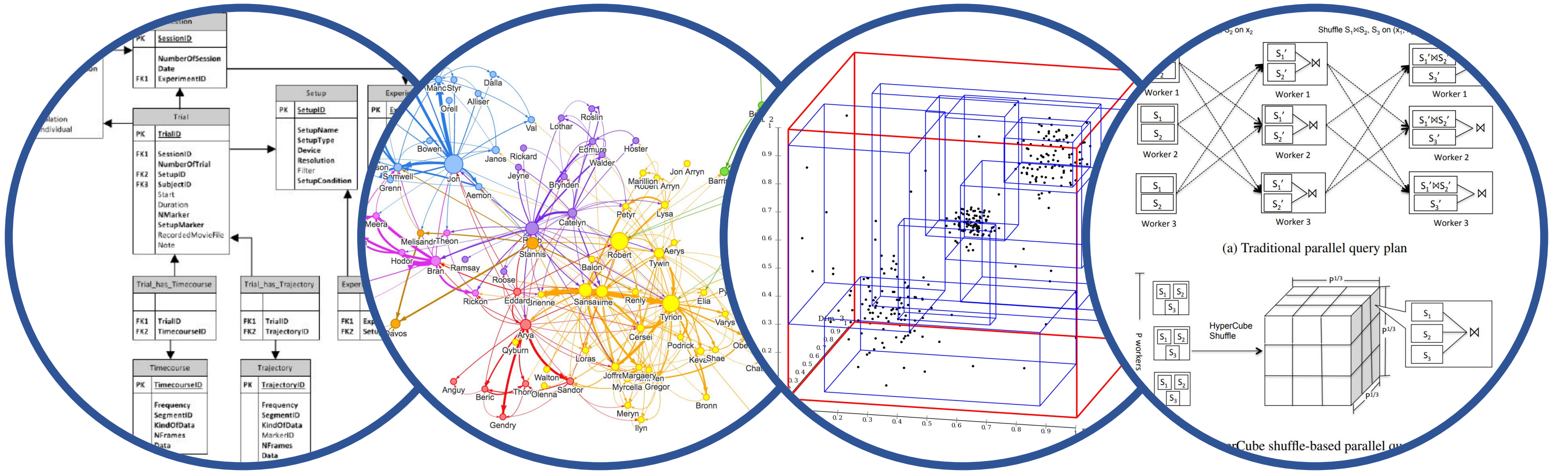


Introduction to Data Management

Database Tuning

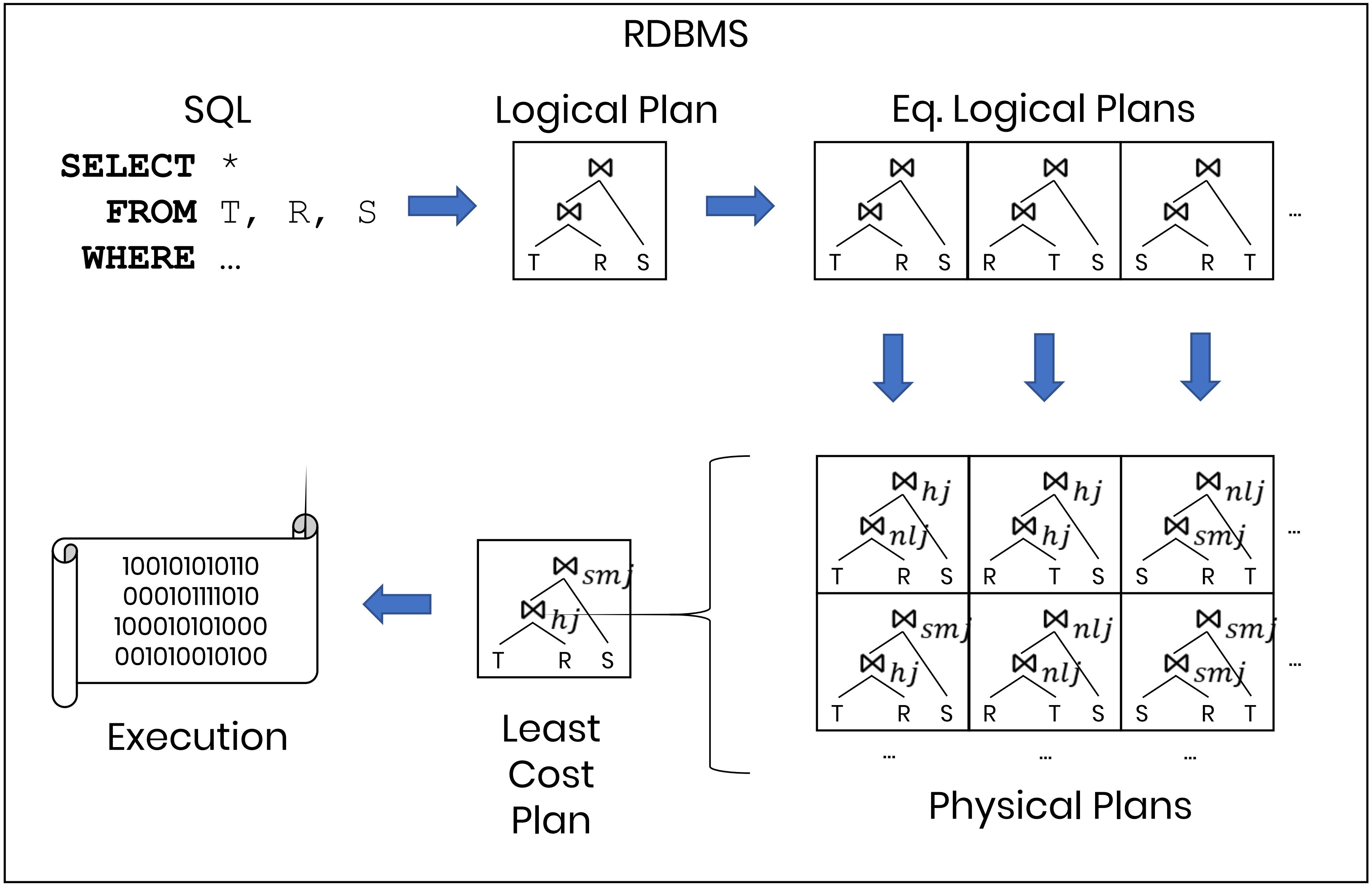
Alyssa Pittman
Based on slides by Jonathan Leang, Dan Suciu, et al

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

Goals for Today

- We gave a baseline for what join algorithms (and respective costs) were possible
- Use DB structures to expand optimization options

Recap – Plan Enumeration



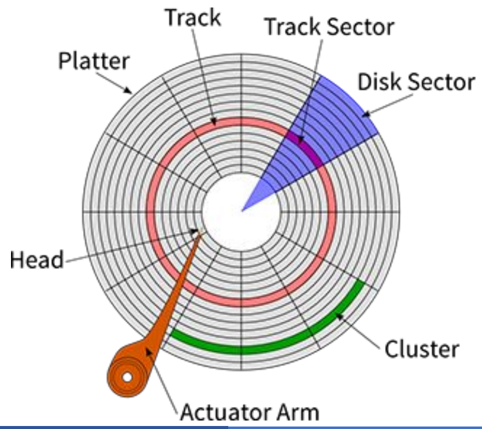
Recap – Assumptions

For this class we make a lot of assumptions

- **Disk-based storage**
  - HDD not SDD
- **Row-based storage**
  - Tuples are stored contiguously
- **IO cost** only
  - One disk access is ~100000x more expensive than one main memory access
- **Cold cache**
  - No data preloaded into main memory

Recap – Disk Storage

- Can only read **1 block per read operation**
  - Usually 512B to 4kB
- **Sequential disk reads are faster than random ones**
  - Cost ~1–2% random scan = full sequential scan



Recap – Making Cost Estimations

- RDBMS keeps statistics about our tables
  - **B(R)** = **# of blocks** in relation R
  - **T(R)** = **# of tuples** in relation R
  - **V(attr, R)** = **# of distinct values** of attr in R

Recap – Disk Storage

- Tables are stored as files
  - Heap file Unsorted tuples
    - Nested-Loop Joins
      - Block-at-a-time Nested Loop Join (cost = B(R)+B(R)*B(S))
      - Block-Nested-Loop Join (cost = B(R)+B(R)/N*B(S))
    - Hash Join (B(R)<M, cost = B(R)+B(S))
    - Sort-Merge Join (B(R)+B(S)<M, cost = B(R)+B(S))
  - **Sequential file** Sorted tuples

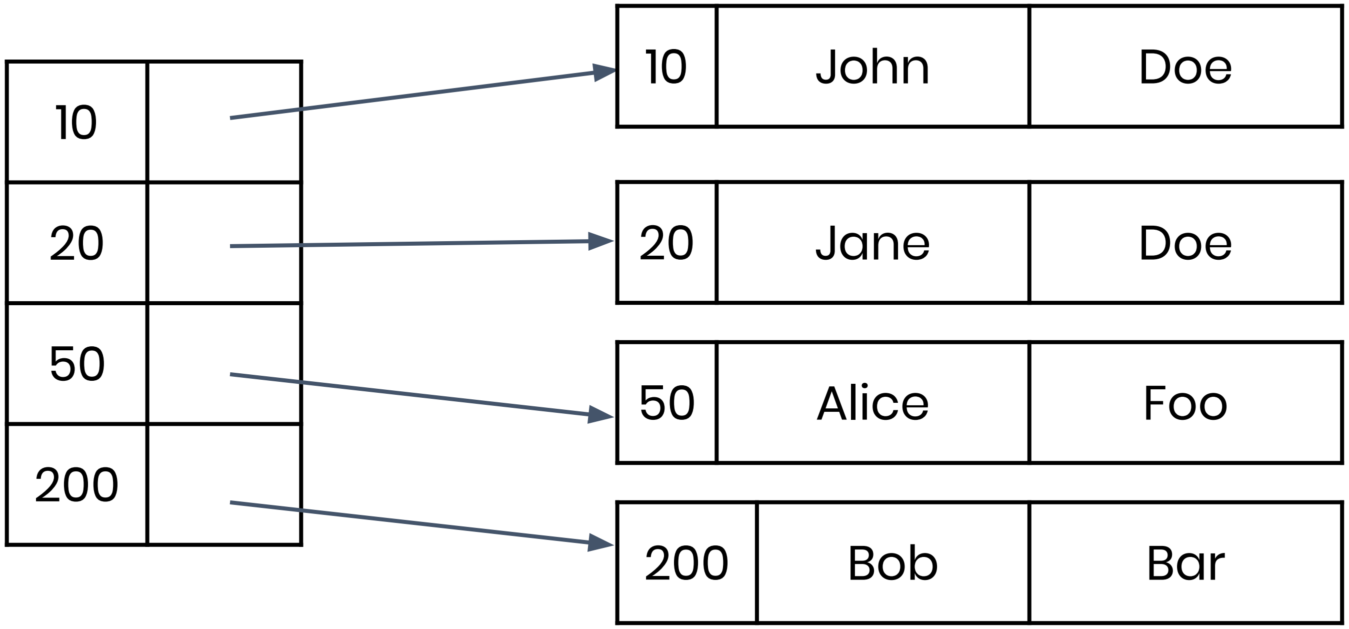
Outline

- Index structures
- Index join cost estimation
- Database tuning

Indexing

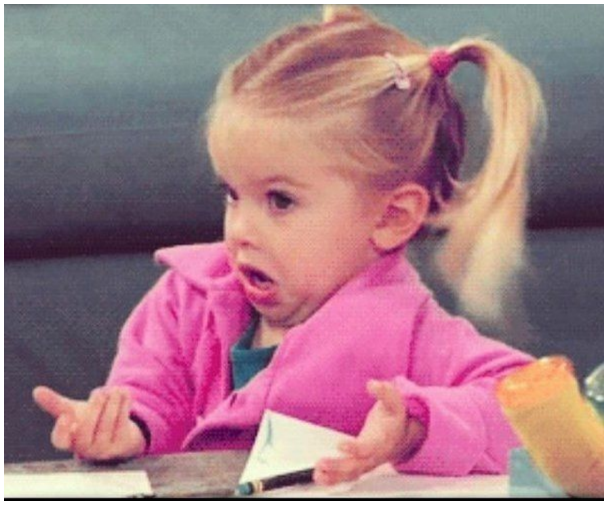
An **index** is an additional file allowing **fast access** to records given a **search key.**

It stores **(key, value)** pairs:
(attribute, pointer to the record)



Which key?

- Primary key: uniquely identifies a tuple
- Candidate Keys: other keys defined on relations
- Key of the sequential file: how the data file is sorted, if at all
- Index key: how the index is organized

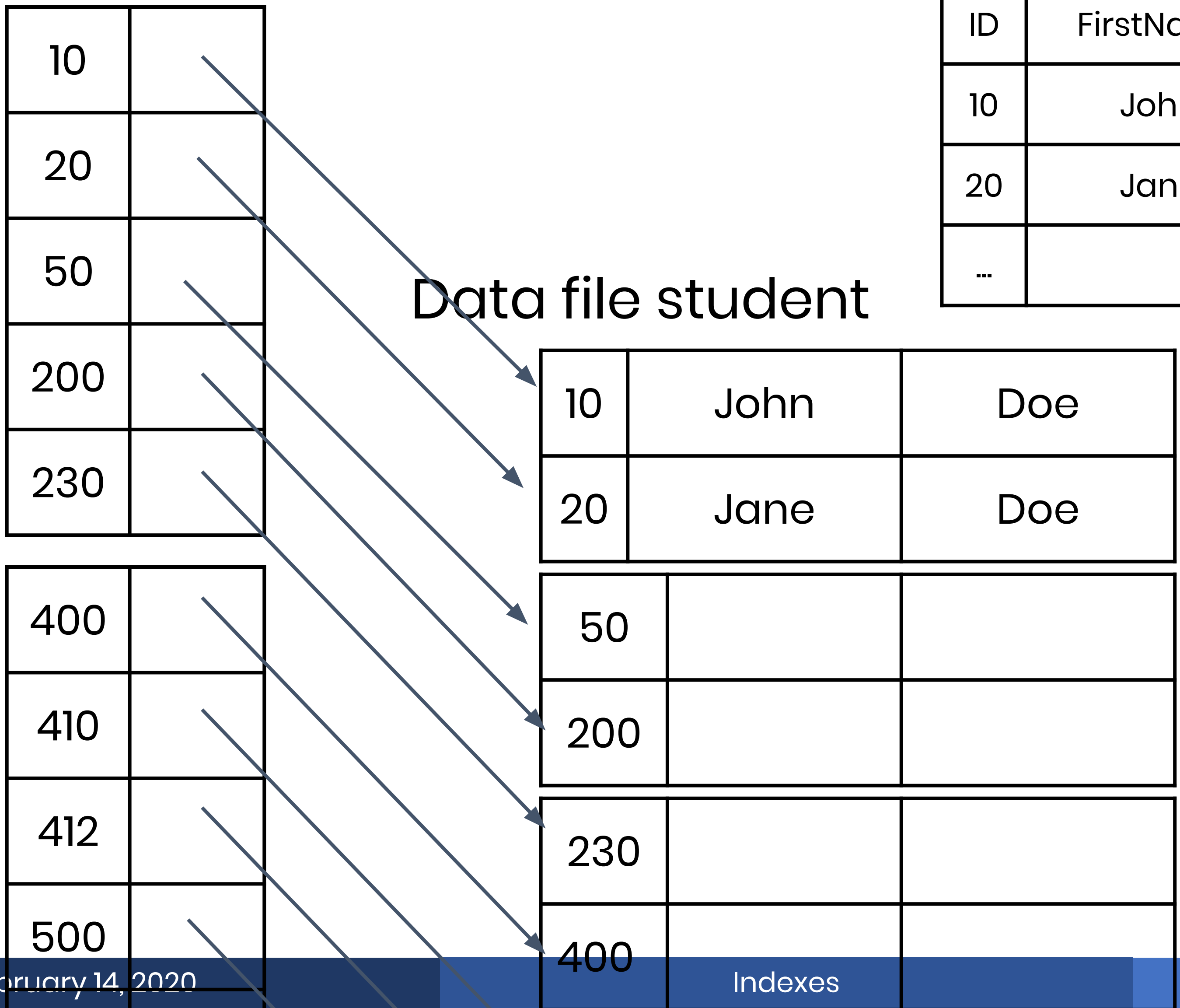


Example: Student, sorted data file

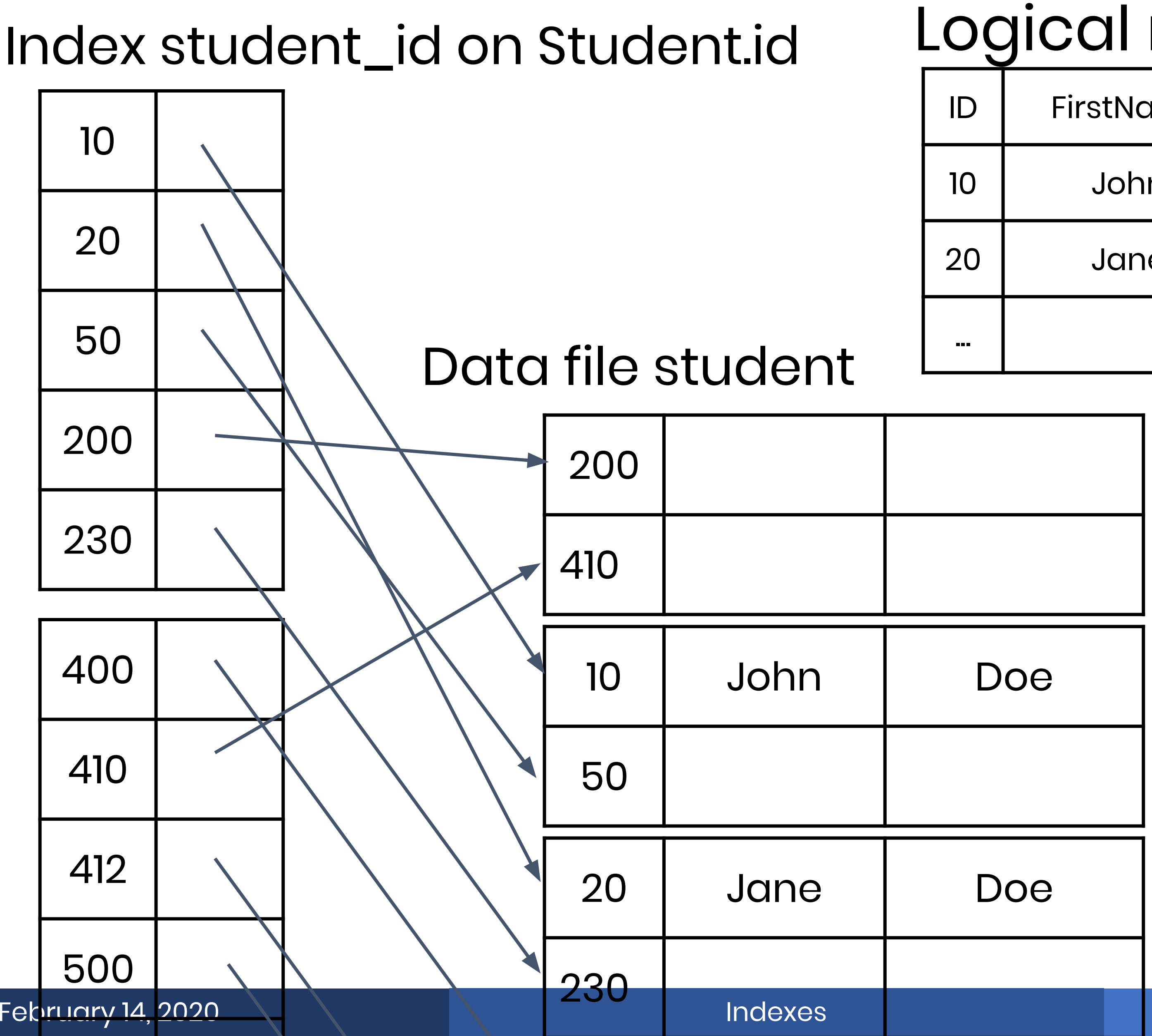
Index student_id on Student.id

Logical relation

ID	FirstName	LastName
10	John	Doe
20	Jane	Doe
...		...



Example: Student, unsorted data file



Logical relation

ID	FirstName	LastName
10	John	Doe
20	Jane	Doe
...		...

Indexing

- Indexes (for this class) can be assumed to be **already loaded into memory**
- An index does not have to contain all tuple data
  - Only key values are stored in the index
  - If an index contains all tuple data it is called a "covering index"
- A table can have multiple indexes

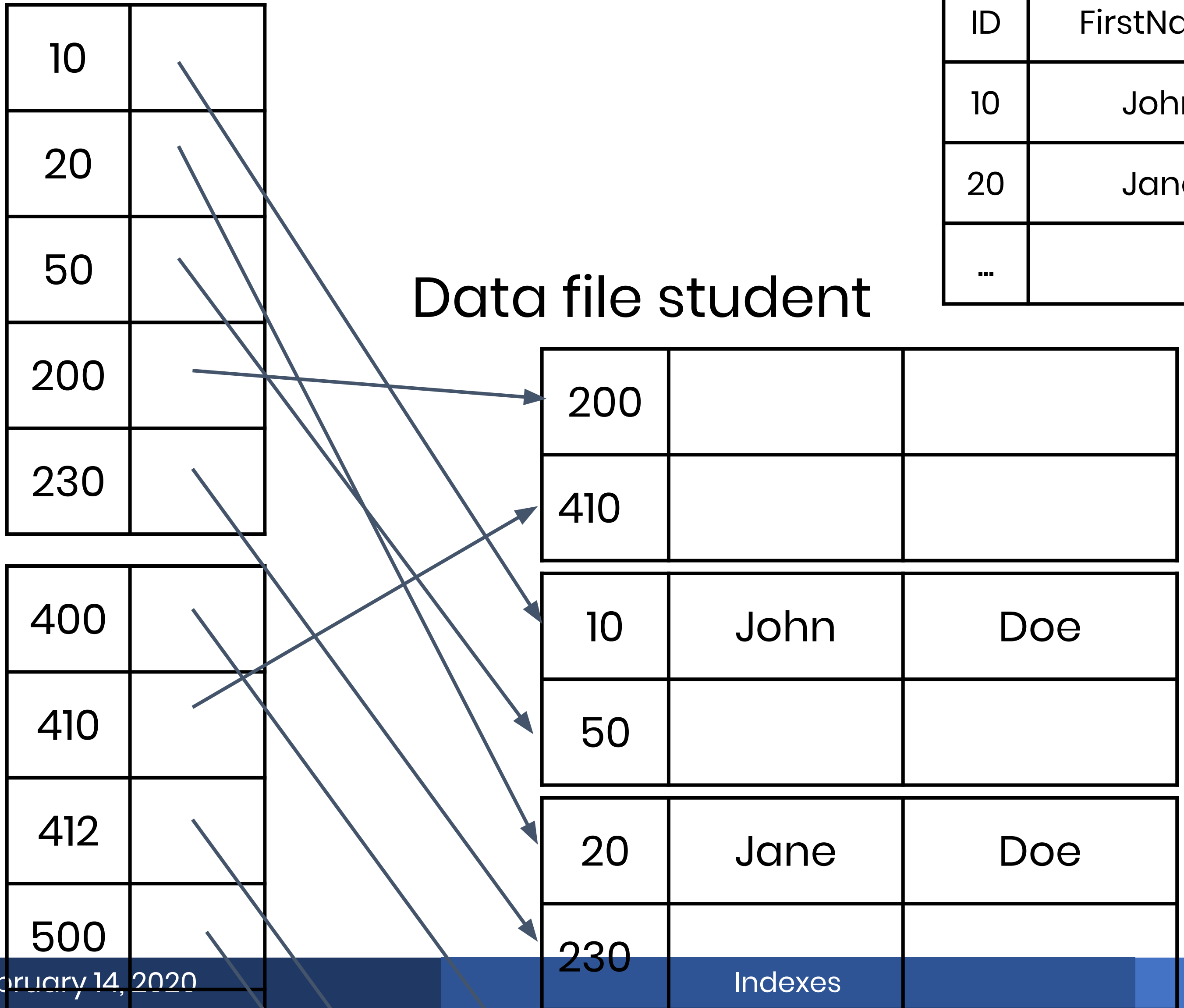
Index Structures

- **Hash Index**
- B+ Tree Index
  - Clustered
  - Unclustered
- R Tree
- Radix Tree
- Bloom Filter
- Hilbert Curves
- Inverted Index
- …

Hash Index

Index student_id on Student.id

Data file student



Logical relation

ID	FirstName	LastName
10	John	Doe
20	Jane	Doe
…		…

Index Structures

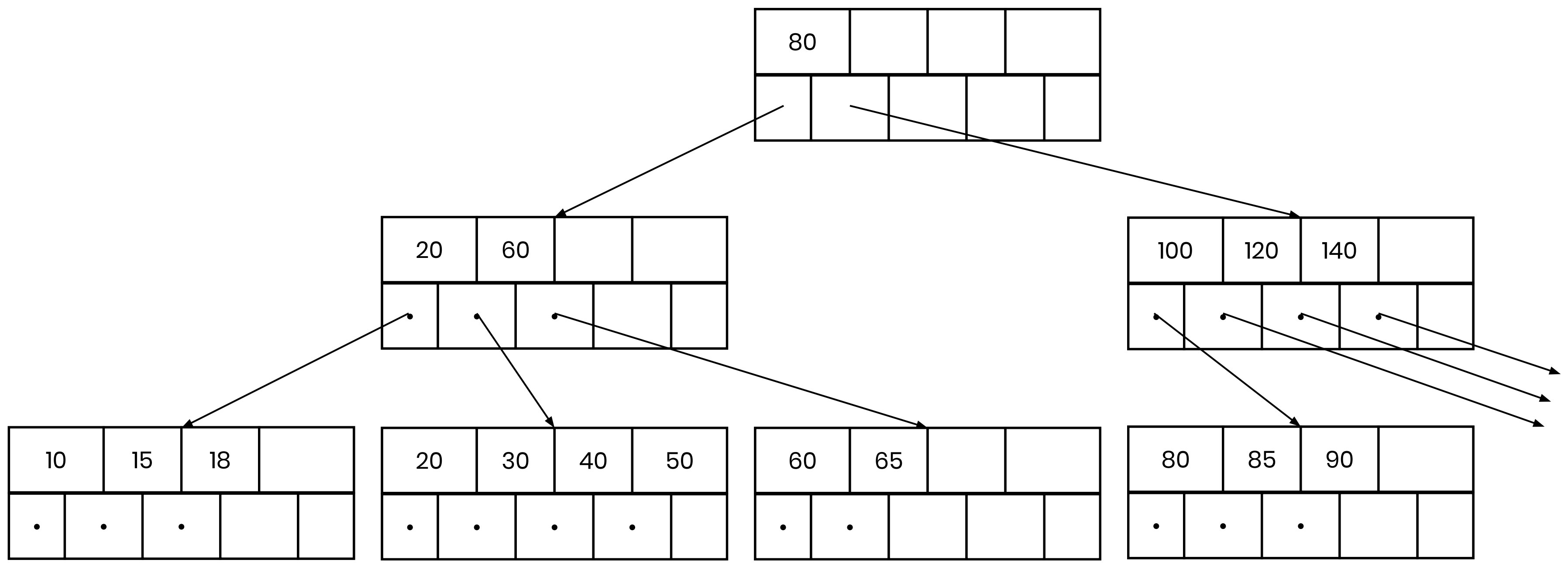
- Hash Index
- **B+ Tree Index**
    - Clustered
    - Unclustered
- R Tree
- Radix Tree
- Bloom Filter
- Hilbert Curves
- Inverted Index
- ...

What is a B Tree?

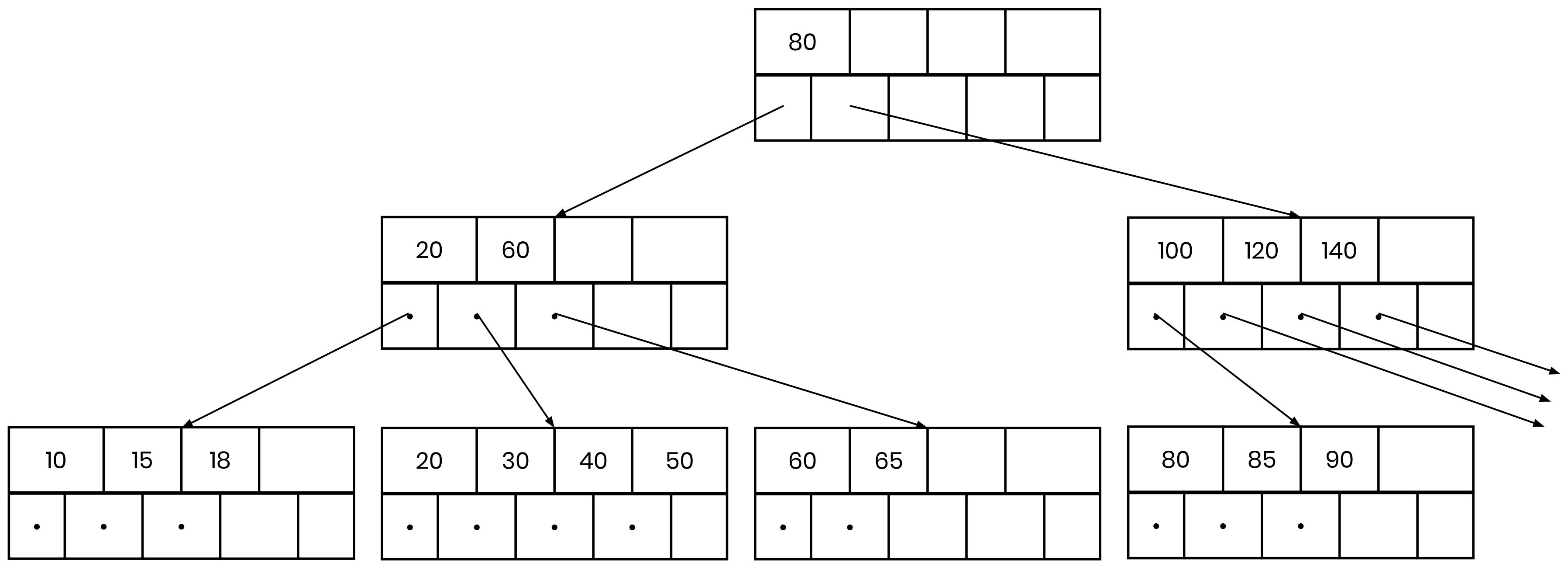
"What, if anything, the *B* stands for has never been established." – Wikipedia

- **Search tree** (like a binary search tree)
  - Nodes annotate max values
  - Large number of children per node
- Tree/node structure that is memory efficient

What is a B Tree?

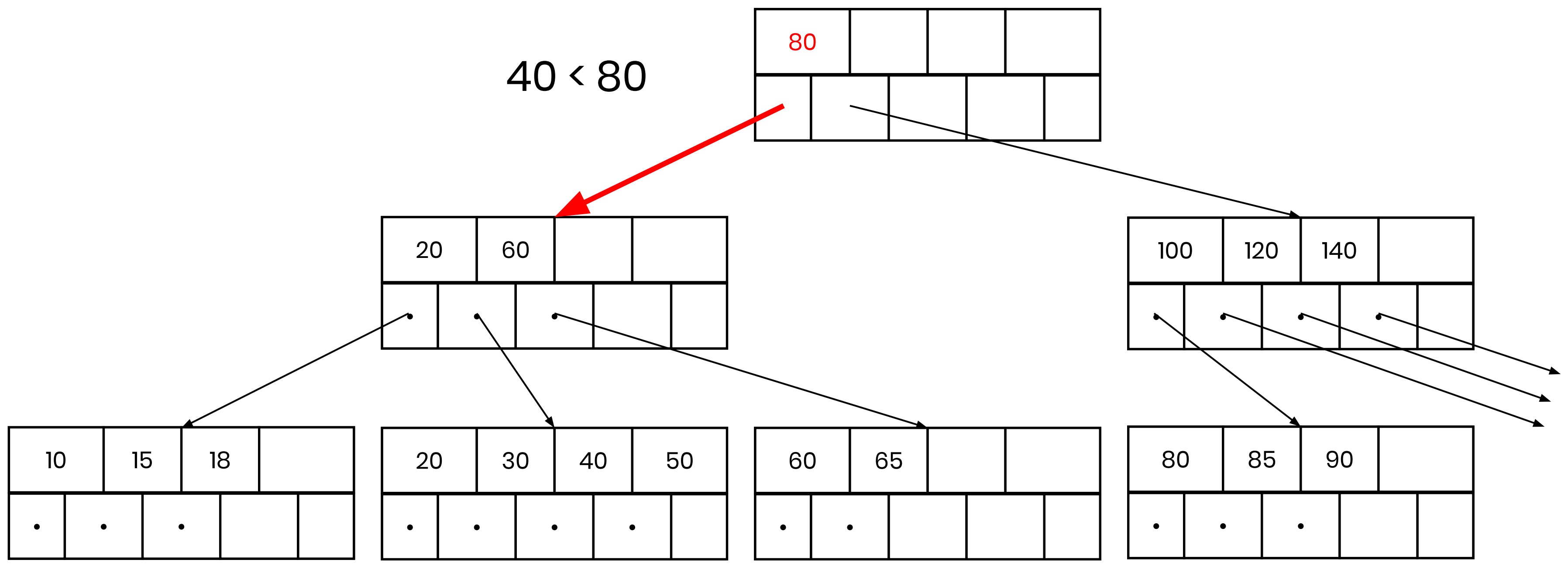


What is a B Tree?



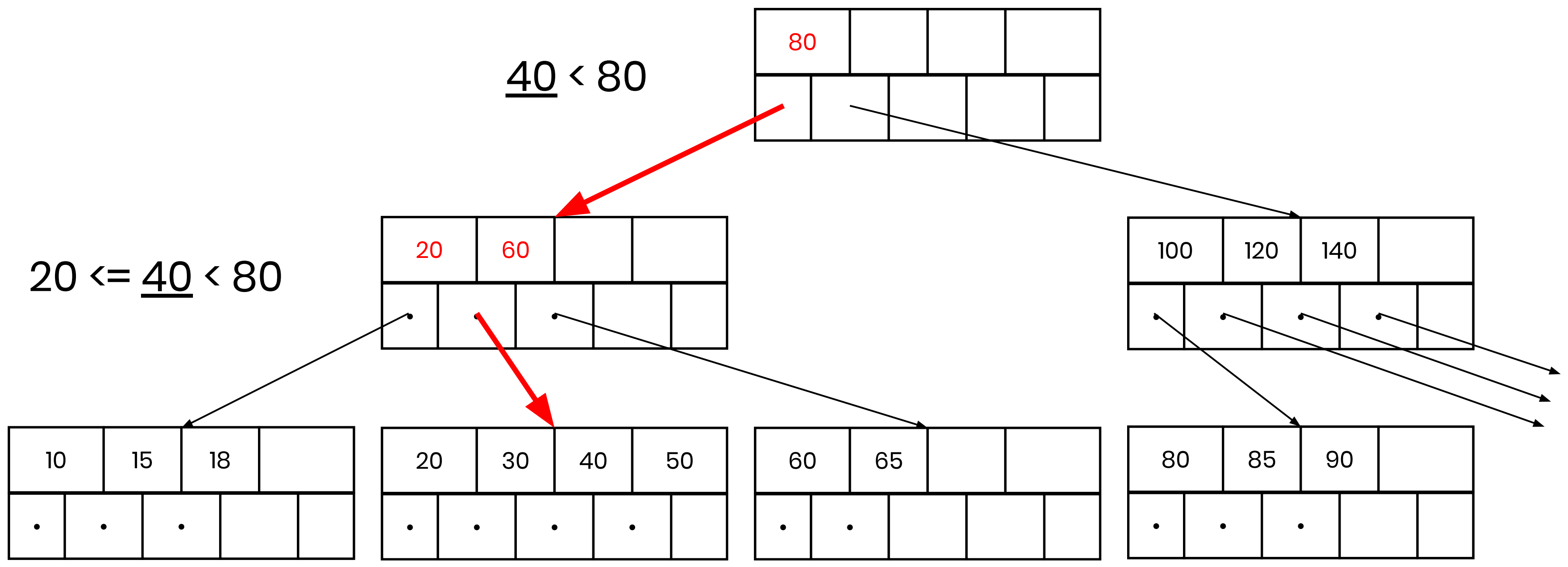
Find the value 40

What is a B Tree?



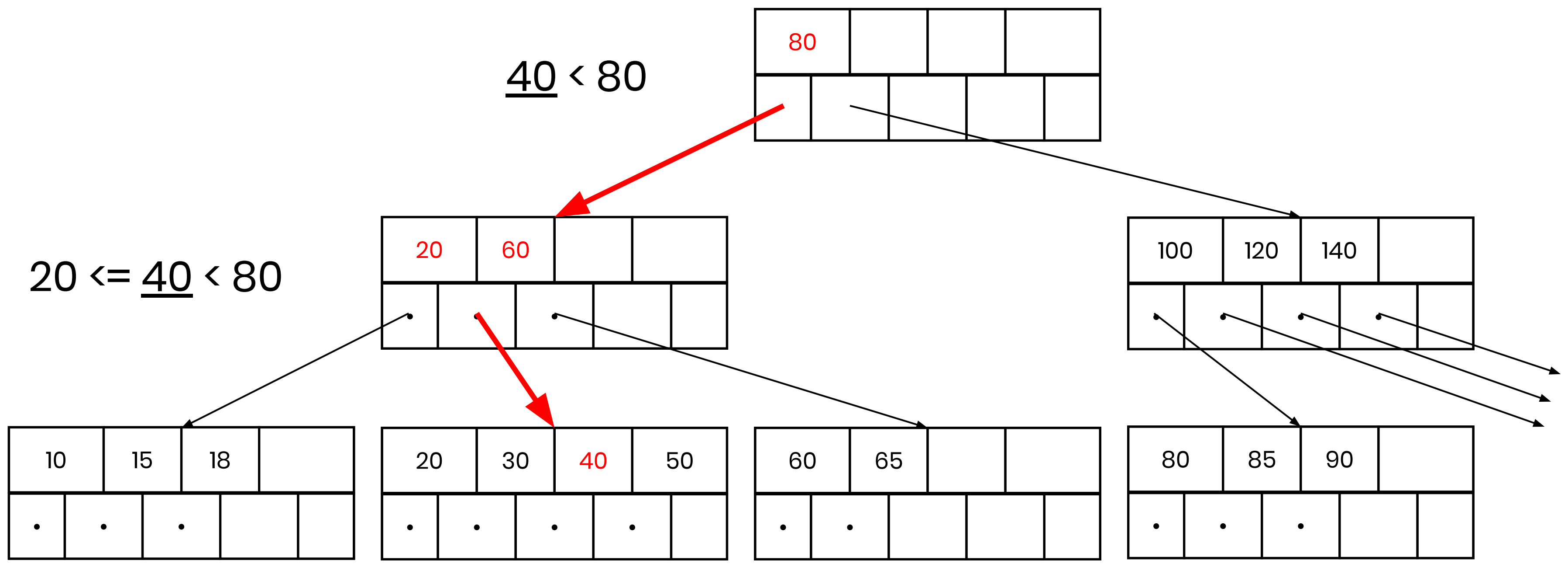
Find the value 40

What is a B Tree?



Find the value 40

What is a B Tree?

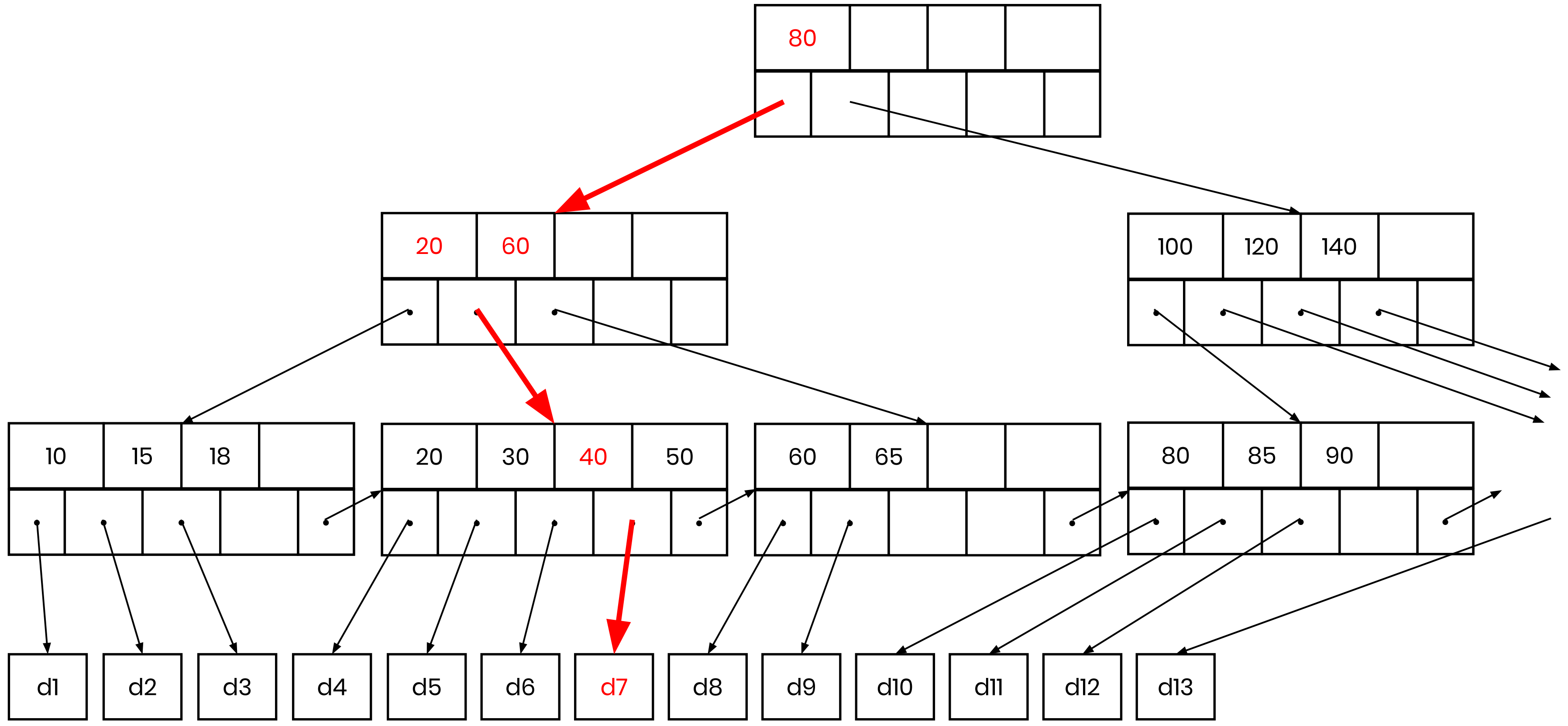


Find the value 40

How is a B+ Tree Different?

- Leaf nodes point to data
  - Data is searchable by **key** value annotated by the node labels
- Leaf nodes form a linked list

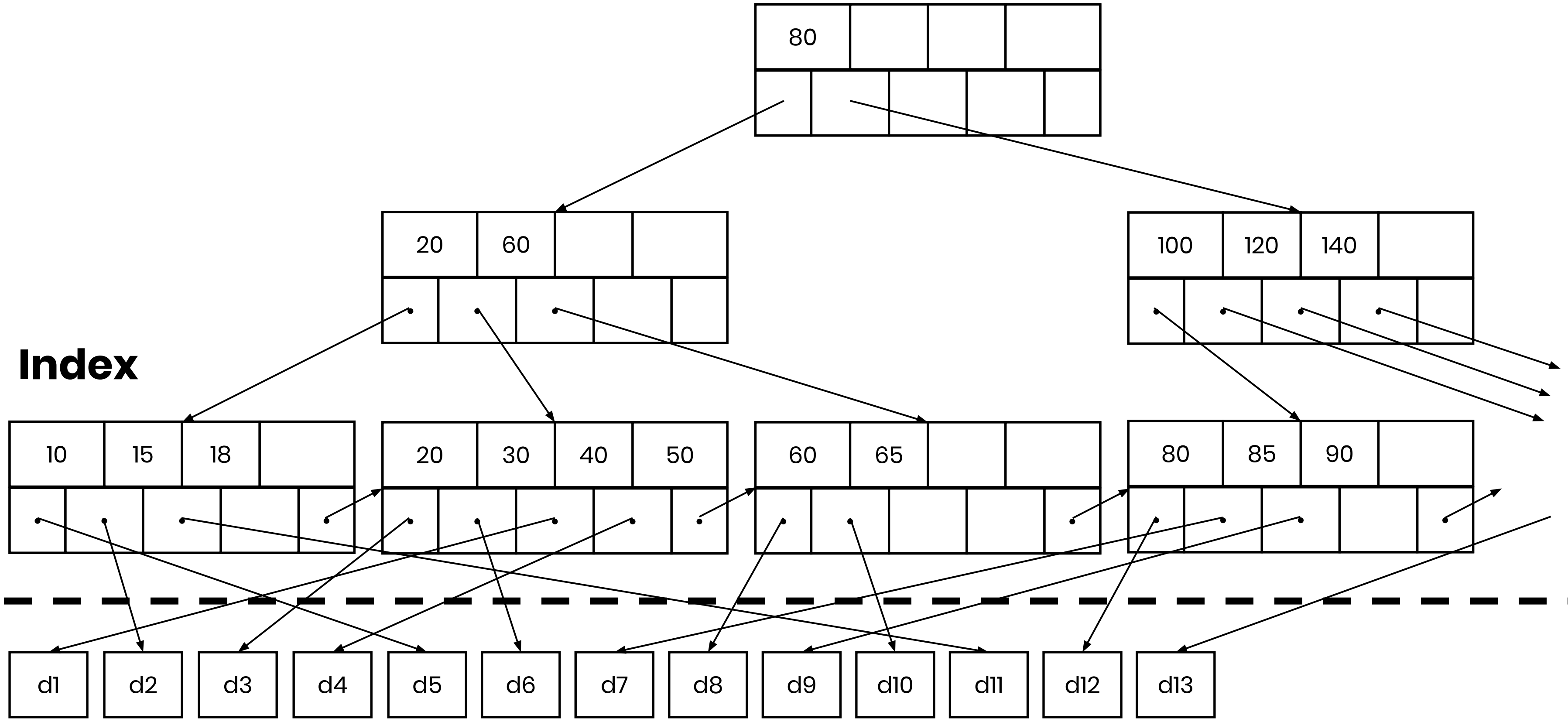
How is a B+ Tree Different?



Find the data associated with the key value 40
(same search process)

Clustered vs Unclustered Index

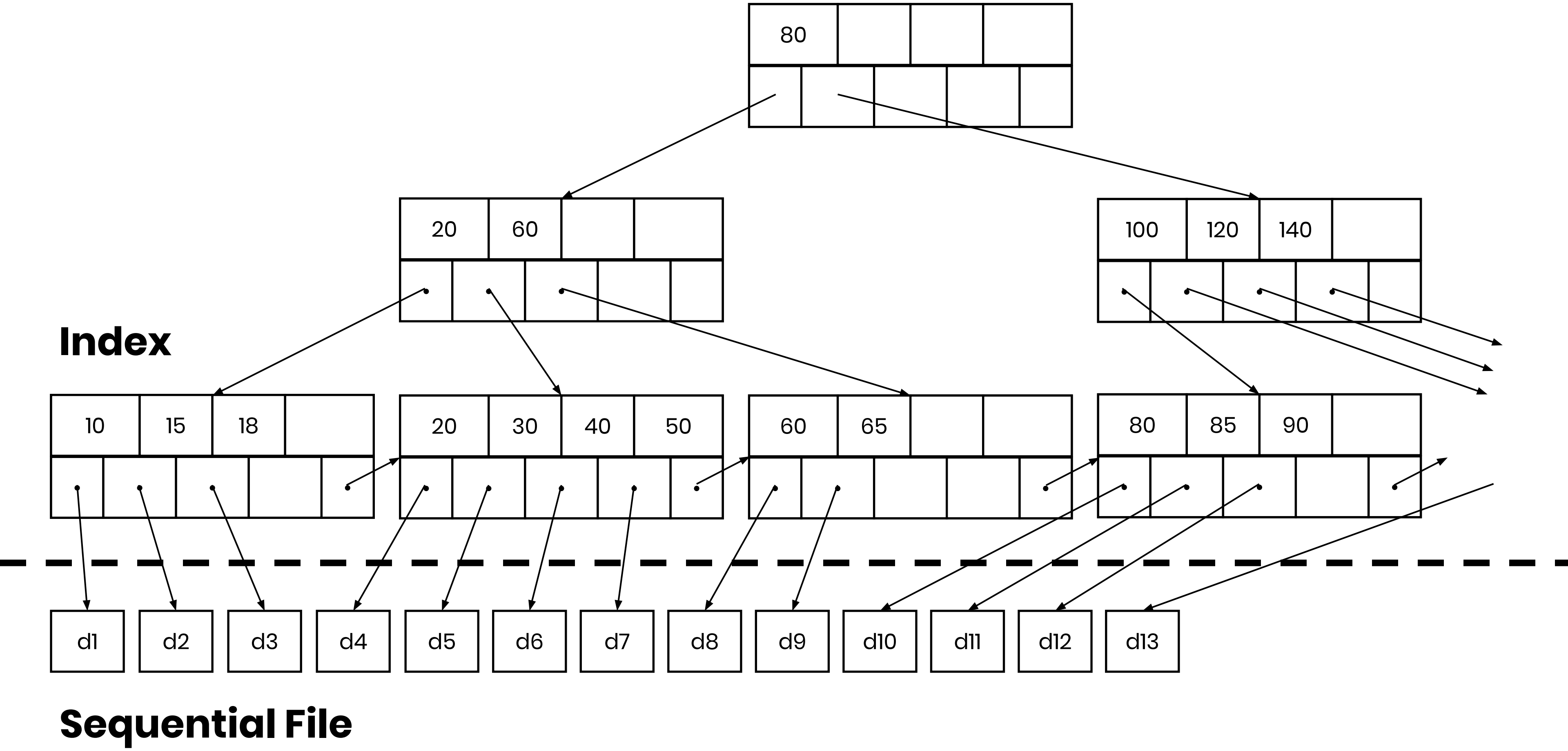
- An **unclustered index** may exist without any ordering on disk (any number per table)



Sequential File with a different key or **Heap File**

Clustered vs Unclustered Index

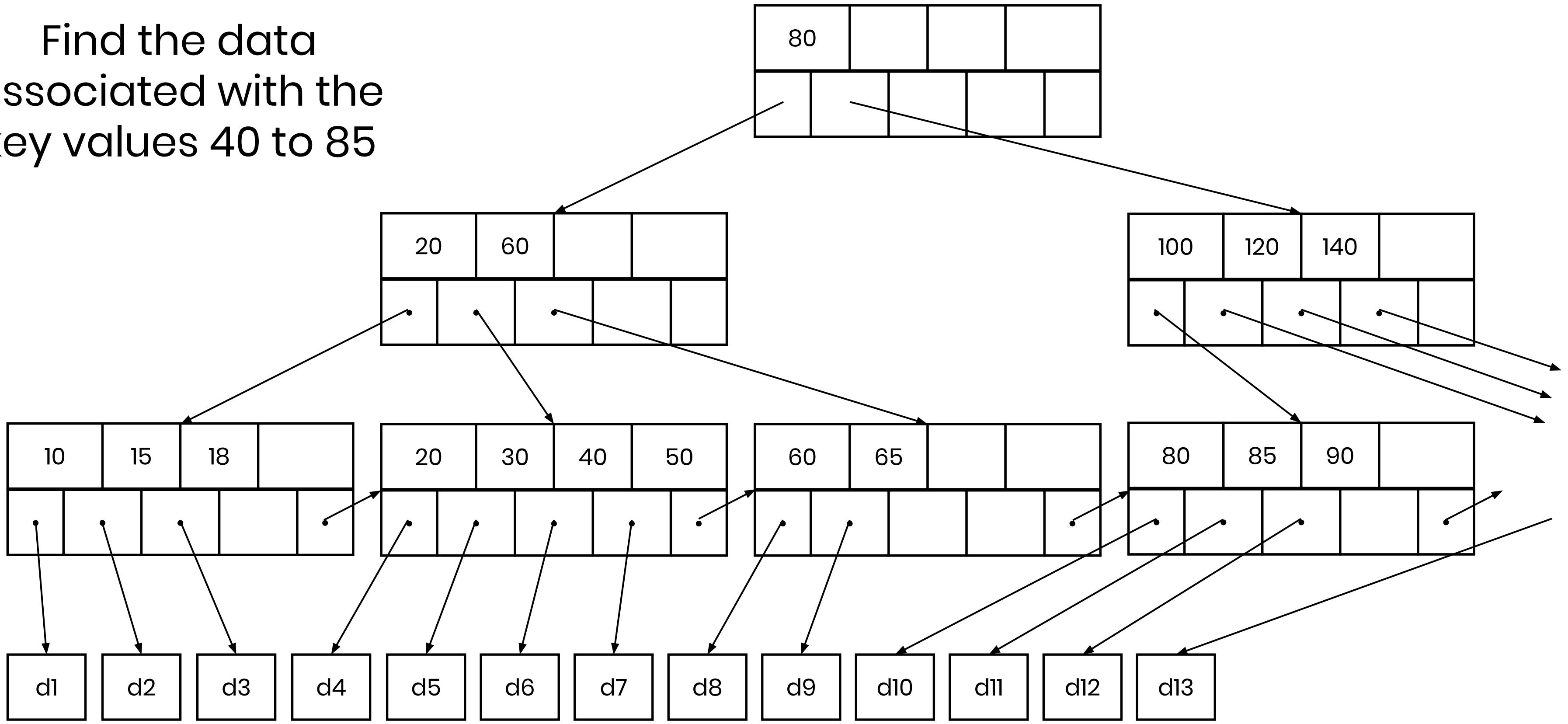
A **clustered index** is one that has the **same key ordering** as what is on disk (one per table)



Benefits of B+ Trees

- Range queries can be fast!
  - Filtering a value on a valid range is essentially looking up some portion of a B+ tree

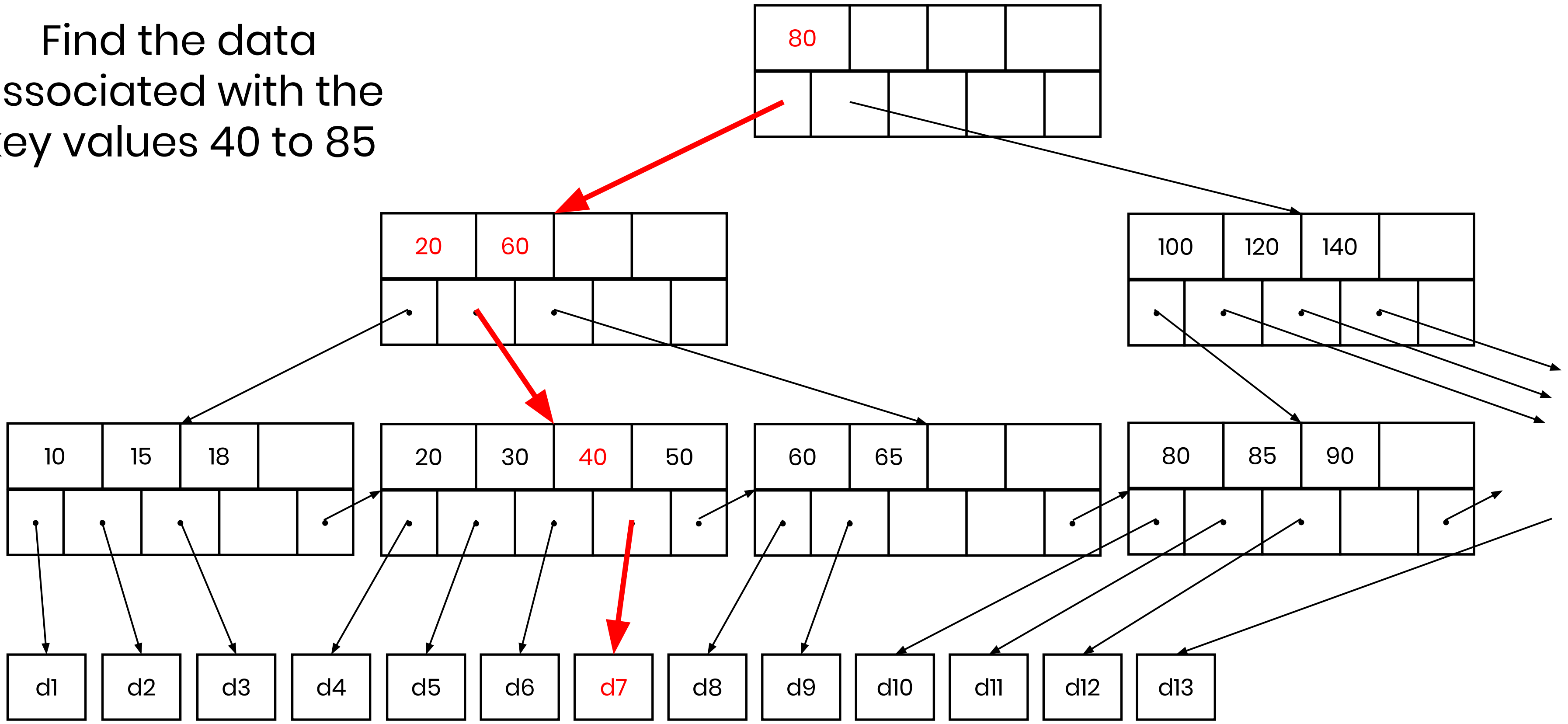
Find the data associated with the key values 40 to 85



Benefits of B+ Trees

- Range queries can be fast!
  - Filtering a value on a valid range is essentially looking up some portion of a B+ tree

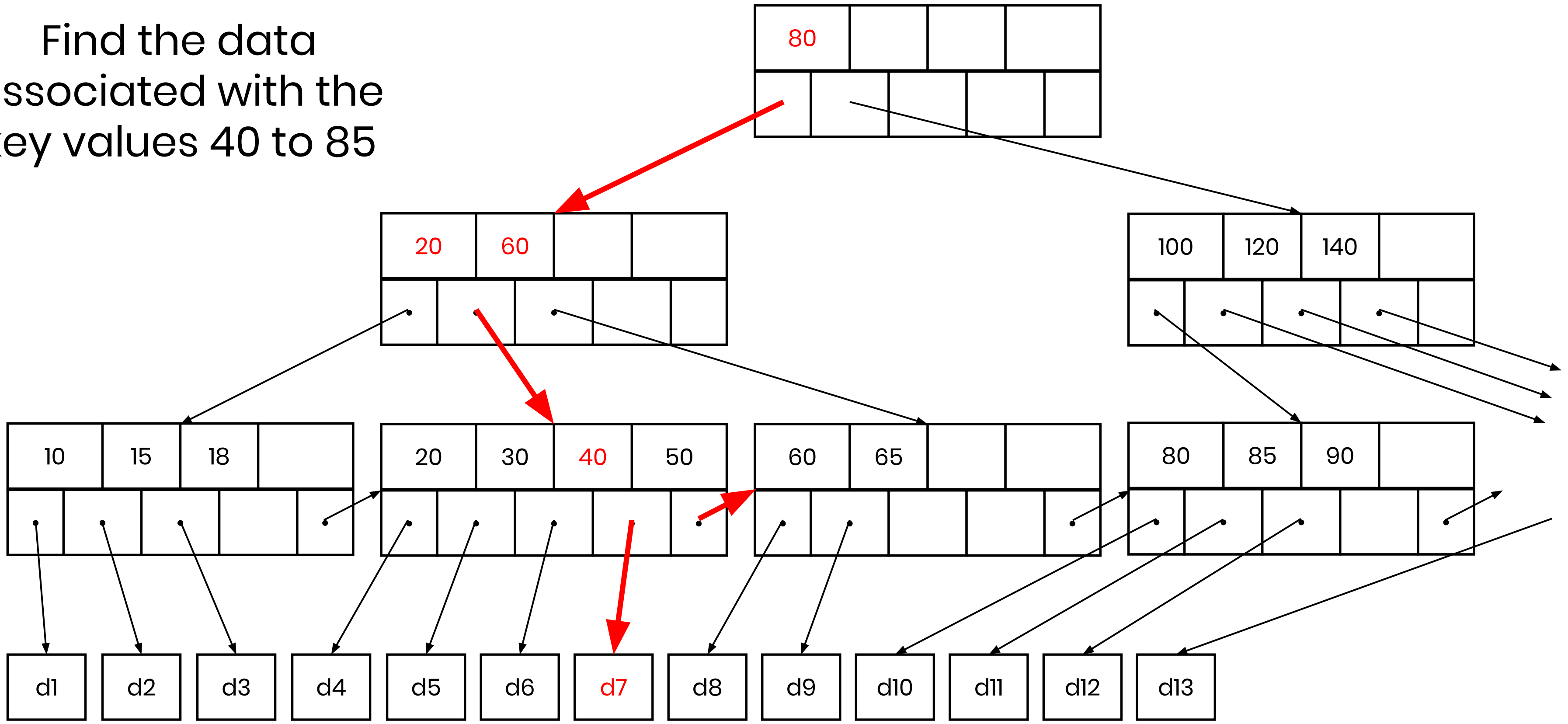
Find the data associated with the key values 40 to 85



Benefits of B+ Trees

- Range queries can be fast!
  - Filtering a value on a valid range is essentially looking up some portion of a B+ tree

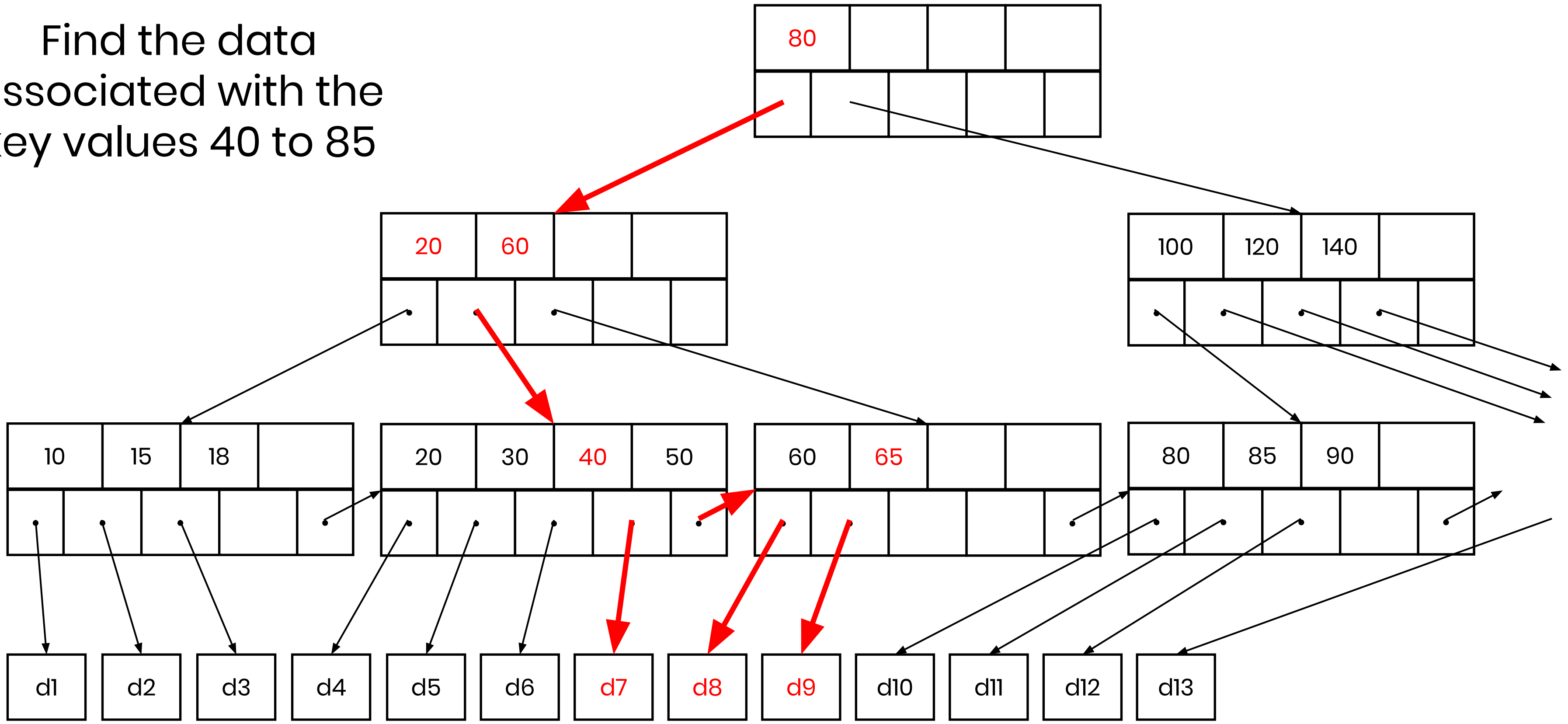
Find the data associated with the key values 40 to 85



Benefits of B+ Trees

- Range queries can be fast!
  - Filtering a value on a valid range is essentially looking up some portion of a B+ tree

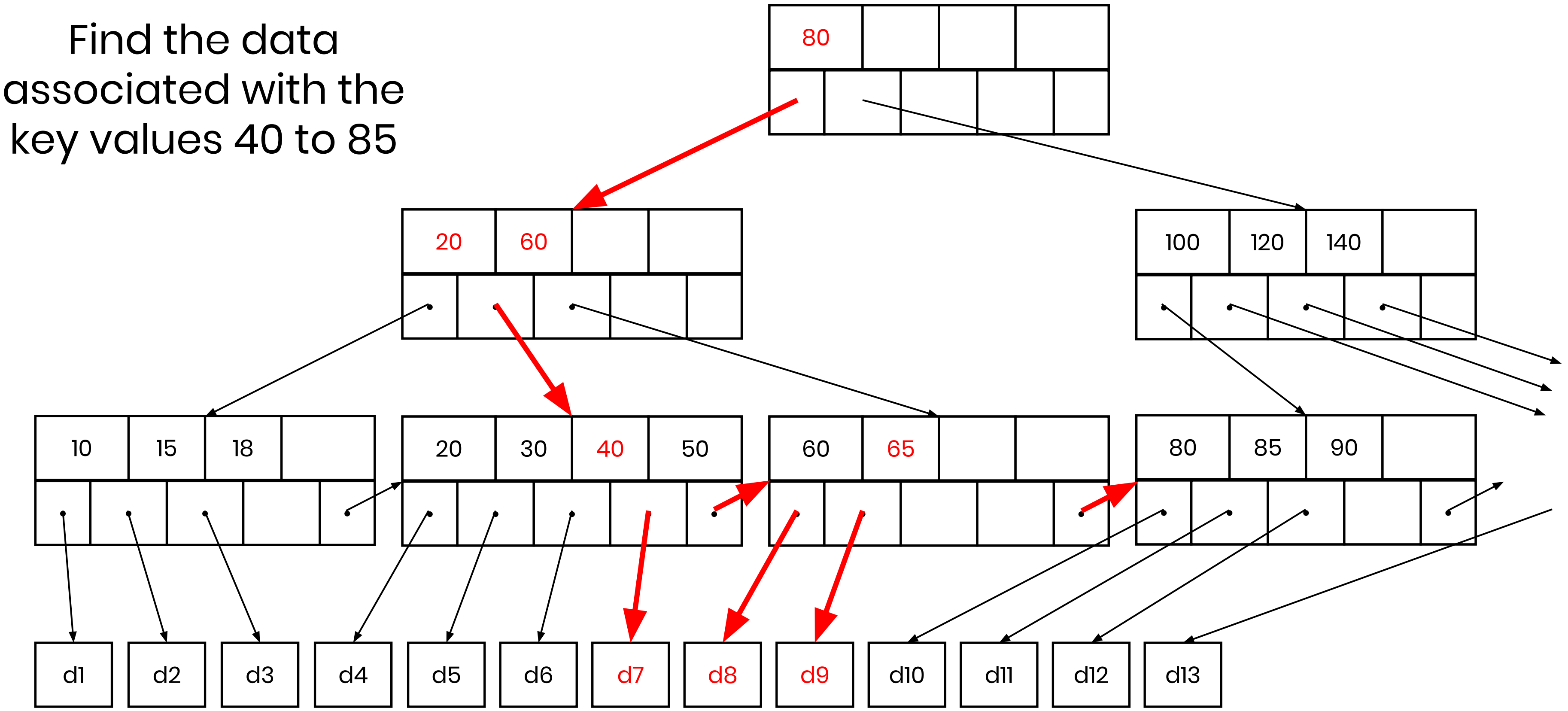
Find the data associated with the key values 40 to 85



Benefits of B+ Trees

- Range queries can be fast!
  - Filtering a value on a valid range is essentially looking up some portion of a B+ tree

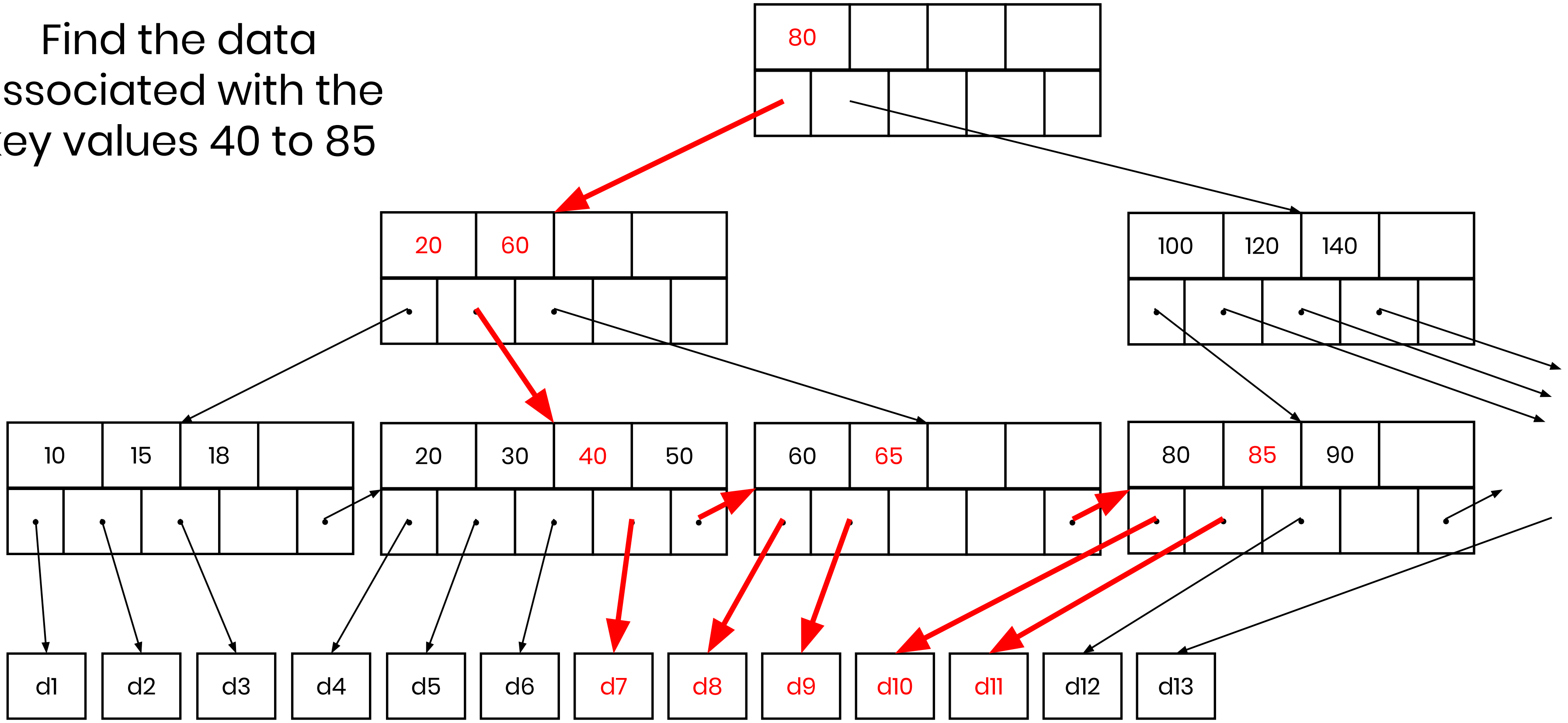
Find the data associated with the key values 40 to 85



Benefits of B+ Trees

- Range queries can be fast!
  - Filtering a value on a valid range is essentially looking up some portion of a B+ tree

Find the data associated with the key values 40 to 85



Estimating Amount of Data Read

- ***Selectivity Factor*** (X) → Proportion of total data needed
- Assuming uniform distribution of data values on numeric attribute a in table R, if the condition is:
  - a=c → $X \cong \frac{1}{V(a,R)}$
  - a<c → $X \cong \frac{c - \min(a,R)}{\max(a,R) - \min(a,R)}$
  - c1<a<c2 → $X \cong \frac{c2 - c1}{\max(a,R) - \min(a,R)}$
  - cond1 AND cond2 → $X \cong X_1 * X_2$
- Disclaimer: More thorough selectivity estimation will use a histogram

Estimating Amount of Data Read

- ***Selectivity Factor*** (X) → Proportion of total data needed
- Assuming uniform distribution of data values on numeric attribute a in table R, if the condition is:
  - a=c → $X \cong \frac{1}{V(a,R)}$
  - a<c → $X \cong \frac{c - \min(a,R)}{\max(a,R) - \min(a,R)}$
  - c1<a<c2 → $X \cong \frac{c2 - c1}{\max(a,R) - \min(a,R)}$
  - cond1 AND cond2 → $X \cong X_1 * X_2$
- Disclaimer: More thorough selectivity estimation will use a histogram

a = 4
[1, 1, 2, 2, 3, 3, 4, 4, 5, 5]

X = 1 / 5

Estimating Amount of Data Read

- ***Selectivity Factor*** (X) → Proportion of total data needed
- Assuming uniform distribution of data values on numeric attribute a in table R, if the condition is:
  - a=c → $X \cong \frac{1}{V(a,R)}$
  - a<c → $X \cong \frac{c - \min(a,R)}{\max(a,R) - \min(a,R)}$
  - c1<a<c2 → $X \cong \frac{c2 - c1}{\max(a,R) - \min(a,R)}$
  - cond1 AND cond2 → $X \cong X_1 * X_2$

a < 4

[1, 1, 2, 2, 3, 3, 4, 4, 5, 5]

X = (4 - 1) / (5 - 1)

- Disclaimer: More thorough selectivity estimation will use a histogram

Estimating Amount of Data Read

- ***Selectivity Factor*** (X) → Proportion of total data needed
- Assuming uniform distribution of data values on numeric attribute a in table R, if the condition is:
  - a=c → $X \cong \frac{1}{V(a,R)}$
  - a<c → $X \cong \frac{c - \min(a,R)}{\max(a,R) - \min(a,R)}$
  - c1<a<c2 → $X \cong \frac{c2 - c1}{\max(a,R) - \min(a,R)}$
  - cond1 AND cond2 → $X \cong X_1 * X_2$

2 < a < 4

[1, 1, 2, 2, 3, 3, 4, 4, 5, 5]

X = (4 - 2) / (5 - 1)

- Disclaimer: More thorough selectivity estimation will use a histogram

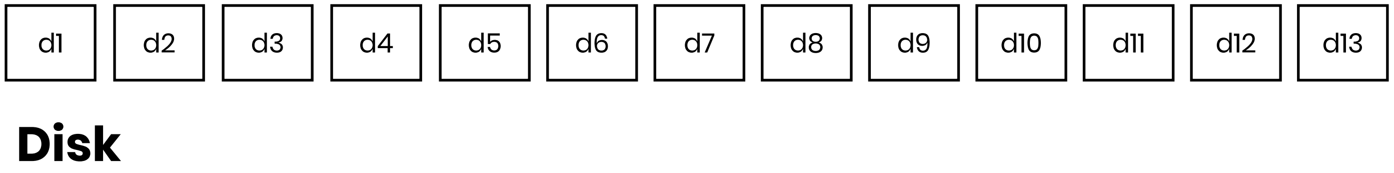
Index-Based Selection

- For reference, a full sequential scan of data costs B(R) IOs
- Provided some condition to read data:
  - Full **sequential scan □ B(R)**
  - Scan on **clustered index □ X*B(R)**
    - Able to read a contiguous chunk of the file
  - Scan on **unclustered index □ X*T(R)**
    - Worst case would read a different block every time

Sequential Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

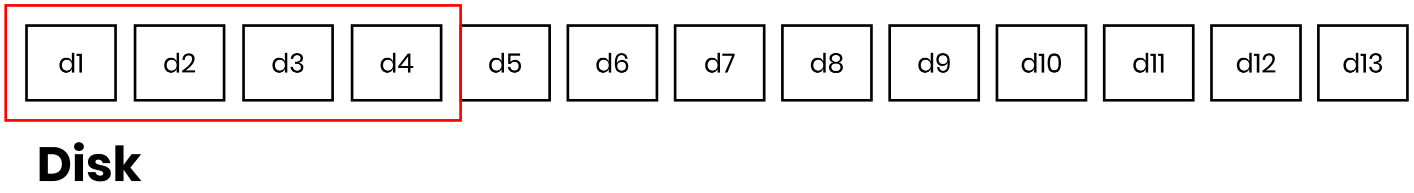
Without an index, finding a value must be done the “old fashioned way”



Sequential Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

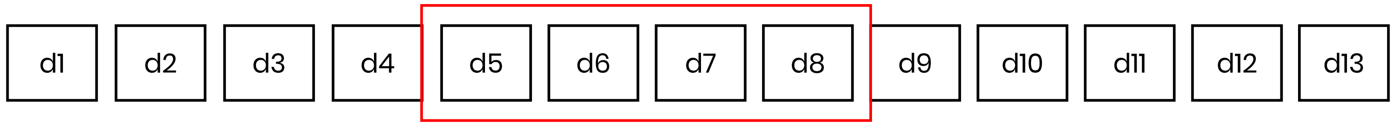
Without an index, finding a value must be done the "old fashioned way"



Sequential Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

Without an index, finding a value must be done the “old fashioned way”

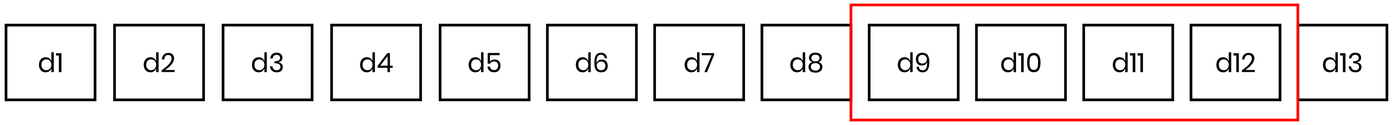


Disk

Sequential Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

Without an index, finding a value must be done the "old fashioned way"

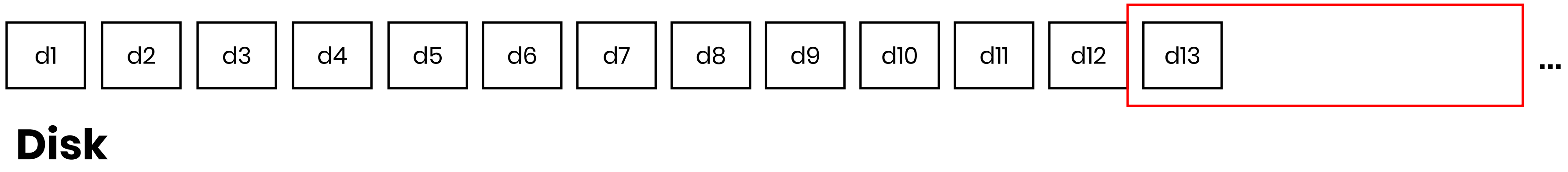


Disk

Sequential Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

Without an index, finding a value must be done the "old fashioned way"

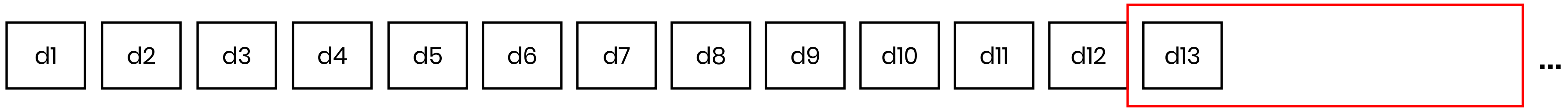


Sequential Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

Without an index, finding a value must be done the “old fashioned way”

Total cost: B(R)

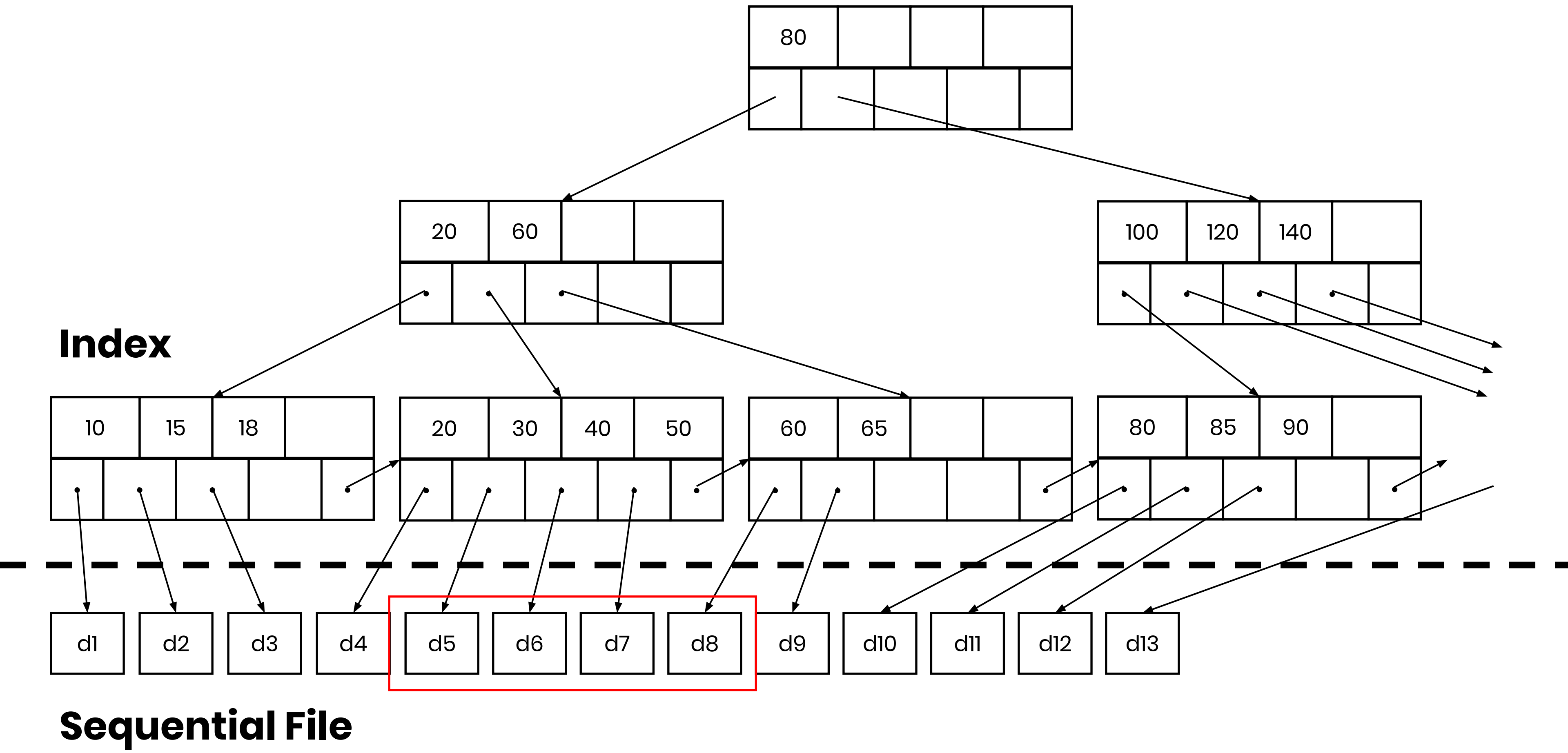


Disk

Clustered Index Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

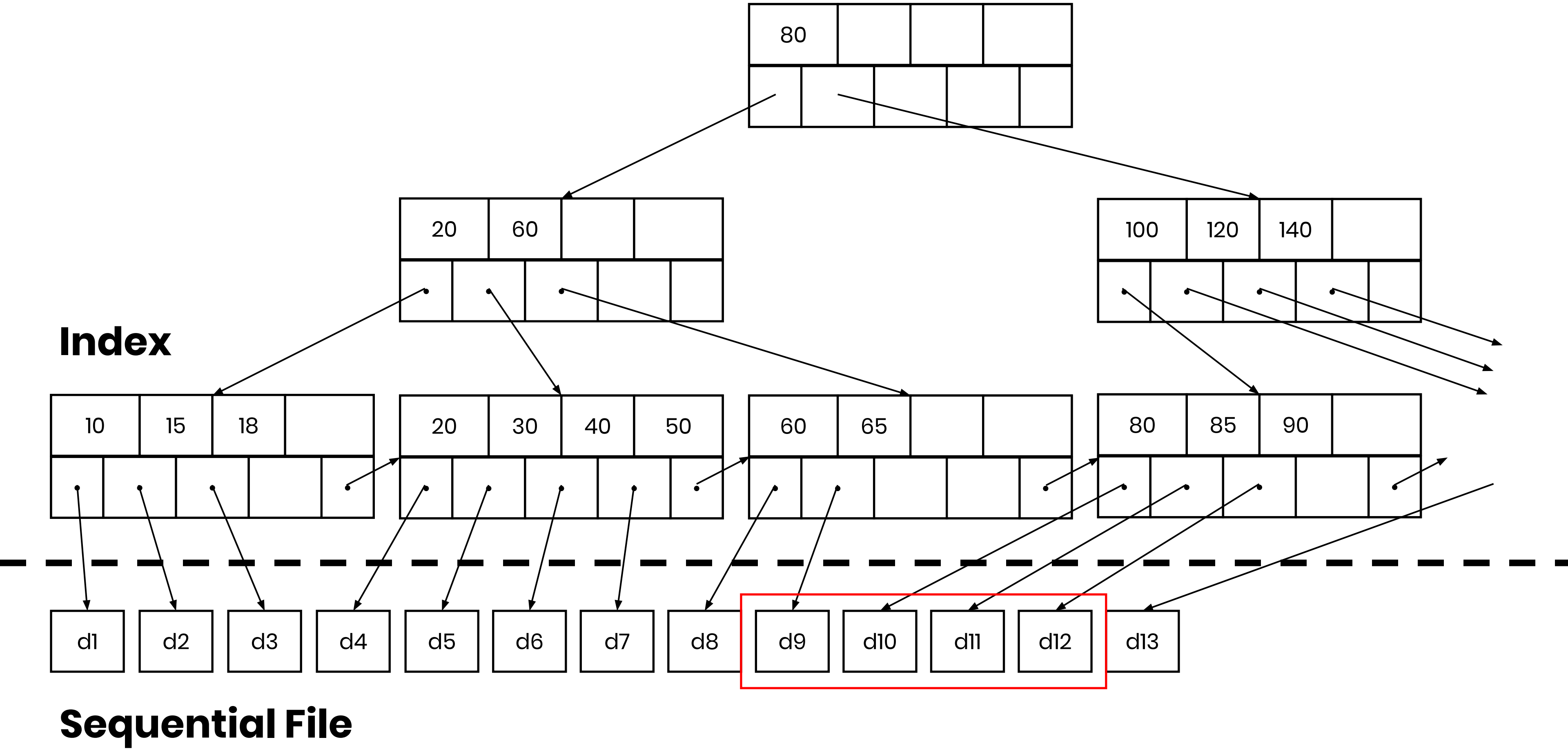
With a clustered index, I start scanning blocks in the range they are at



Clustered Index Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

With a clustered index, I start scanning blocks in the range they are at

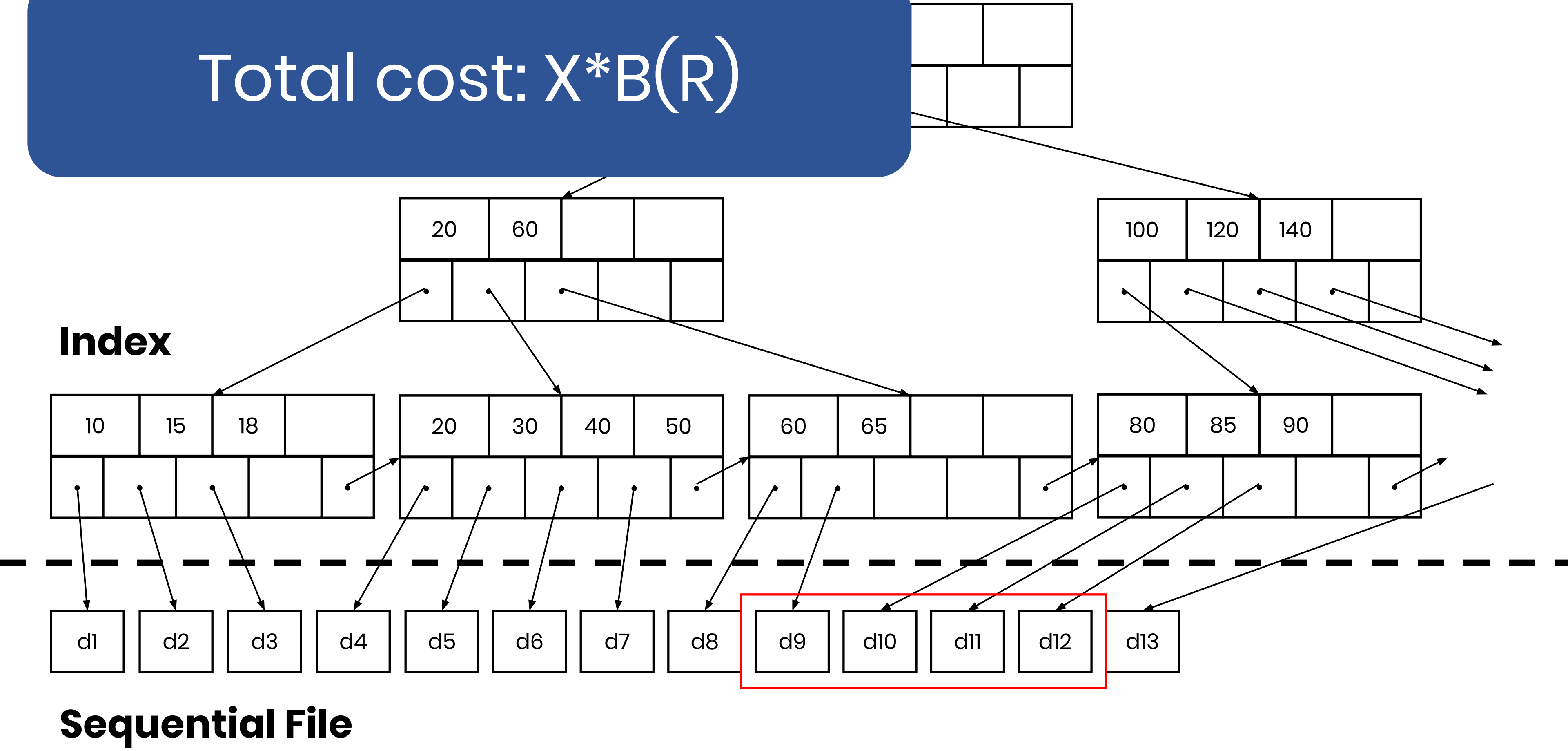


Clustered Index Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

With a clustered index, I start scanning blocks in the range they are at

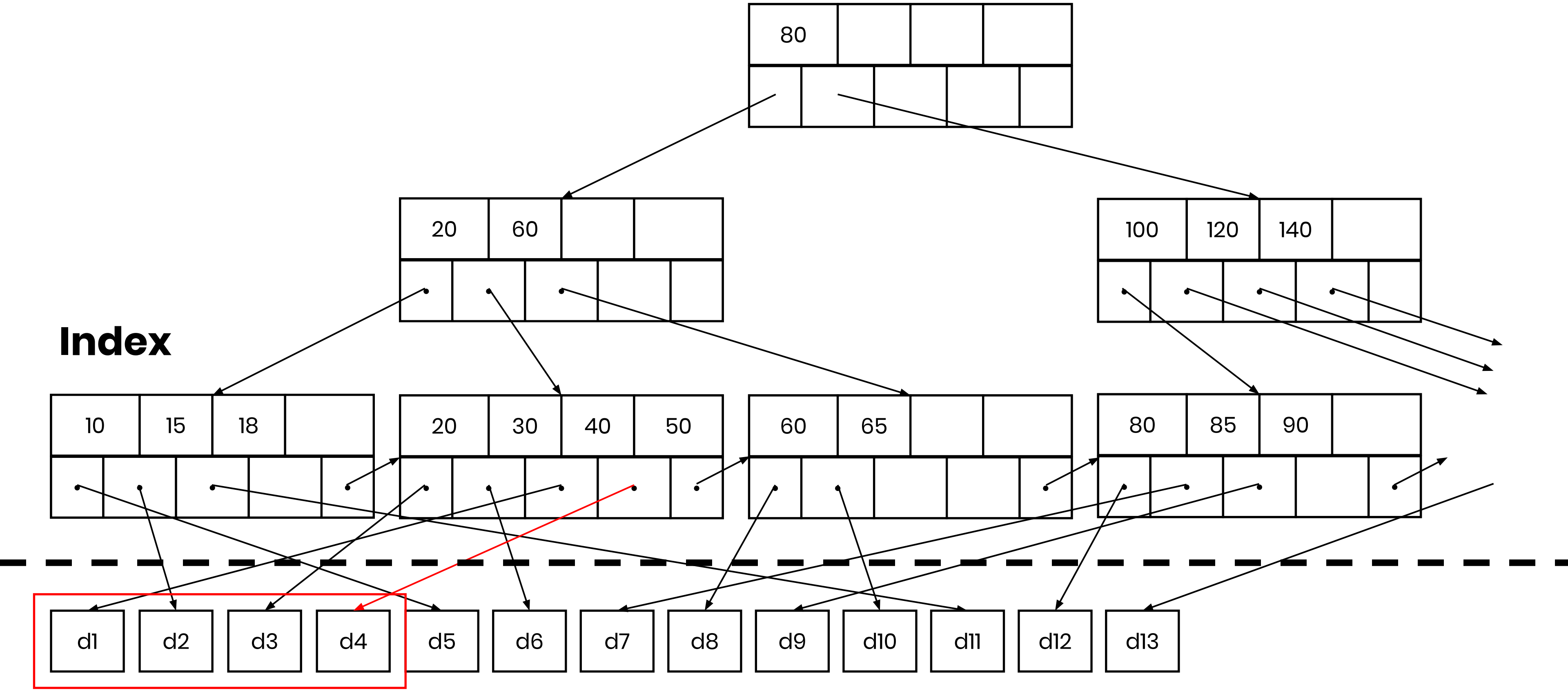
Total cost: X*B(R)



Unclustered Index Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

With an unclustered index, I scan tuples wherever they occur

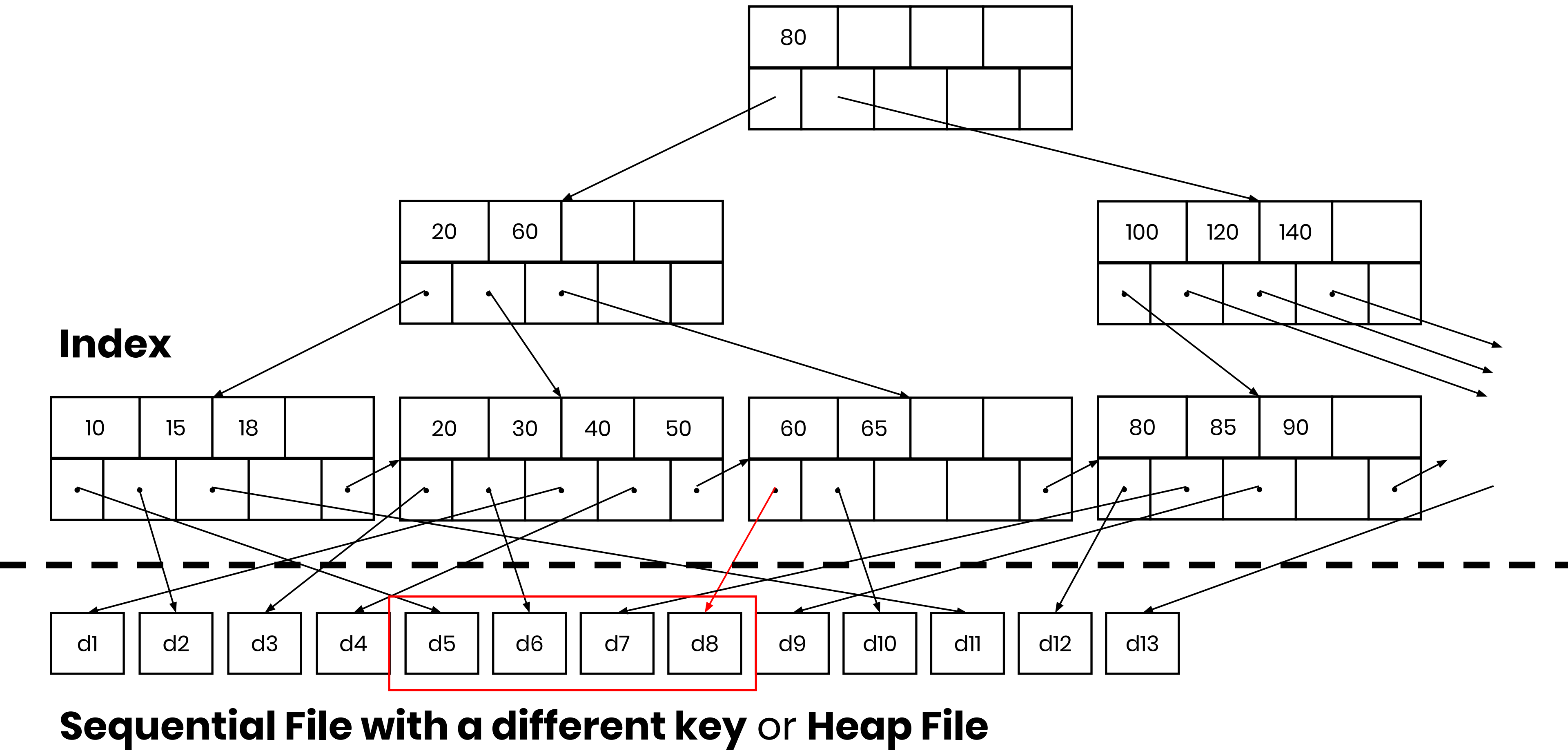


Sequential File with a different key or **Heap File**

Unclustered Index Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

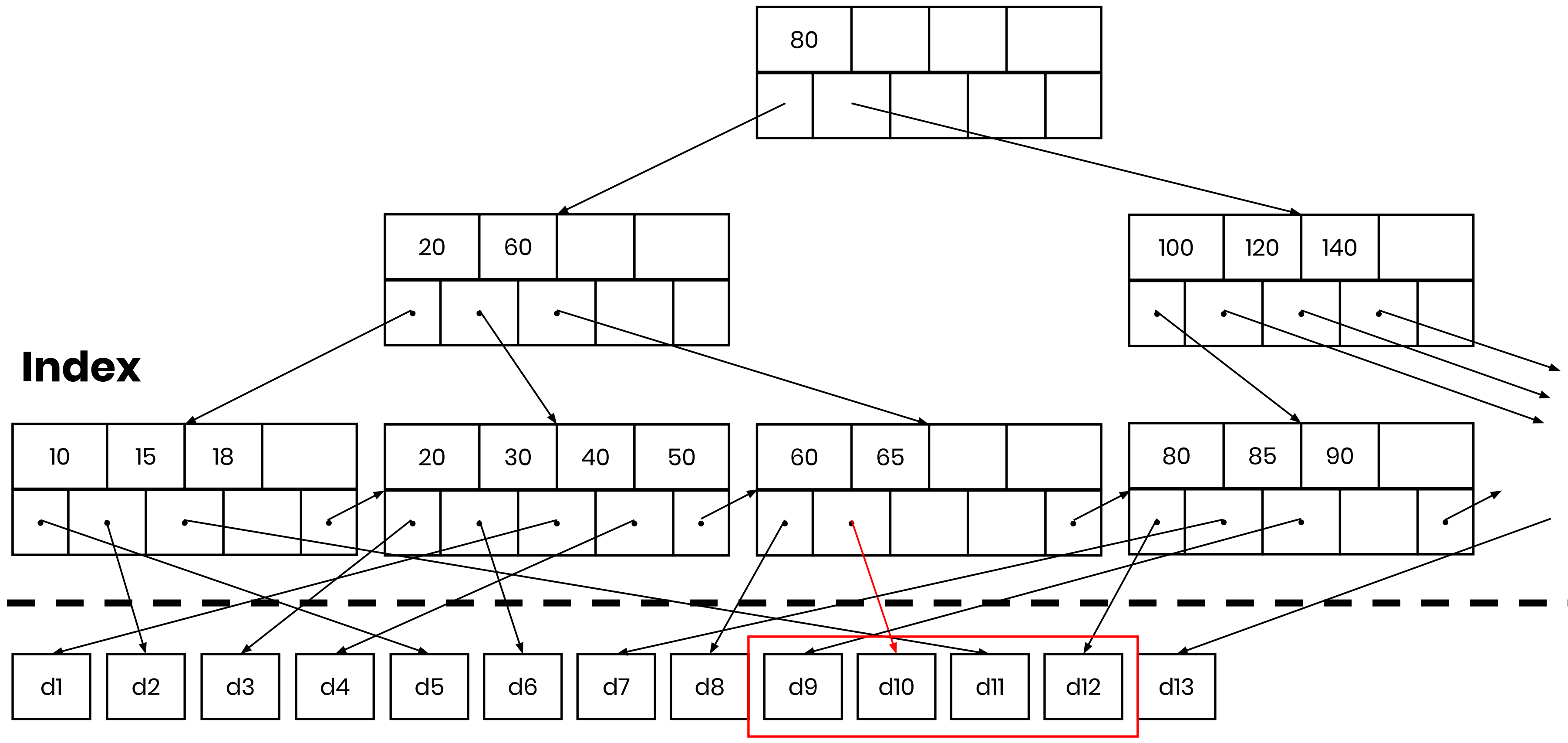
With an unclustered index, I scan tuples wherever they occur



Unclustered Index Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

With an unclustered index, I scan tuples wherever they occur

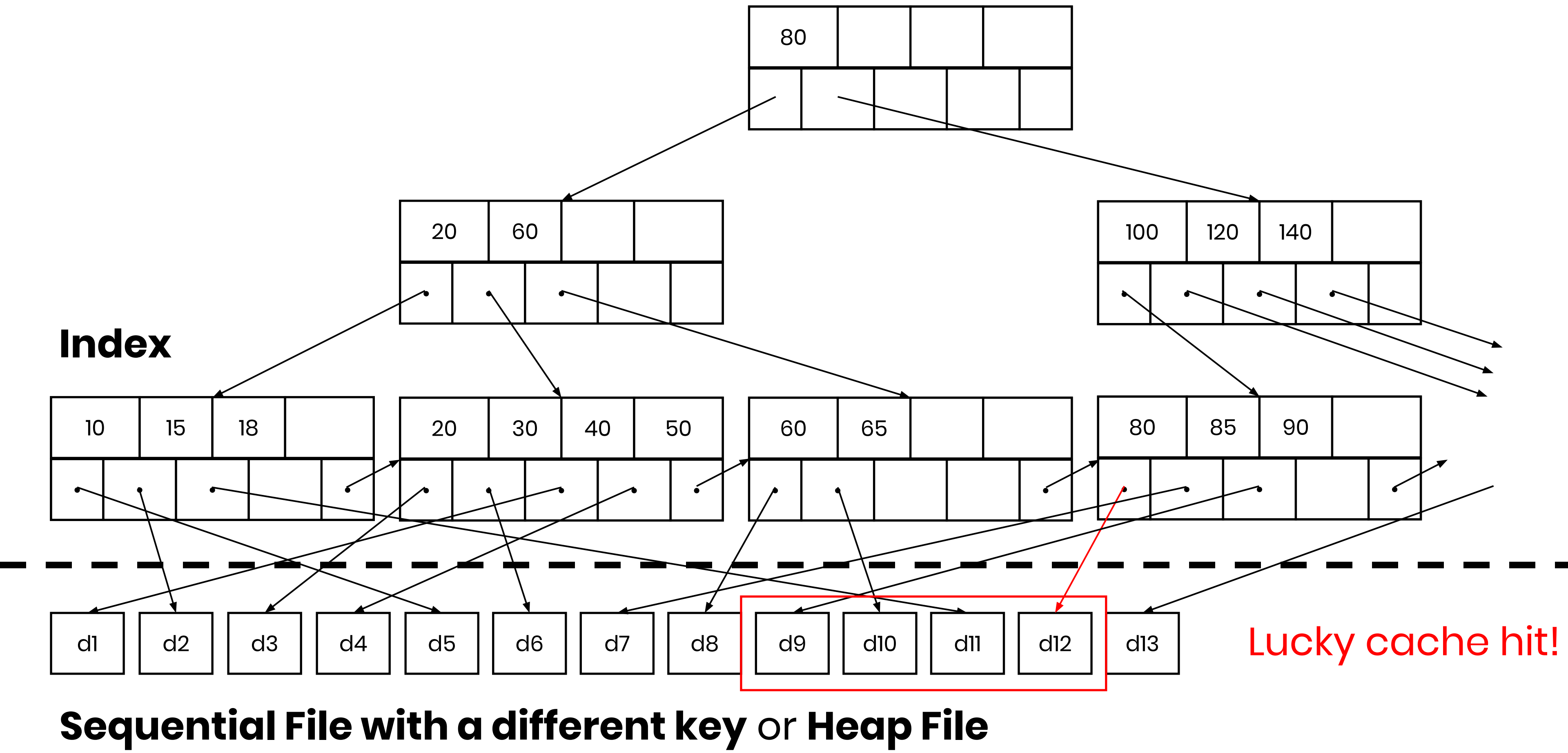


Sequential File with a different key or **Heap File**

Unclustered Index Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

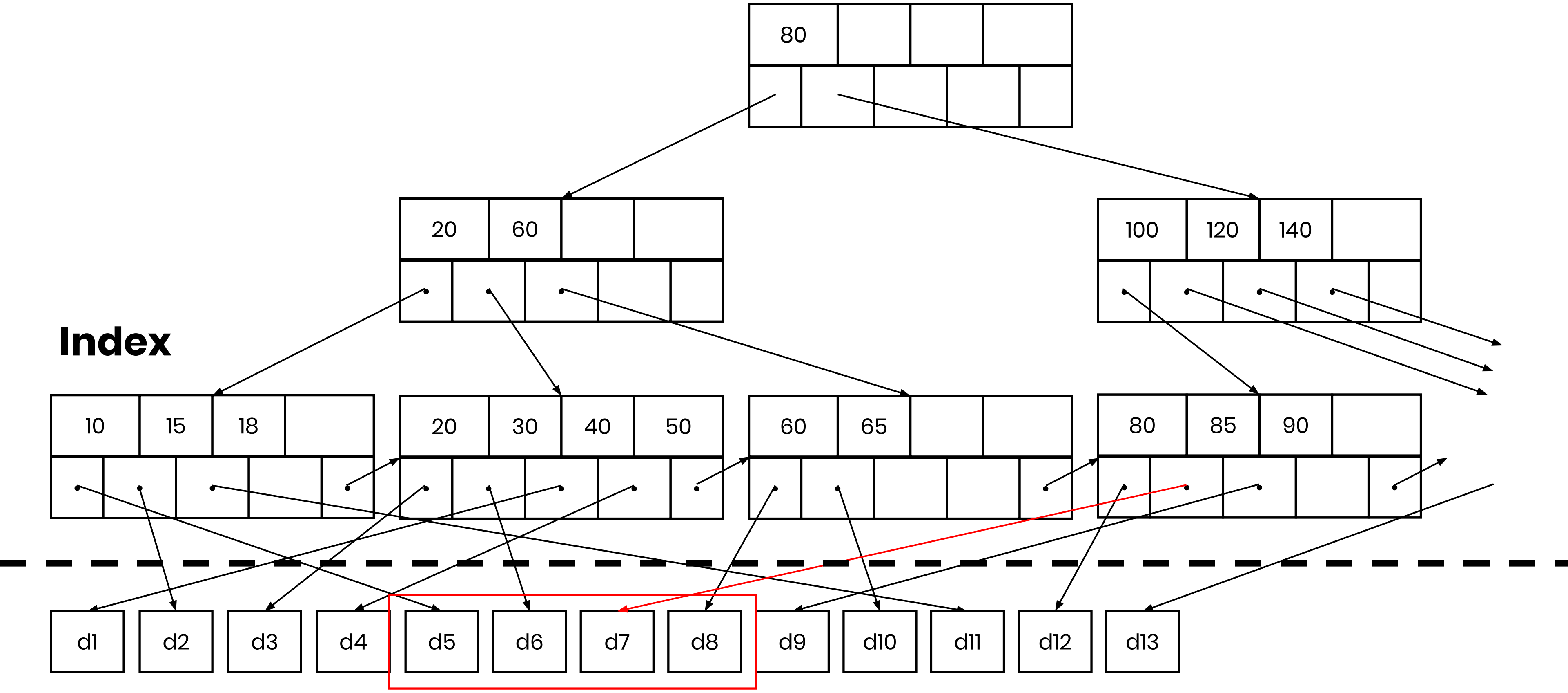
With an unclustered index, I scan tuples wherever they occur



Unclustered Index Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

With an unclustered index, I scan tuples wherever they occur



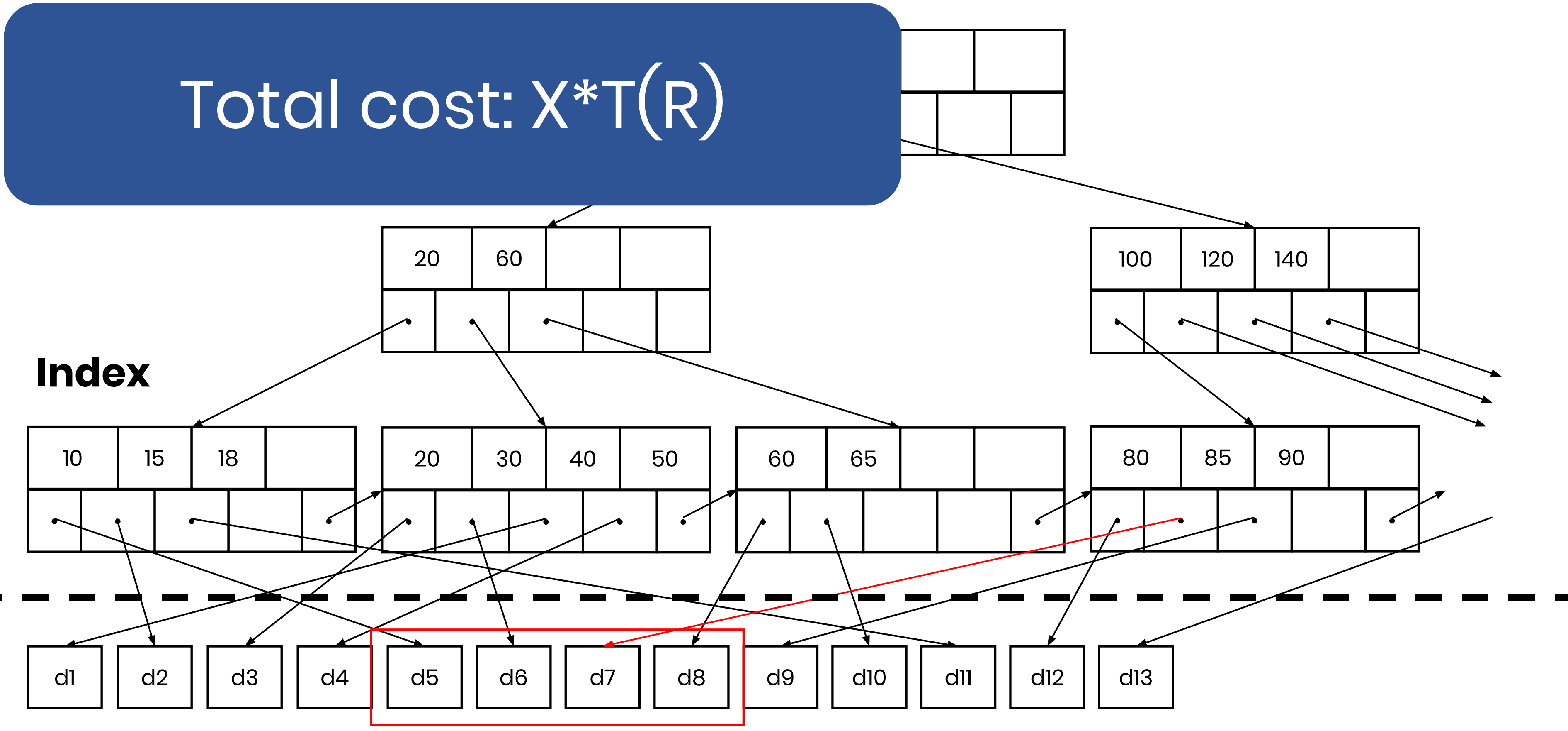
Sequential File with a different key or **Heap File**

Unclustered Index Scan

Assume a block holds 4 tuples. I want tuples associated with values 40-85.

With an unclustered index, I scan tuples wherever they occur

Total cost: X*T(R)



Sequential File with a different key or **Heap File**

Index Expectations

- Using an index in the wrong scenario can lead to a slowdown!
- Common example:

  Full sequential scan vs unclustered index scan with **high X value** and/or **small tuple size** (large T(R):B(R) ratio)

Known:
B(R) = 100
T(R) = 10000

Consider a query with X=1/10

Sequential scan
= B(R)
= **100**

Index scan
= X*T(R)
= 1/10 * 10000
= **1000**

Index Expectations

- Using an index in the wrong scenario can lead to a slowdown!
- Common example:

  Full sequential scan vs unclustered index scan with **high X value** and/or **small tuple size** (large T(R):B(R) ratio)

Having indexes doesn't mean you will see a speedup!

Known:
B(R) = 100
T(R) = 10000

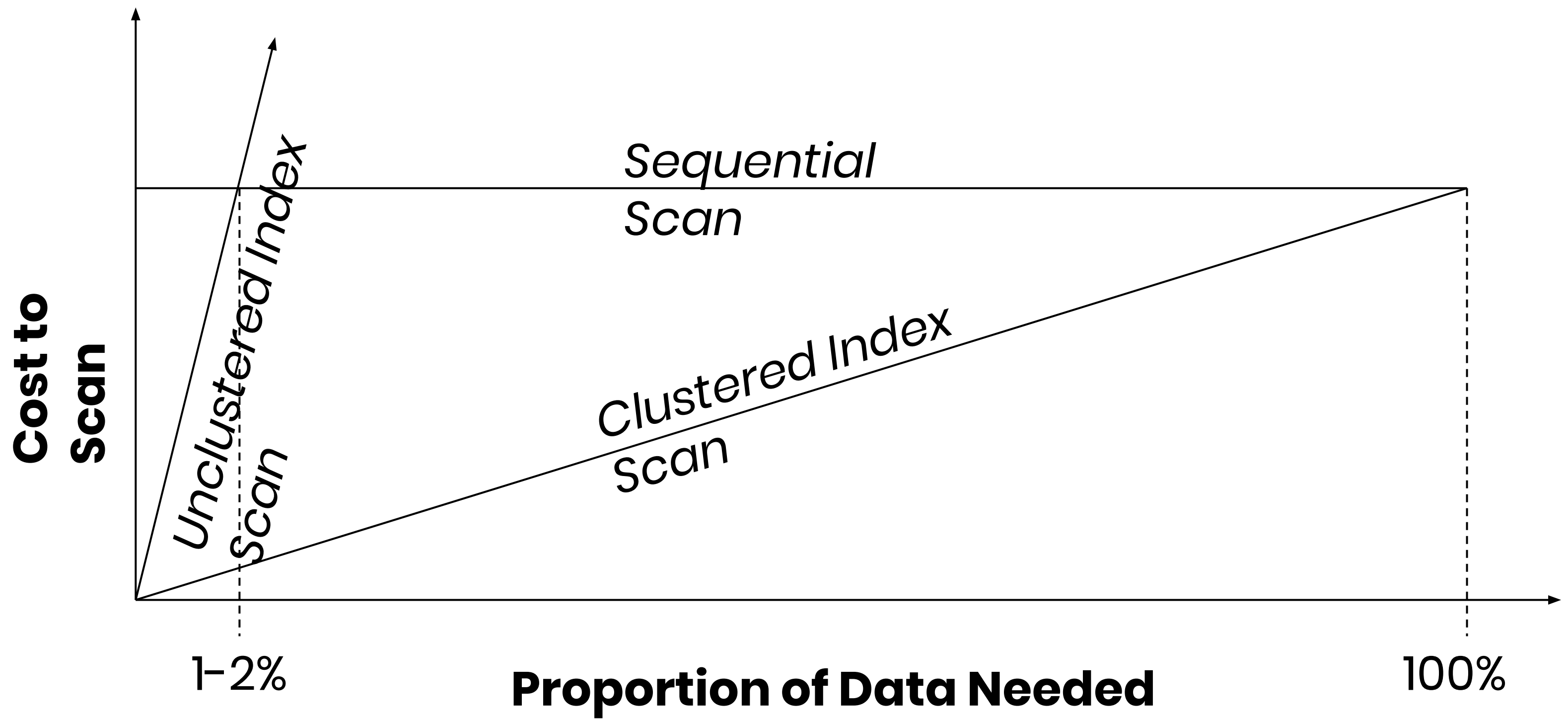
Consider a query with X=1/10

Sequential scan
= B(R)
= **100**

Index scan
= X*T(R)
= 1/10 * 10000
= **1000**

Index Expectations

- **Sequential disk reads are faster than random ones**
  - Cost ~1–2% random scan = full sequential scan


Cost to Scan
Unclustered Index Scan
Sequential Scan
Clustered Index Scan
Proportion of Data Needed