+tree
- Periodic reorganization may still be necessary to improve range scan performance

Important terms

- Search key (≠ primary key)
- Primary index vs. secondary index
  - Clustering index vs. non-clustering index
- Dense index vs. sparse index
- Multi-level index
- Indexed Sequential Access Method (ISAM)