CSE544
Data Management

Lectures 7

Storage Manager

Announcements

- Project teams due Friday
- HW2 is posted
- Review of PAX due on Monday

Relational Algebra

Five operators:

- Selection σ
- Projection Π
- Join or cartesian product $\bowtie$, $\times$
- Union $\cup$
- Difference $-$

Which operations are monotone?

Monotone

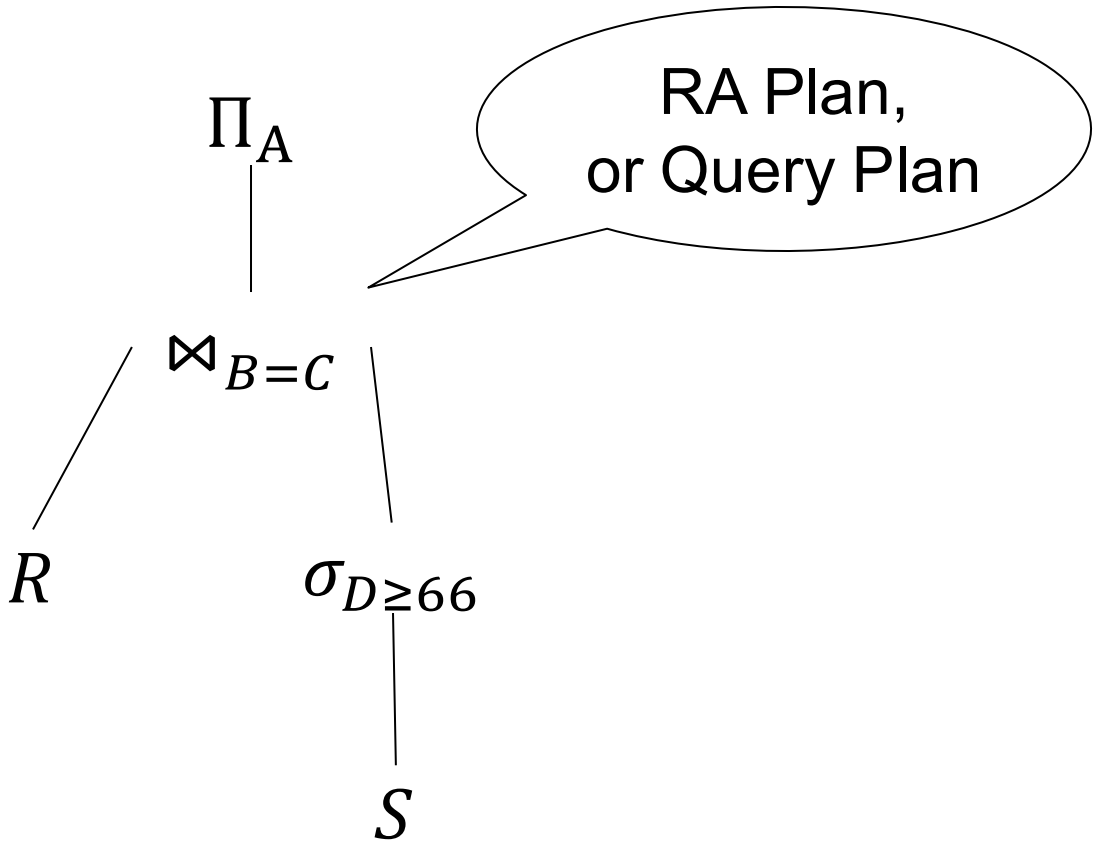
Non-monotone

RA by Example

$$\Pi_{\mathrm{A}}(R \bowtie_{B=C} \sigma_{D \geq 66}(S))$$

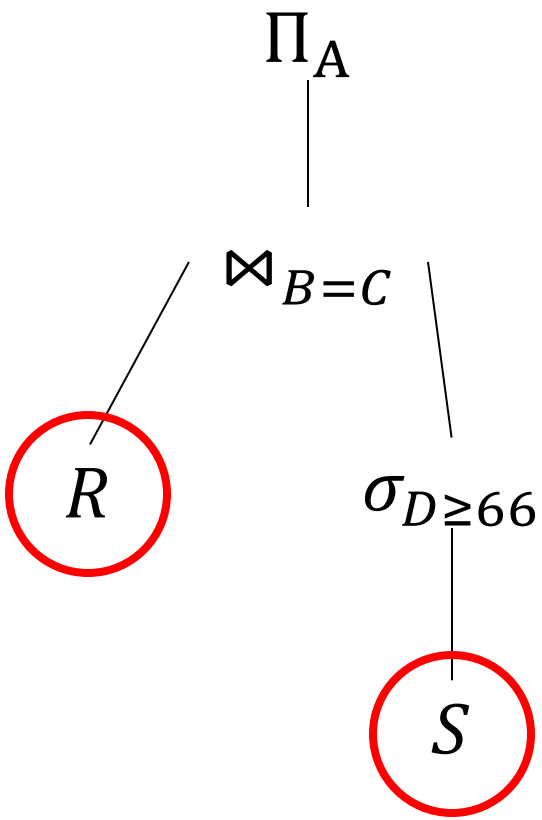
RA by Example

$$\Pi_A(R \bowtie_{B=C} \sigma_{D \geq 66}(S))$$



RA by Example

$$\Pi_A(R \bowtie_{B=C} \sigma_{D \geq 66}(S))$$



R=

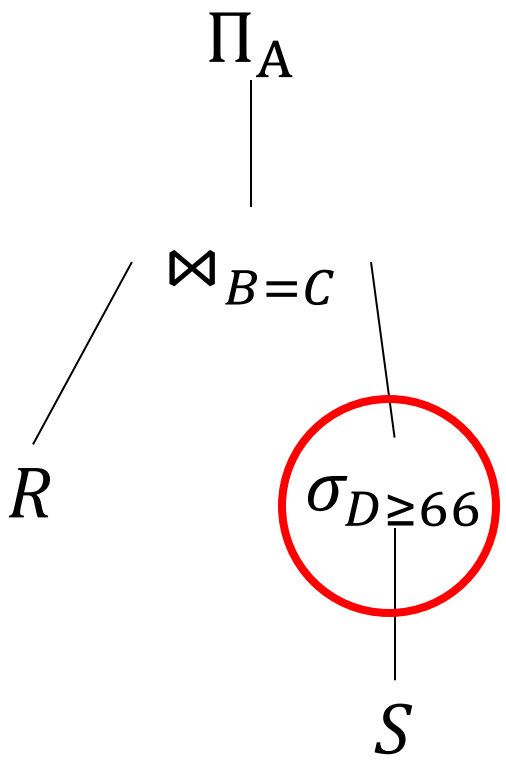
A	B
1	10
1	20
2	20

S=

C	D
10	33
10	44
20	55
20	66
20	77
30	66

RA by Example

$$\Pi_A(R \bowtie_{B=C} \sigma_{D \geq 66}(S))$$



R=

A	B
1	10
1	20
2	20

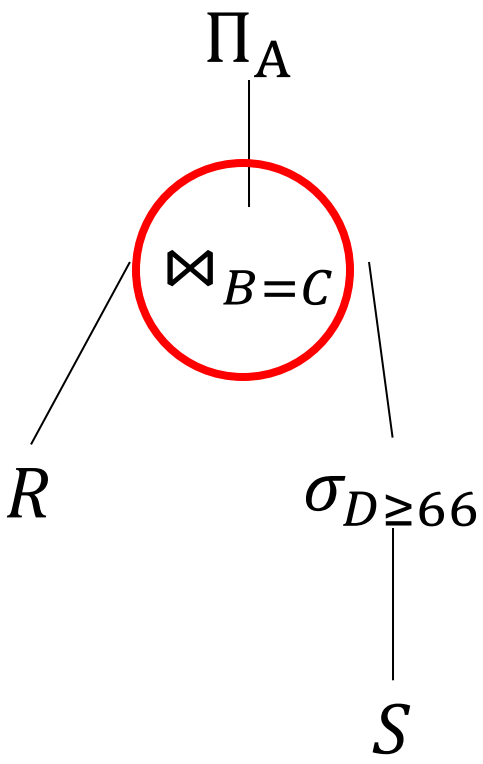
S=

C	D
10	33
10	44
20	55
20	66
20	77
30	66

C	D
20	66
20	77
30	66

RA by Example

$$\Pi_A(R \bowtie_{B=C} \sigma_{D \geq 66}(S))$$



A	B	C	D
1	20	20	66
1	20	20	77
2	20	20	66
2	20	20	77

C	D
20	66
20	77
30	66

R=

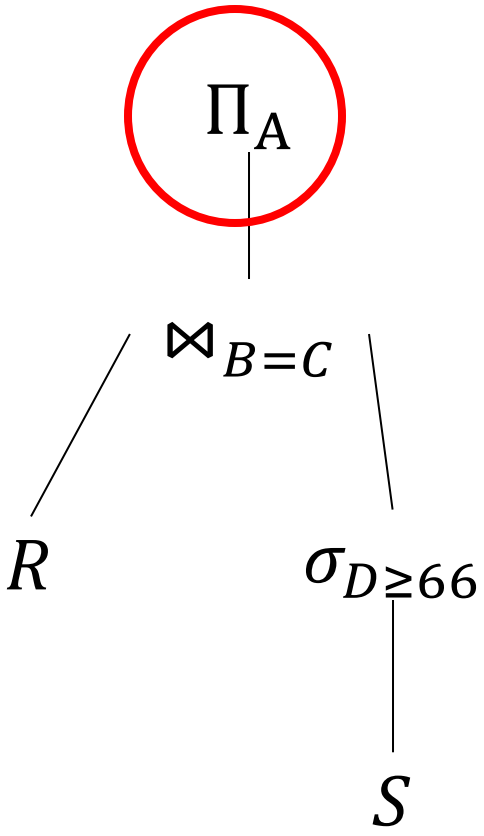
A	B
1	10
1	20
2	20

S=

C	D
10	33
10	44
20	55
20	66
20	77
30	66

RA by Example

$$\Pi_A(R \bowtie_{B=C} \sigma_{D \geq 66}(S))$$



A
1
2

Final output

A	B	C	D
1	20	20	66
1	20	20	77
2	20	20	66
2	20	20	77

C	D
20	66
20	77
30	66

R=

A	B
1	10
1	20
2	20

S=

C	D
10	33
10	44
20	55
20	66
20	77
30	66

Relational Algebra

Five operators:

- Selection σ
- Projection Π
- Join or cartesian product $\bowtie$, $\times$
- Union $\cup$
- Difference $-$

Extended Relational Algebra

- Group-by and aggregate: γ

- Duplicate elimination: δ
- Sorting: τ

Group-by and Aggregates

$\gamma_{col1,col2,\ldots,agg1,\ldots}(T)$

Standard group-by:

```
SELECT col1,…,agg1(..),agg2(..)
FROM T
GROUP-BY condition;
```

$\gamma_{C,sum(D)}(S) =$

C	D
10	77
20	196
30	66

S=

C	D
10	33
10	44
20	55
20	66
20	77
30	66

Translation

Every SQL query can be translated into an expression in the Extended RA

Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

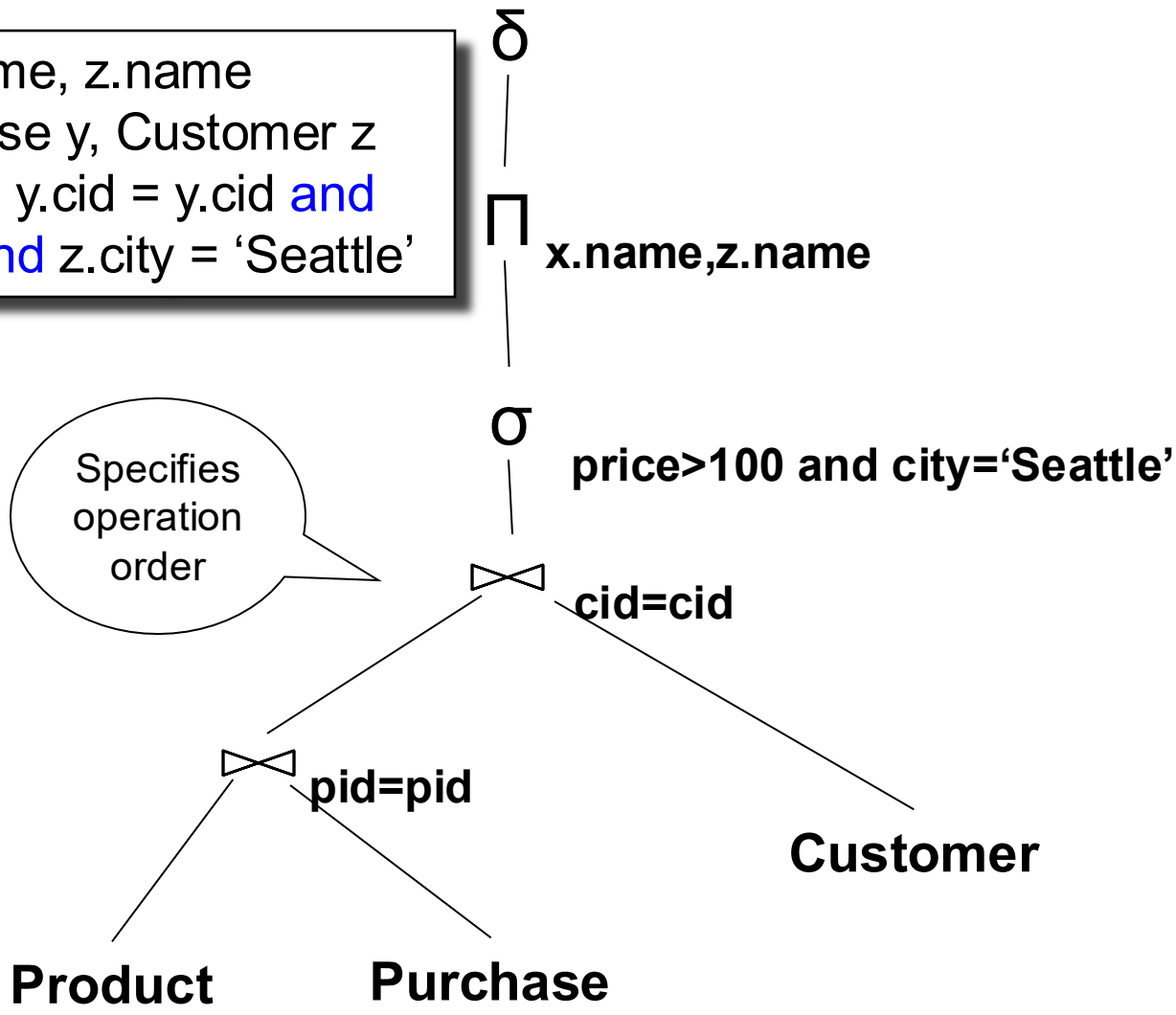
SQL…

```
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = y.cid and
      x.price > 100 and z.city = 'Seattle'
```

```
Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)
```

…to RA

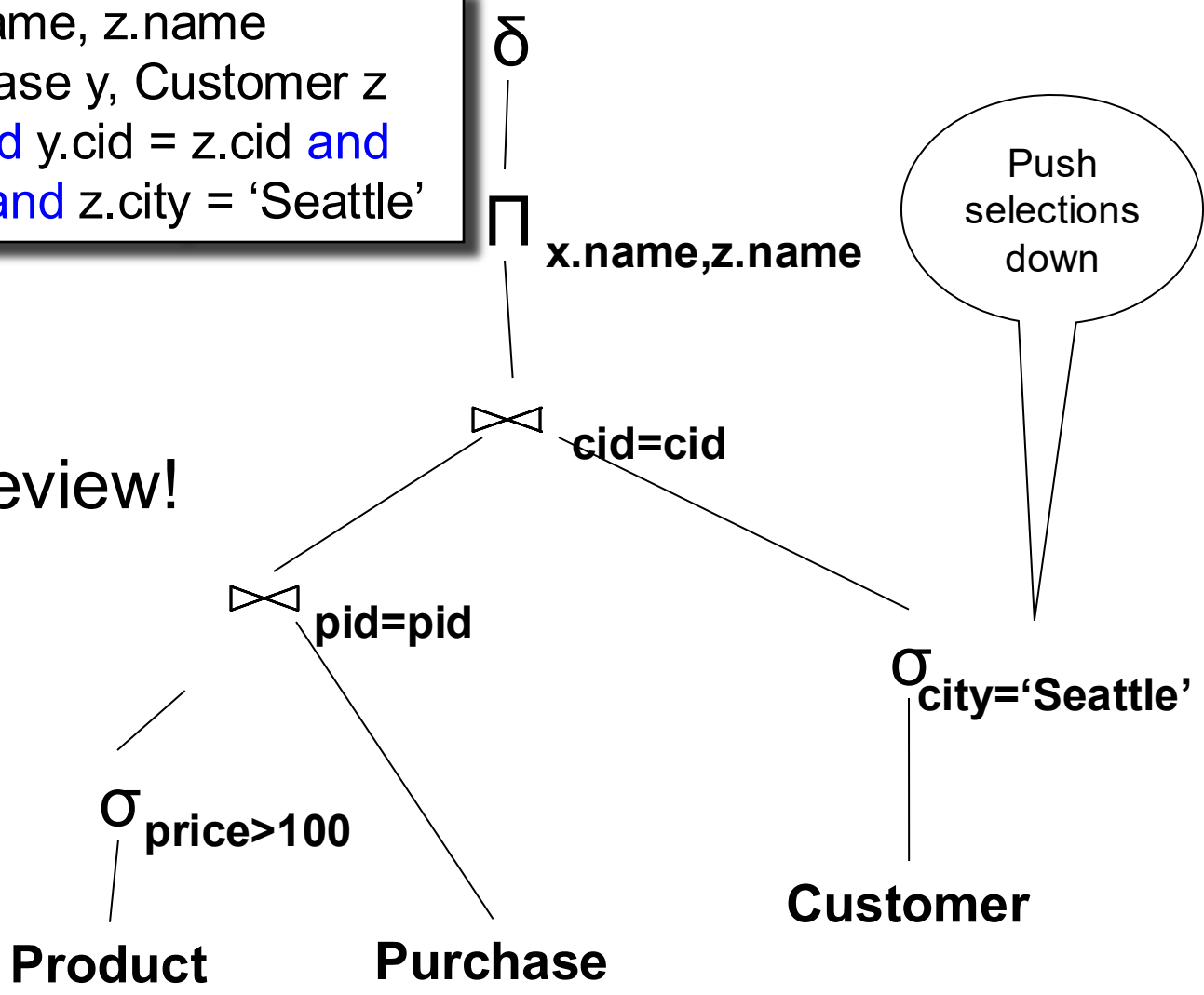
```
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = y.cid and
      x.price > 100 and z.city = 'Seattle'
```



Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

Optimization

```
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = z.cid and
      x.price > 100 and z.city = 'Seattle'
```



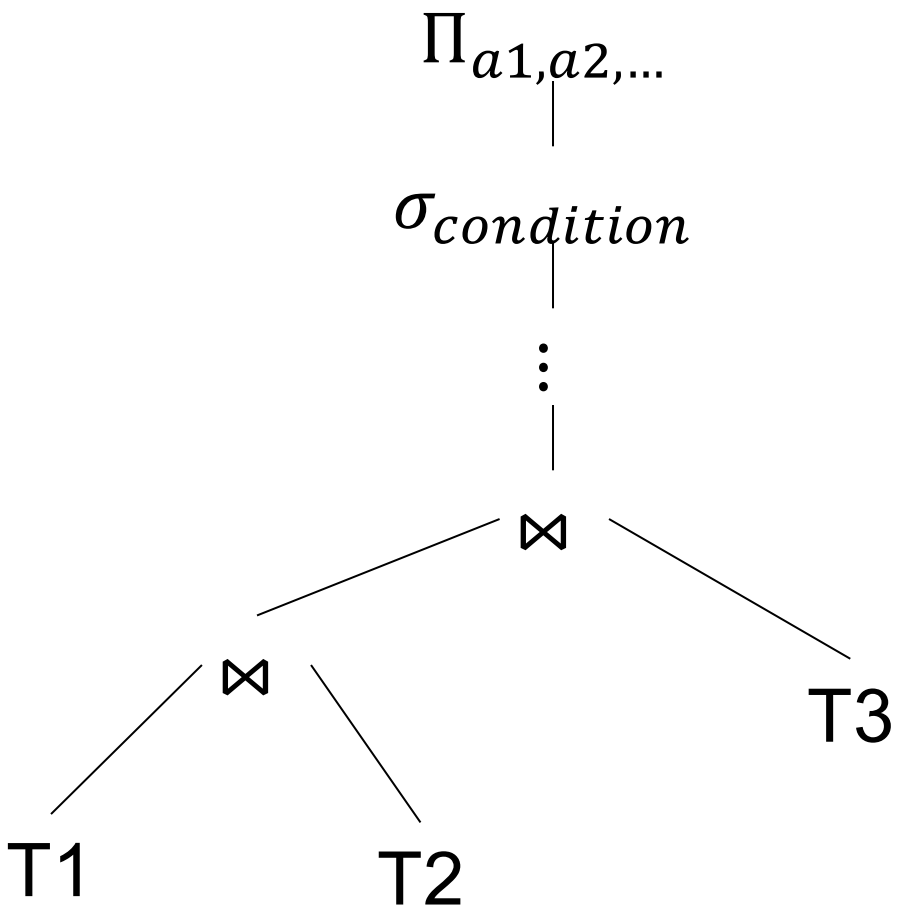
This is a quick preview!

More about this next lectures

SQL to RA

Simple SFW

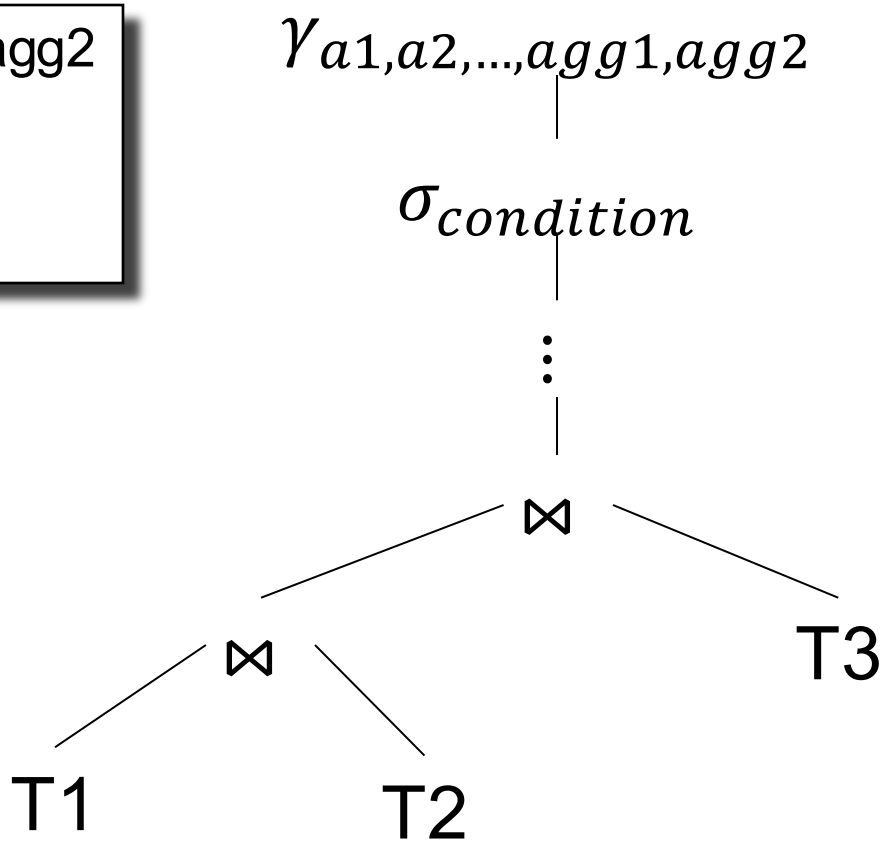
```
SELECT a1,a2,…
FROM T1, T2, …
WHERE condition
```



SQL to RA

…add GROUP-BY

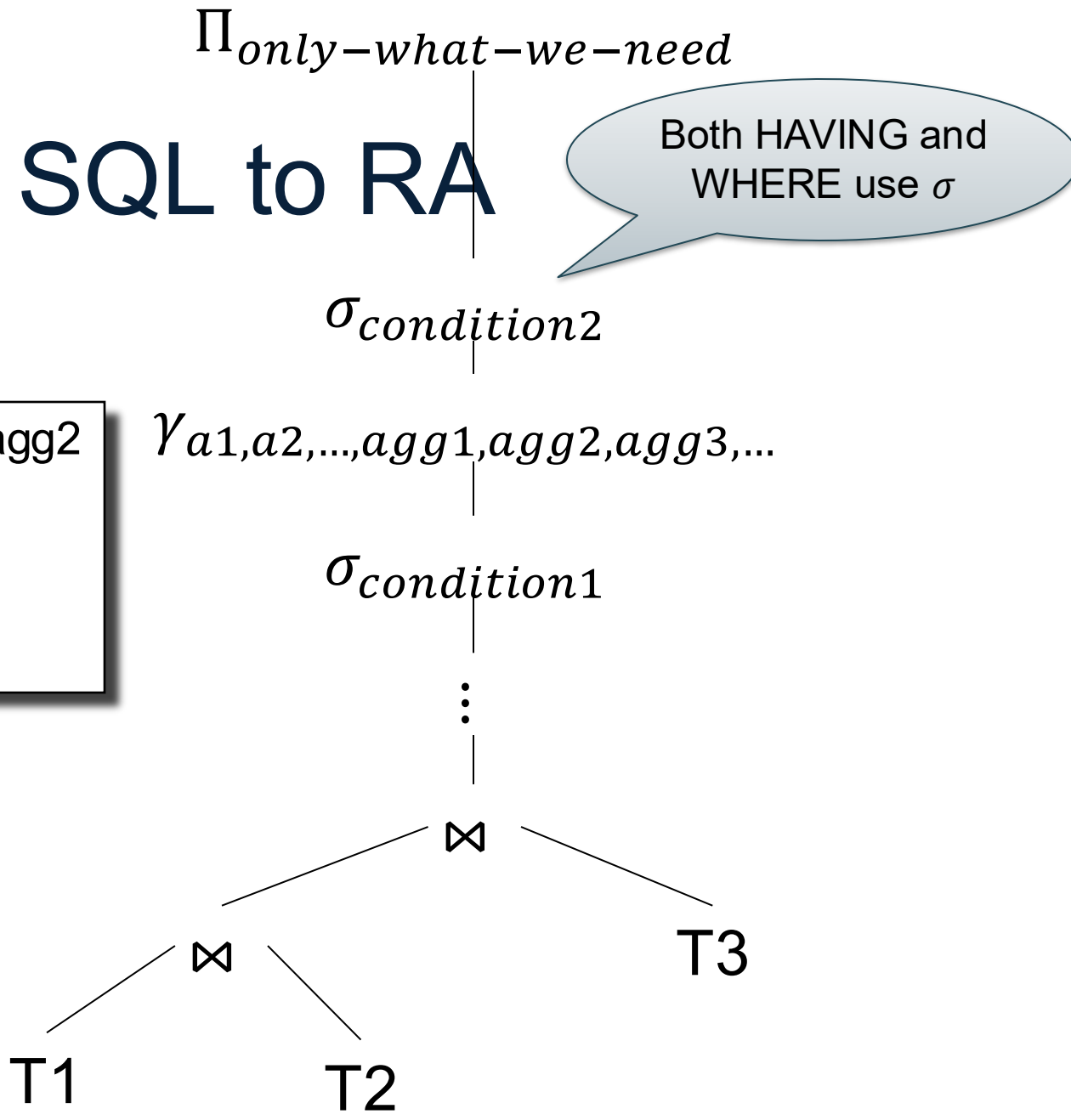
```
SELECT a1,a2,…,agg1,agg2
FROM T1, T2, …
WHERE condition
GROUP BY a1,a2,…
```



SQL to RA

…add HAVING

```
SELECT a1,a2,…,agg1,agg2
FROM T1, T2, …
WHERE condition1
GROUP BY a1,a2,…
HAVING condition2
```



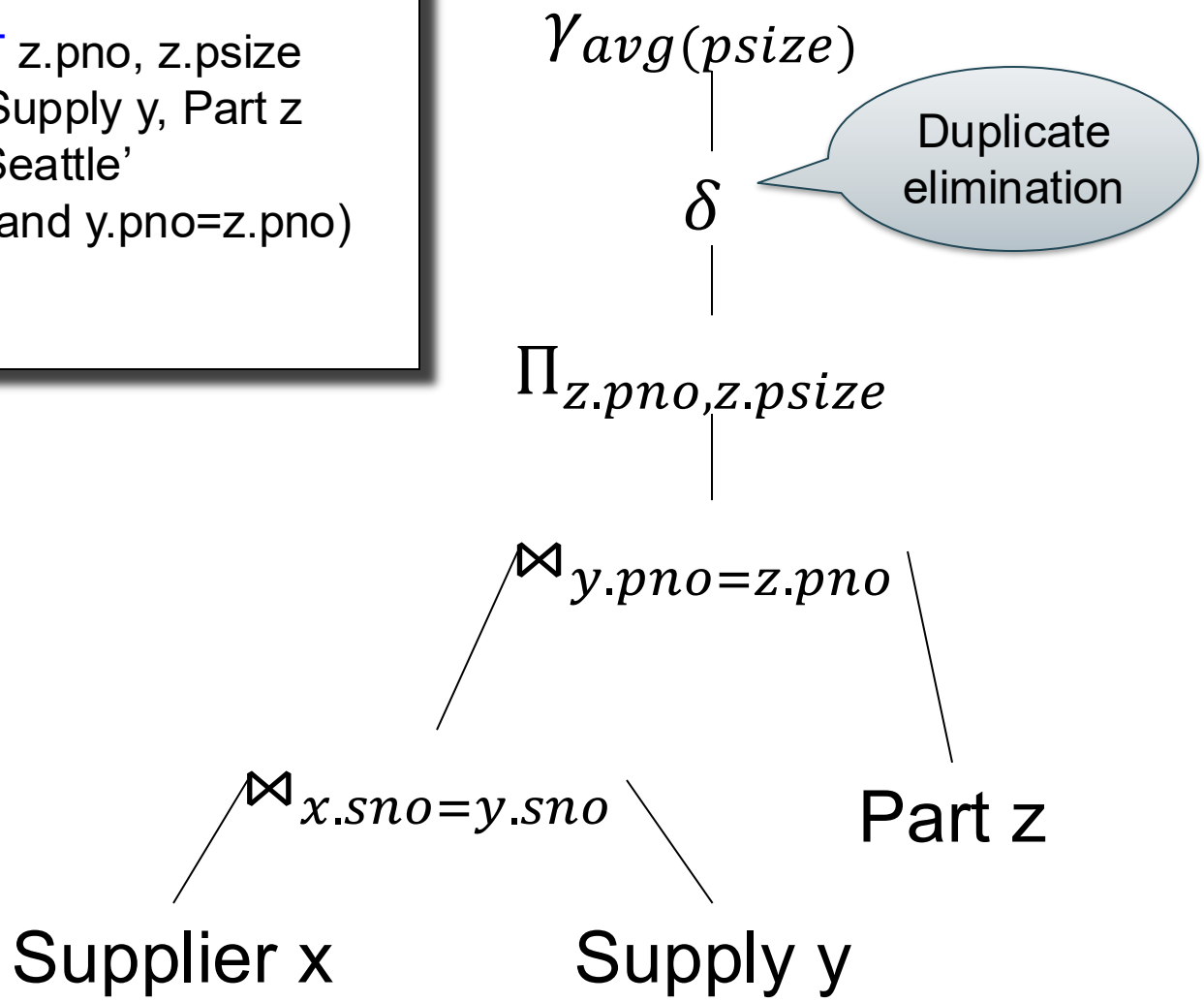
SQL to RA

- SQL queries without subqueries are straightforward to translate
- Subqueries may need to be flattened, then translated to SQL -- next

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subqueries

```
WITH Tmp AS (
        SELECT DISTINCT z.pno, z.psize
        FROM Supplier x, Supply y, Part z
        WHERE x.scity = 'Seattle'
           and x.sno=y.sno and y.pno=z.pno)
SELECT avg(psize)
FROM Tmp;
```



Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subqueries

Find all suppliers in 'WA' that supply *only* parts under $100

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Subqueries

```
SELECT  x.sno
FROM Supplier x
WHERE  x.sstate = 'WA'
    and not exists
       (SELECT *
        FROM Supply y
        WHERE x.sno = y.sno
              and y.price > 100)
```

Find all suppliers in 'WA' that supply *only* parts under $100

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Subqueries

```
SELECT  x.sno
FROM Supplier x
WHERE  x.sstate = 'WA'
    and not exists
       (SELECT *
        FROM Supply y
        WHERE x.sno = y.sno
              and y.price > 100)
```

Find all suppliers in 'WA' that supply *only* parts under $100

Translate to RA

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subqueries

```
SELECT  x.sno
FROM Supplier x
WHERE  x.sstate = 'WA'
    and not exists
       (SELECT *
        FROM Supply y
        WHERE x.sno = y.sno
              and y.price > 100)
```


Supplier x

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subqueries

```
SELECT  x.sno
FROM Supplier x
WHERE  x.sstate = 'WA'
    and not exists
       (SELECT *
        FROM Supply y
        WHERE x.sno = y.sno
              and y.price > 100)
```

$$\sigma_{x.sstate='WA' \wedge \neg\exists(\ldots)}$$

Supplier x Supply y

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subqueries

```
SELECT  x.sno
FROM Supplier x
WHERE  x.sstate = 'WA'
    and not exists
       (SELECT *
        FROM Supply y
        WHERE x.sno = y.sno
              and y.price > 100)
```

Totally wrong!!

$$\sigma_{x.sstate='WA' \wedge \neg\exists(\dots)}$$

Supplier x Supply y

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subqueries

```
SELECT  x.sno
FROM Supplier x
WHERE  x.sstate = 'WA'
   and not exists
      (SELECT *
       FROM Supply y
       WHERE x.sno = y.sno
             and y.price > 100)
```

Need to unnest

Totally wrong!!

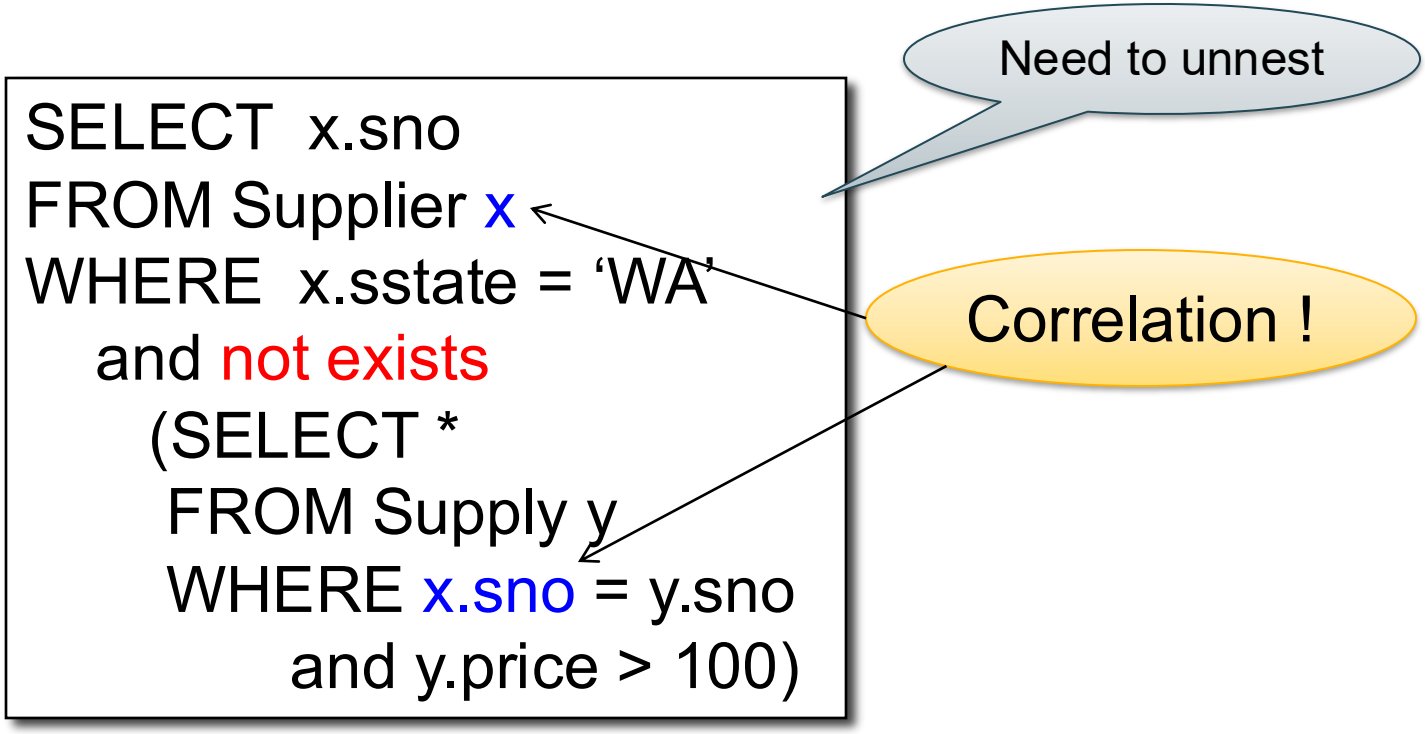
$\sigma_{x.sstate='WA' \wedge \neg\exists(\dots)}$

Supplier x

Supply y

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subqueries



Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subqueries

```
SELECT  x.sno
FROM Supplier x
WHERE  x.sstate = 'WA'
    and not exists
       (SELECT *
        FROM Supply y
        WHERE x.sno = y.sno
              and y.price > 100)
```

De-Correlation

```
SELECT  x.sno
FROM Supplier x
WHERE  x.sstate = 'WA'
    and x.sno not in
       (SELECT y.sno
        FROM Supply y
        WHERE y.price > 100)
```

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subqueries

Un-nest

```
(SELECT  x.sno
 FROM Supplier x
 WHERE  x.sstate = 'WA')
   EXCEPT
 (SELECT y.sno
  FROM Supply y
  WHERE y.price > 100)
```

```
SELECT  x.sno
FROM Supplier x
WHERE  x.sstate = 'WA'
   and x.sno not in
      (SELECT y.sno
       FROM Supply y
       WHERE y.price > 100)
```

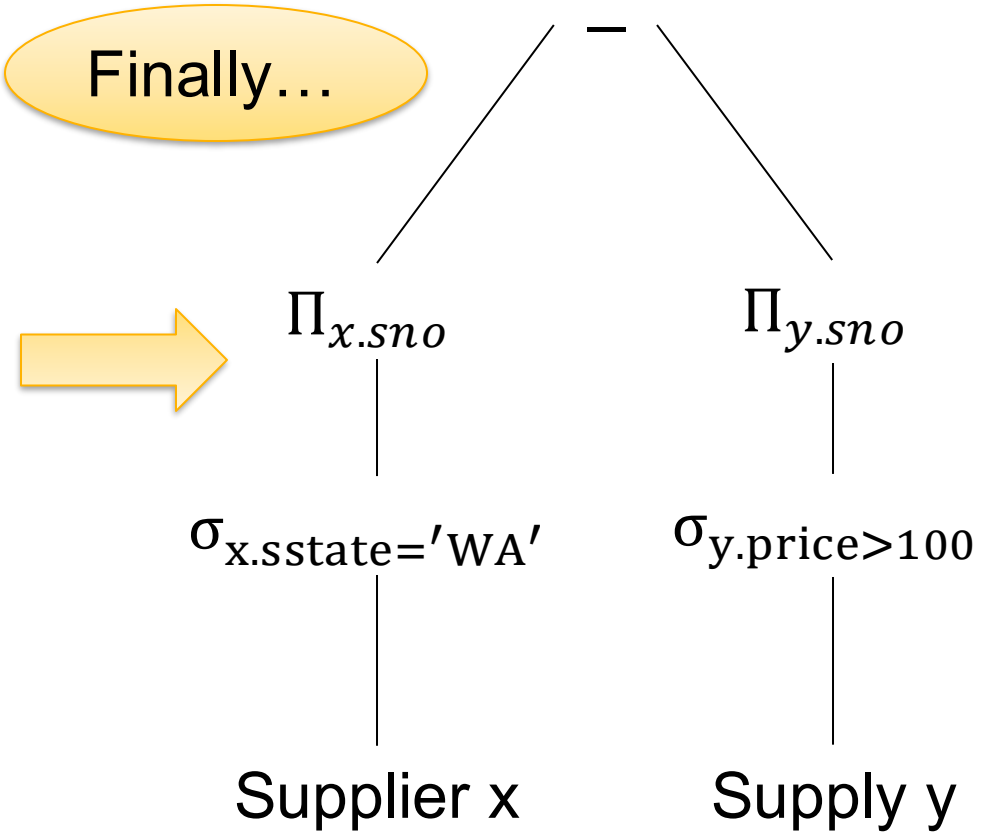
EXCEPT = set difference

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subqueries

```
(SELECT  x.sno
 FROM Supplier x
 WHERE  x.sstate = 'WA')
   EXCEPT
 (SELECT y.sno
  FROM Supply y
  WHERE y.price > 100)
```

Finally…



Summary

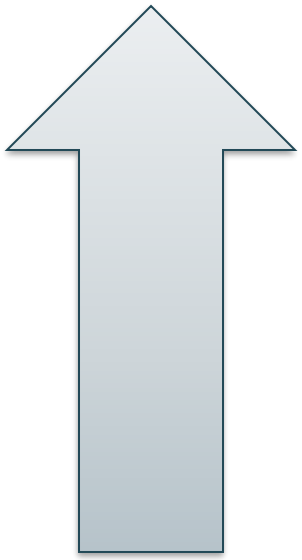
- User writes in SQL
  - Declarative language
  - Users say WHAT they want
- System: SQL → Relational Algebra
  - Explicit operation order: HOW to compute
  - RA expression a.k.a. Query Plan
- Query Plan: is optimized, then executed

Next lectures

Database Internals

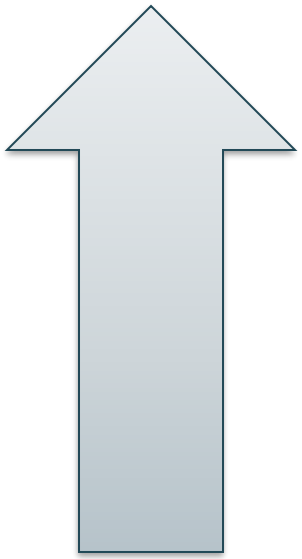
Database Management System

- Query optimizer
- Operator execution
- Access method
- Buffer pool manager
- Disk manager



Database Management System

- Query optimizer
- Operator execution
- Access method
- Buffer pool manager
- Disk manager

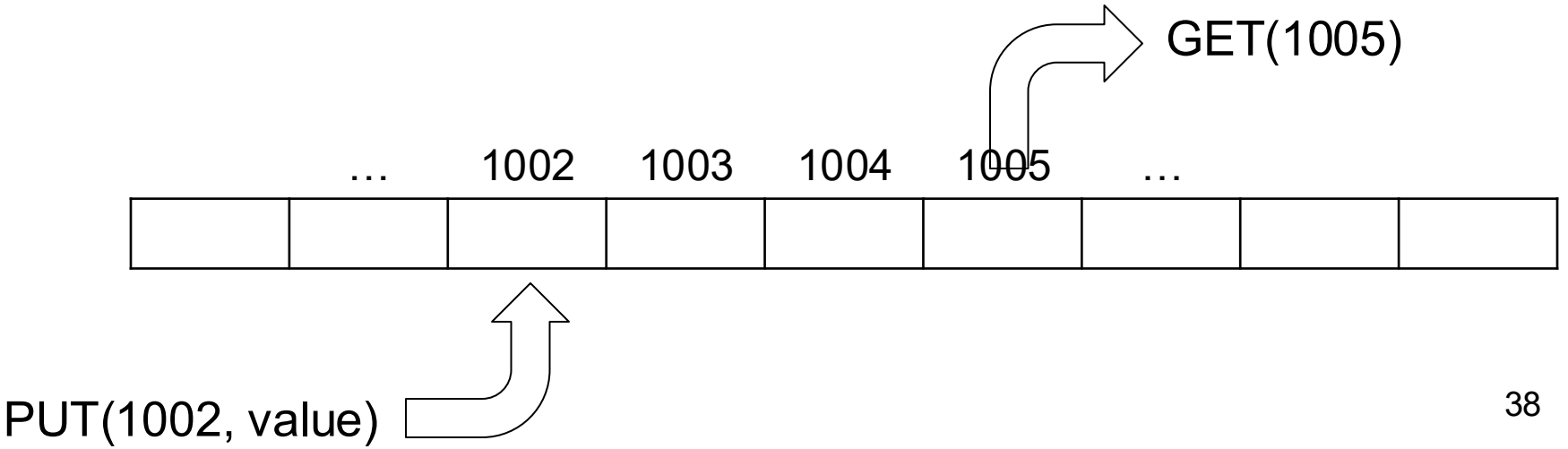


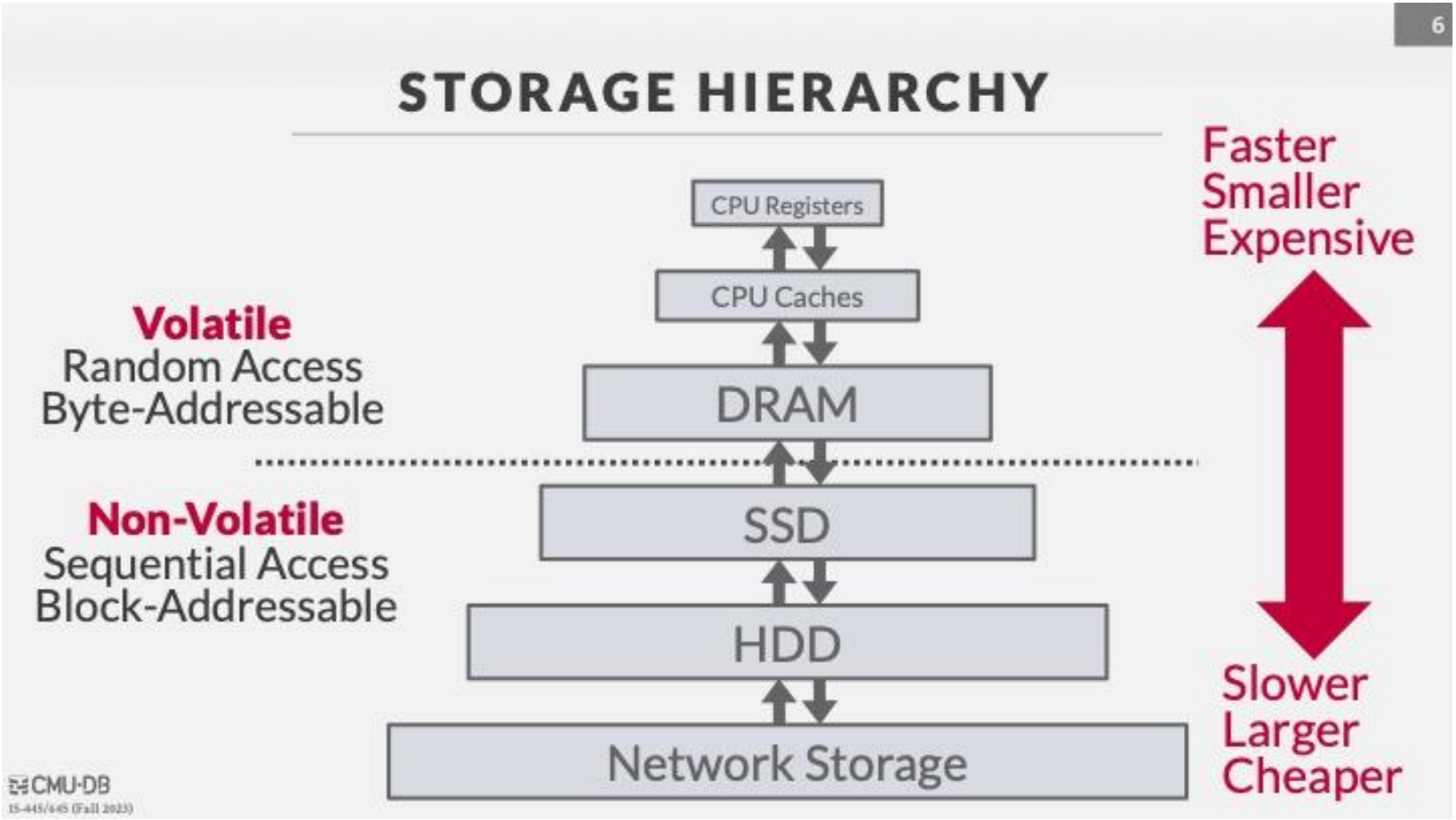
Today: storage manager and (briefly) buffer manager

Storage Manager

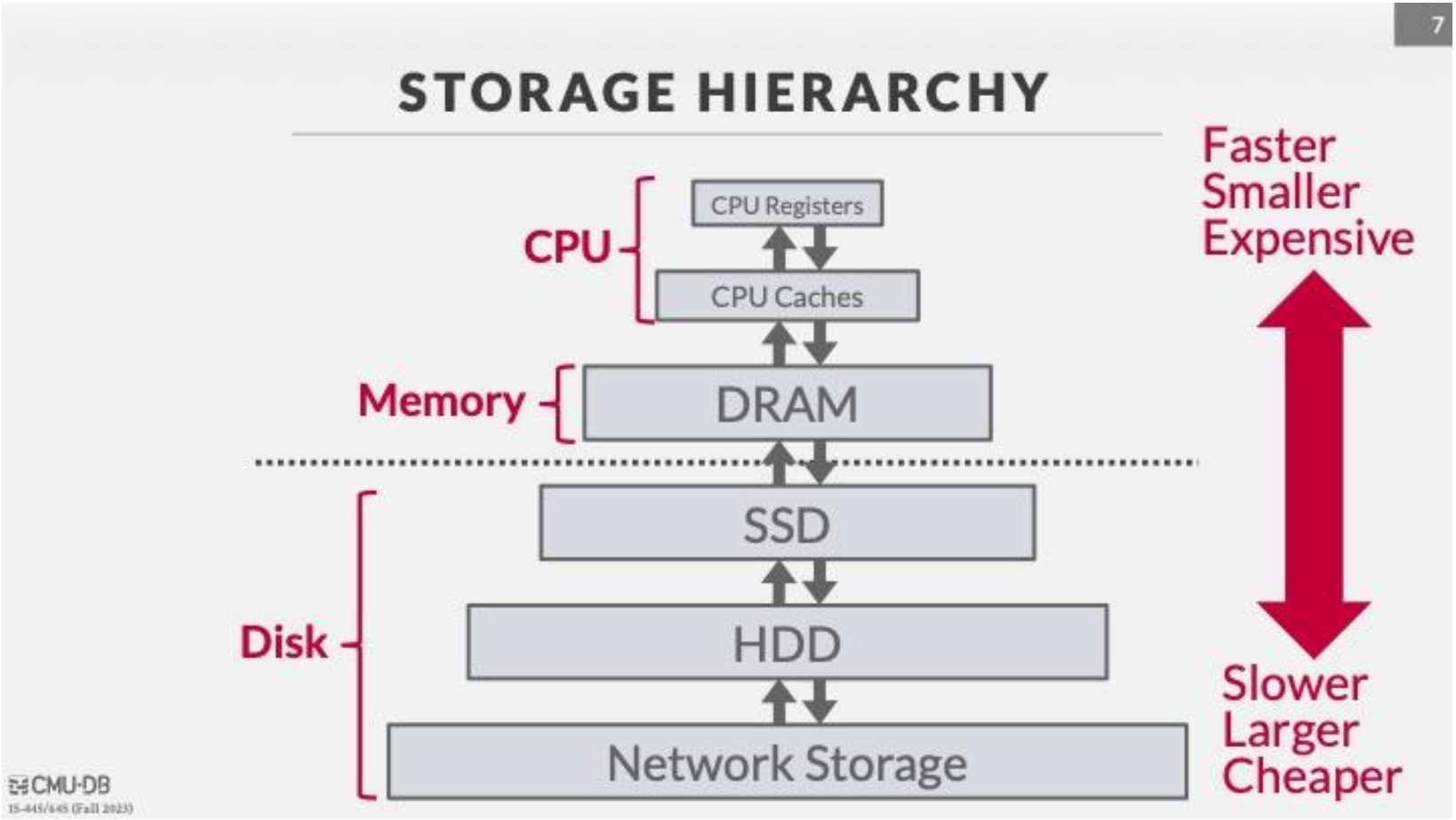
Basics of Storage Methods

- Multiple storage layers: disk, main memory, cache, registers
- Each layer: an array of locations
- Location: word or page/block

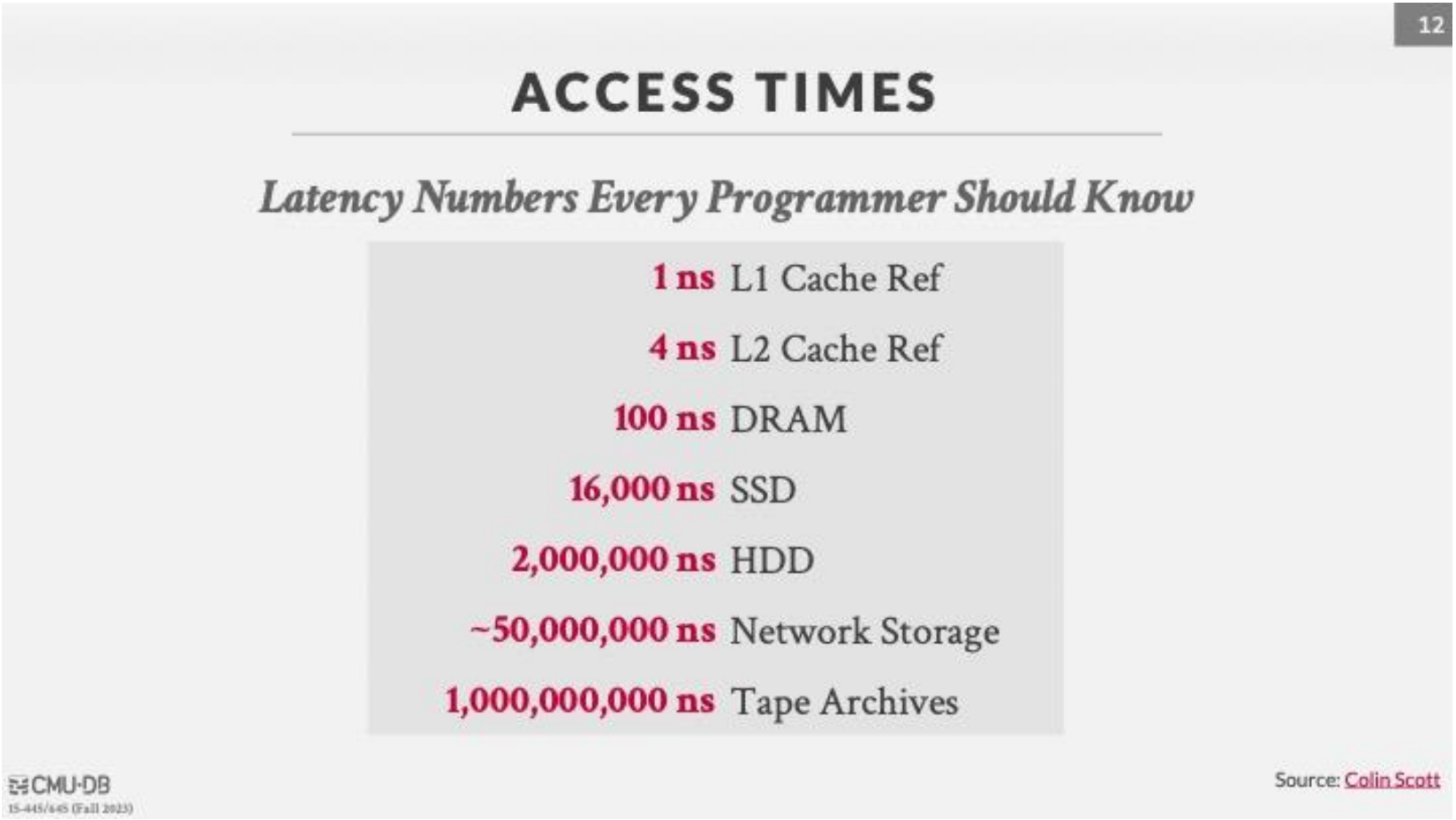




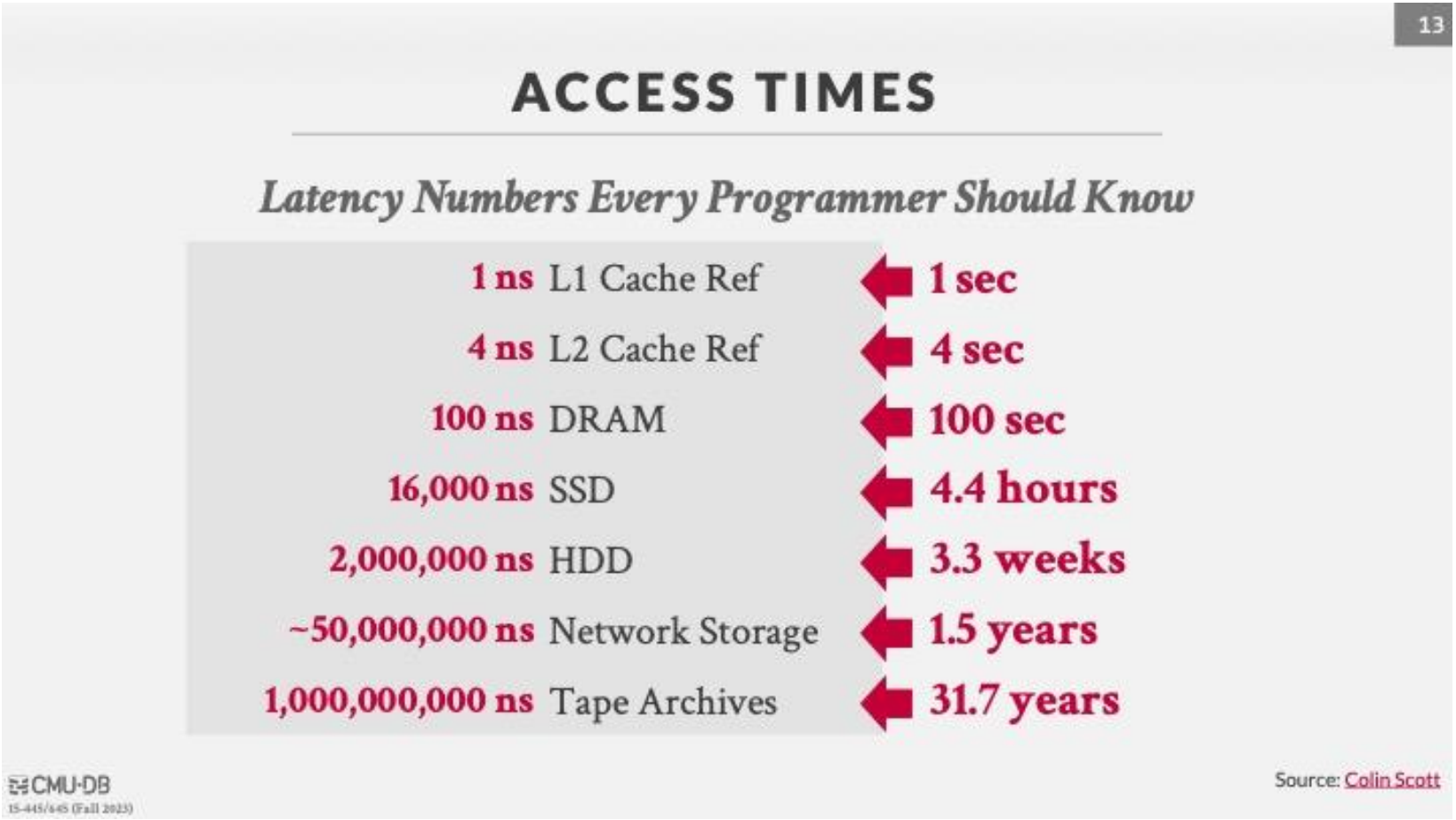
Credit: https://15445.courses.cs.cmu.edu/fall2023/



Credit: https://15445.courses.cs.cmu.edu/fall2023/



Credit: https://15445.courses.cs.cmu.edu/fall2023/



Credit: https://15445.courses.cs.cmu.edu/fall2023/

The Disk and the Buffer Pool

Disk

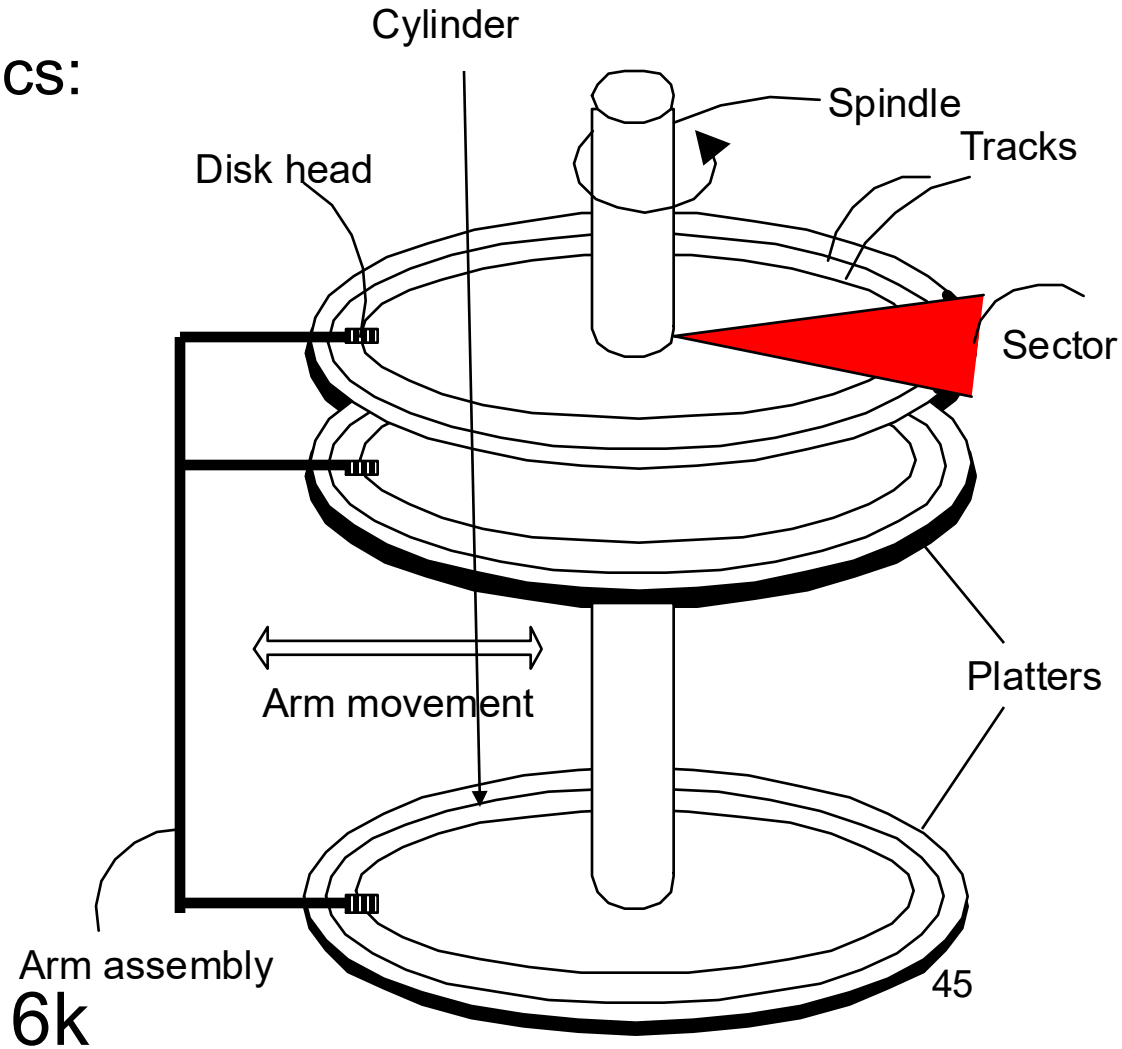
- Stores data persistently
- Different technologies:
  - Tapes: long-term archives
  - HDD: most today's data is stored here
  - SDD: your laptop, or cache for HDD
- Unit of Read/Write operation:

  1 block = 0.5k .. 32k

Hard Drive Disk (HDD)

Mechanical characteristics:

- 5400 RPM
- 1-30 platters
- ≤10000 tracks
- 10^5 bytes/track



Unit of read or write:

disk block

Once in memory:

page

Typically: 4k or 8k or 16k

Disk Access Characteristics

- Disk latency
  - Seek time + rotational latency
- Seek time = time for head to reach cylinder
  - 10 – 40ms
- Rotational latency = time for sector to rotate
  - Rotation time = 10ms
  - Average latency = 10ms / 2
- Transfer time = typically 40-80MB/s

Slow!

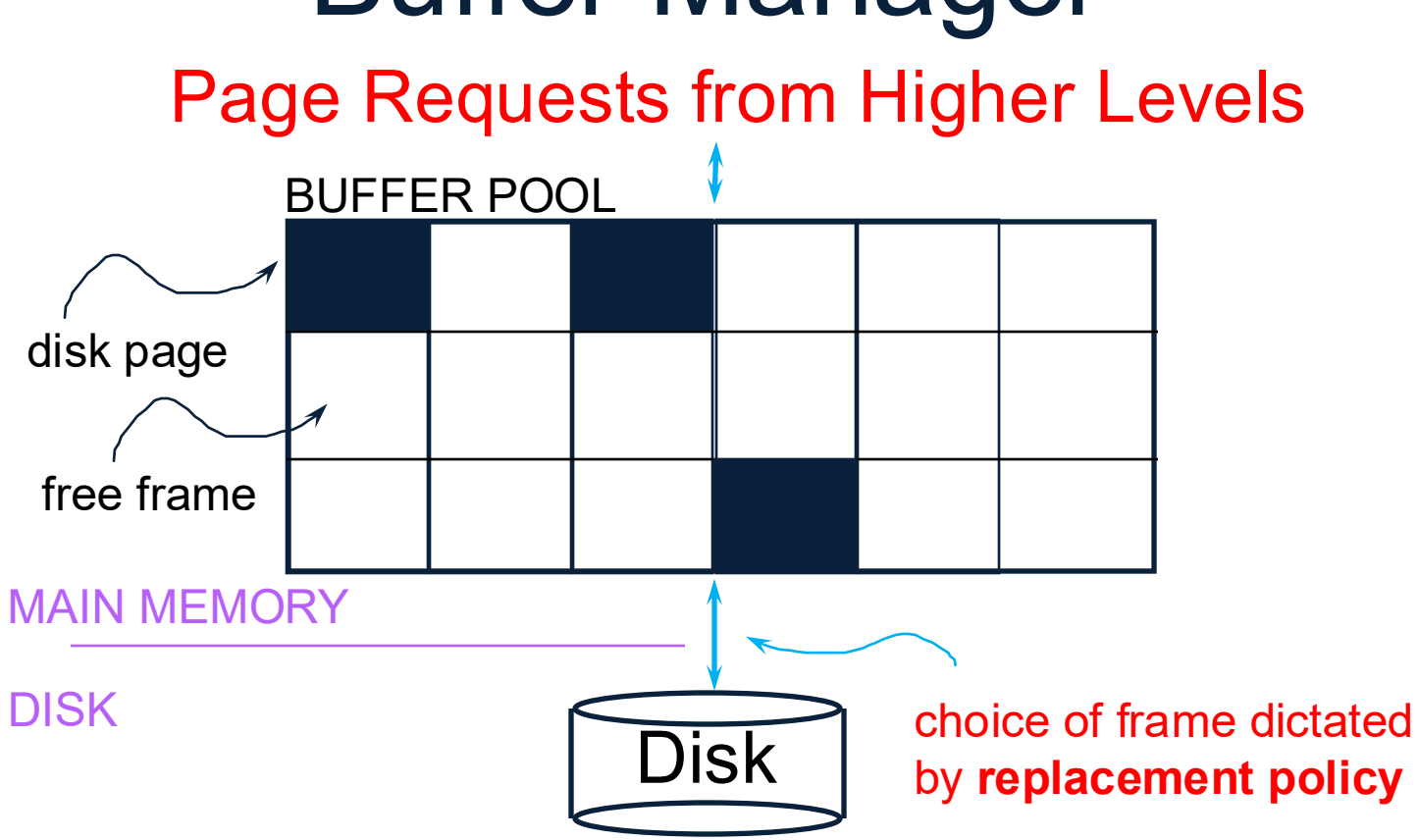
The Buffer Pool

- Fixed chunk of main memory
- When a page is read from disk, it is brought to the buffer pool

The Buffer Pool

- Fixed chunk of main memory
- When a page is read from disk, it is brought to the buffer pool
- If same page is requested later, read it from the buffer pool: saves disk I/O
- If a new page is read, but no room in the buffer pool, one page is evicted

Buffer Manager



- Table of <frame#, pageid> pairs

Buffer Manager

Needs a page replacement policy

- LRU: Least Recently Used (in class)
- Clock algorithm (on your own)

Terminology (Recap)

- Block:
  - Unit of memory accessed from disk
  - HW dependent: 8k or 16k or 32k or…
- Page:
  - Block content as it moves around
- Frame:
  - Fixed location in buffer to store a page

Storage Manager

Maps between the logical view of the data and the physical storage on disk

- Block#
- Page#
- Frame#

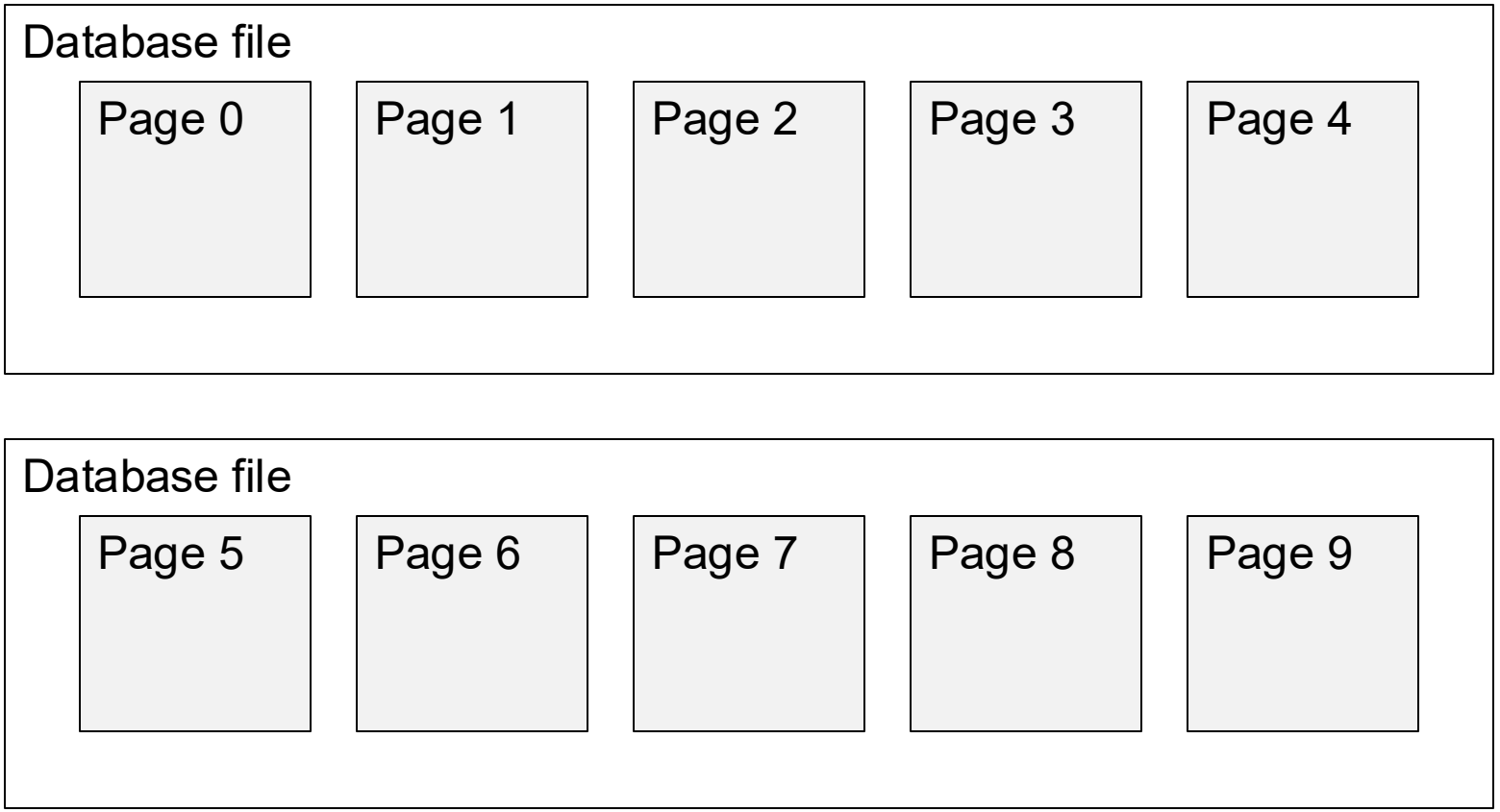
HW3!

Storage Manager

Modern DBMS use OS to create files

- 1 file = multiple (continuous?) blocks
  today
- 1 block = multiple records, free space
  next lecture

Storing Pages on Disk



Catalog of metadata: files, pages, records, free space

File Formats

File Formats

1. Sequential
2. Sorted
3. Index

1. Sequential File

Order of the records is unspecified

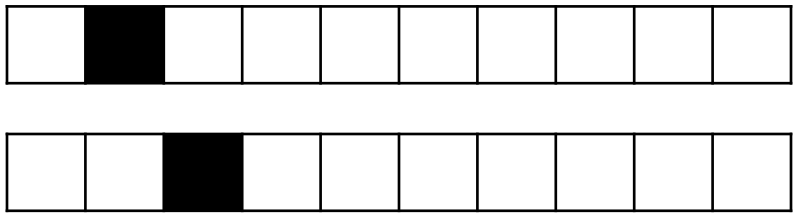
- Sequential access
- Random access

Sequential/Random Access

Sequential

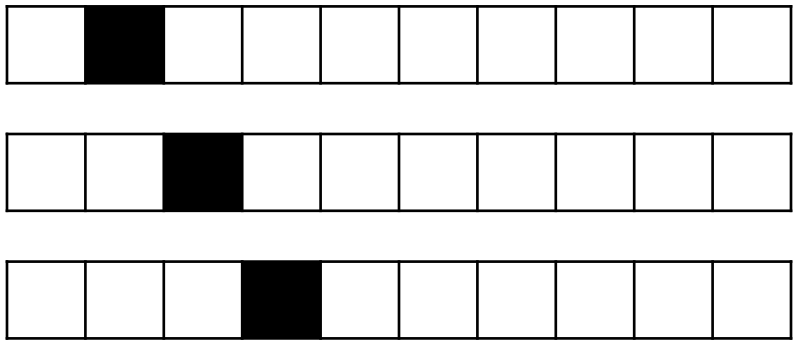
Sequential/Random Access

Sequential



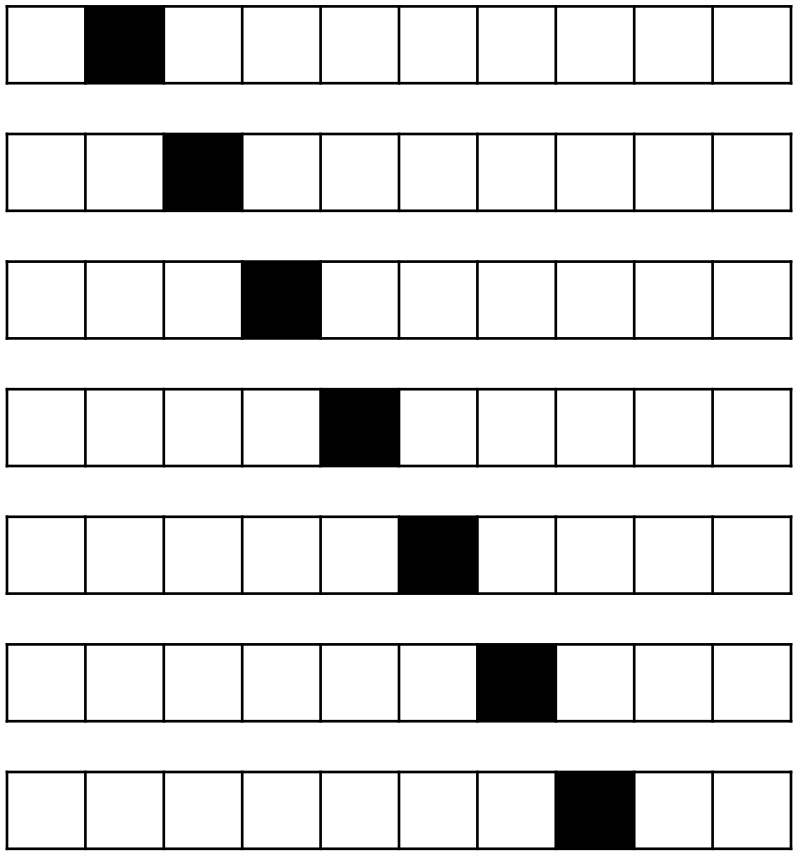
Sequential/Random Access

Sequential



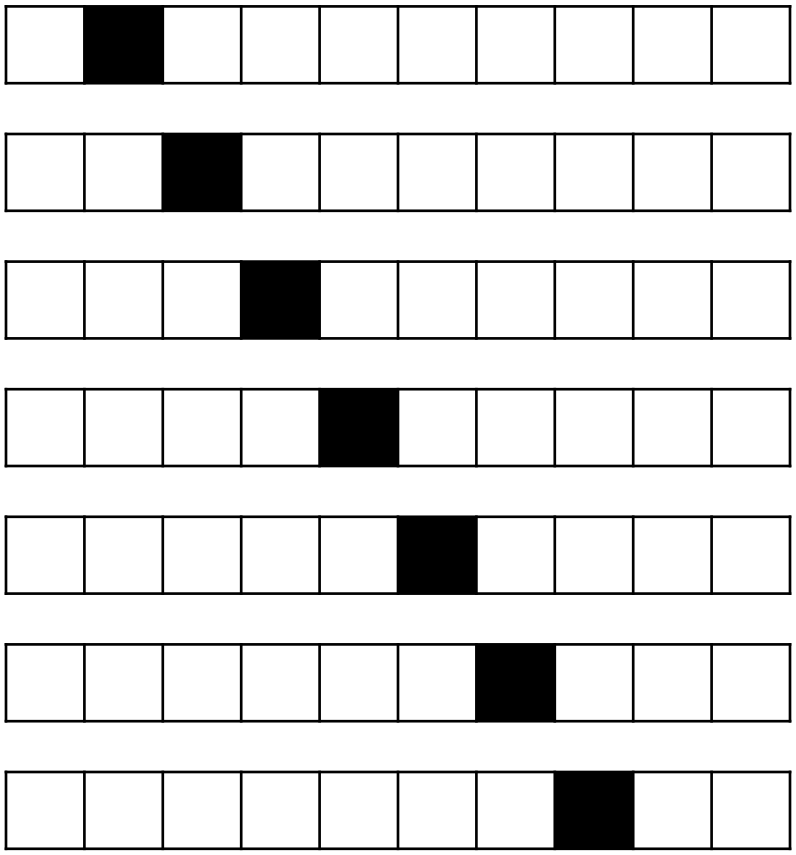
Sequential/Random Access

Sequential

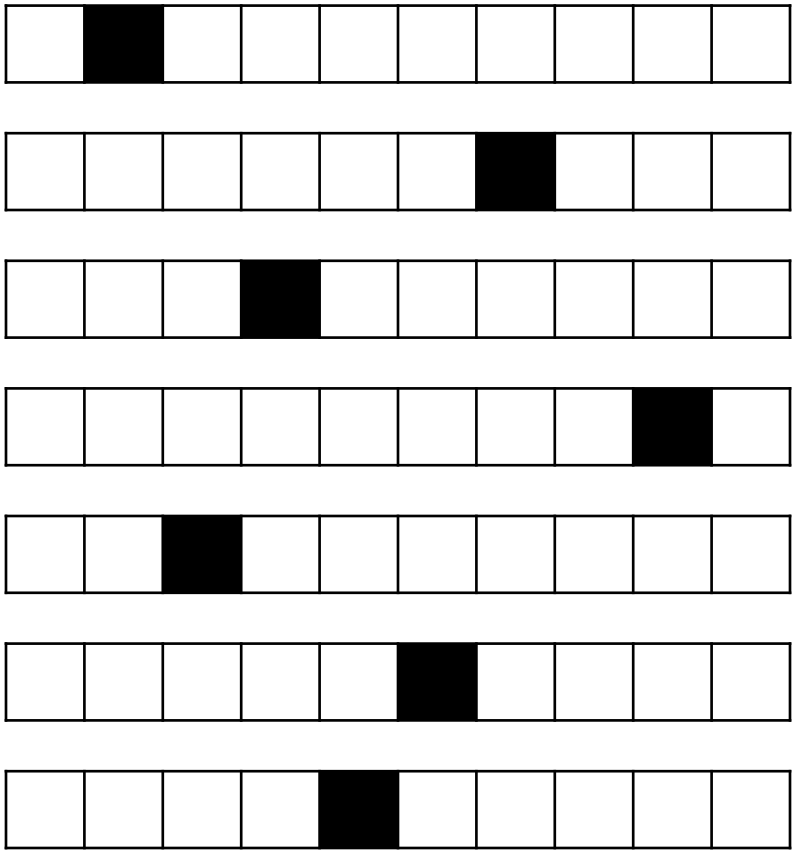


Sequential/Random Access

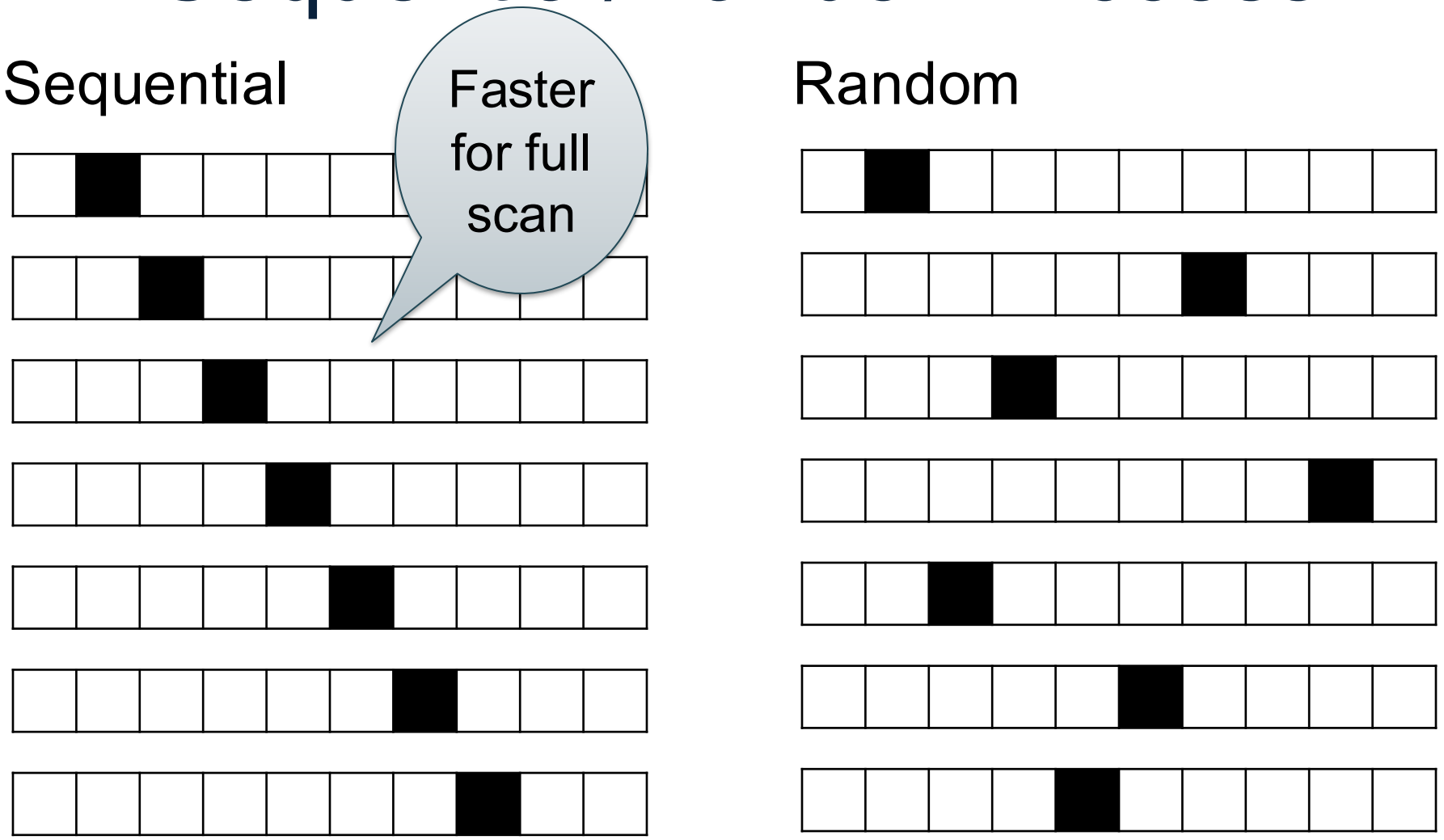
Sequential



Random

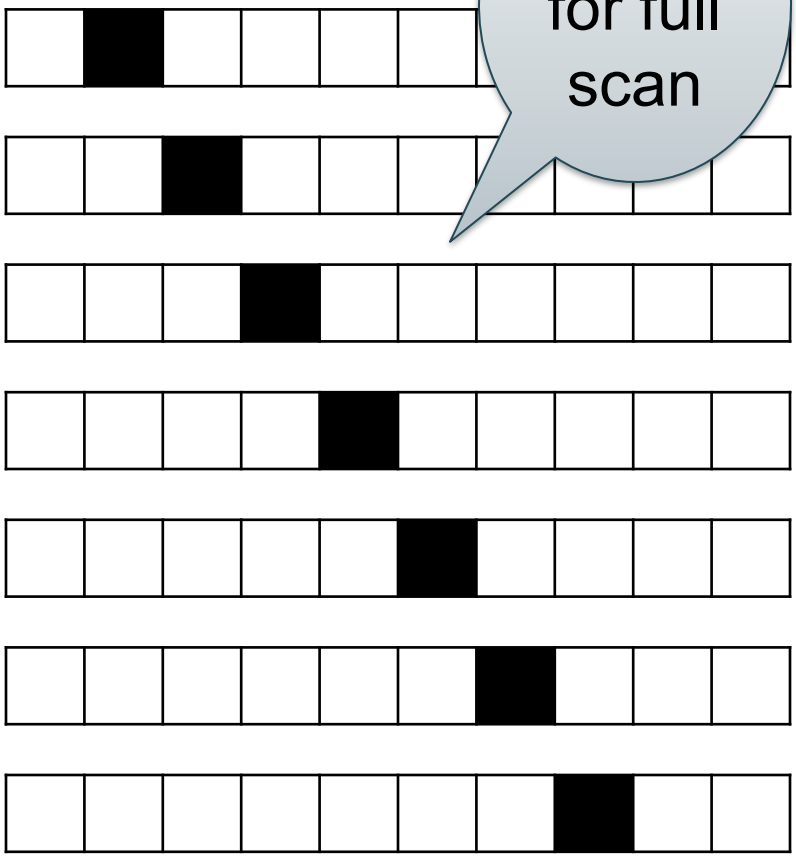


Sequential/Random Access

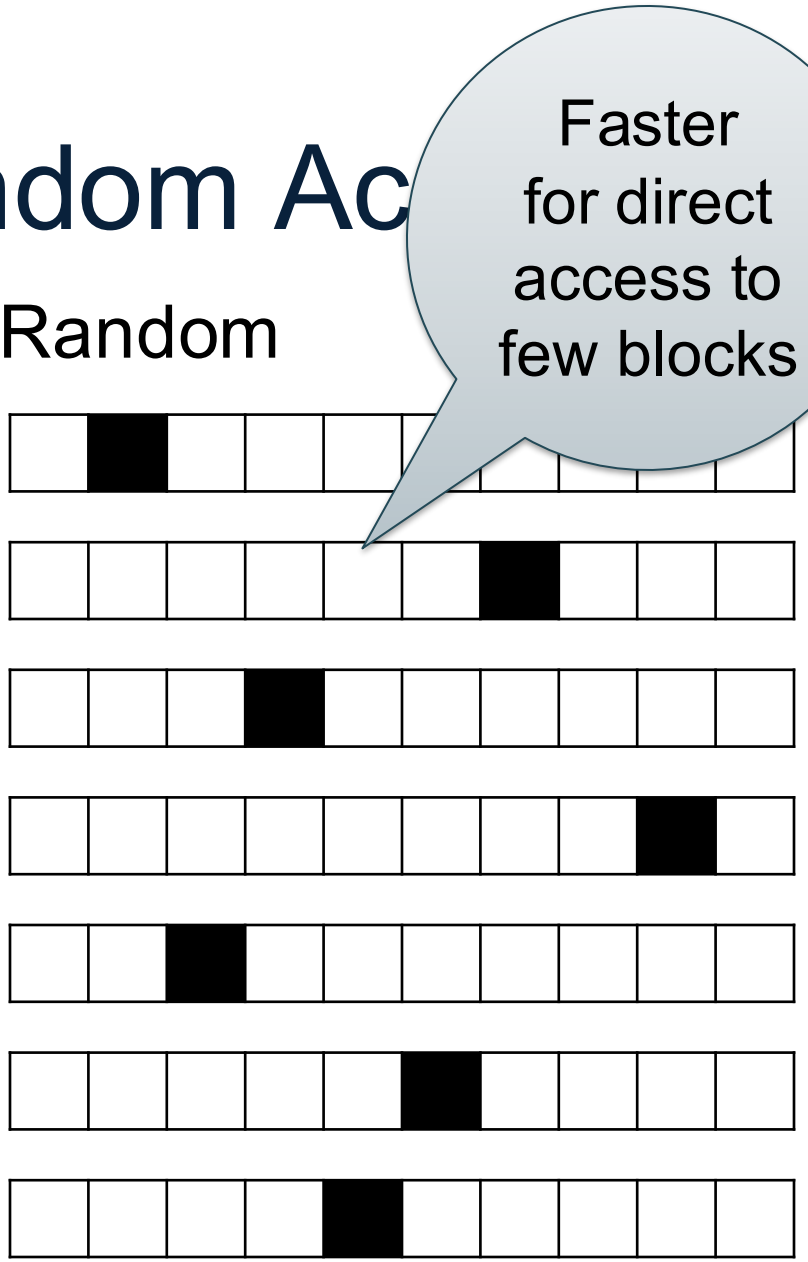


Sequential/Random Ac

Sequential



Random



Sequential File: Summary

- Most common format for large data
- May add index file for direct access
- May sort for even faster access

2. Sorted File

- In addition to sequential/random search, we can also perform binary search

Binary Search

Sorted array:

12	24	31	37	42	48	52	59	61	66	69	75	88	92	95

Binary Search

Sorted array:

12	24	31	37	42	48	52	59	61	66	69	75	88	92	95

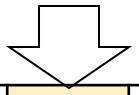
Find 42

Binary Search

Sorted array:

12	24	31	37	42	48	52	59	61	66	69	75	88	92	95

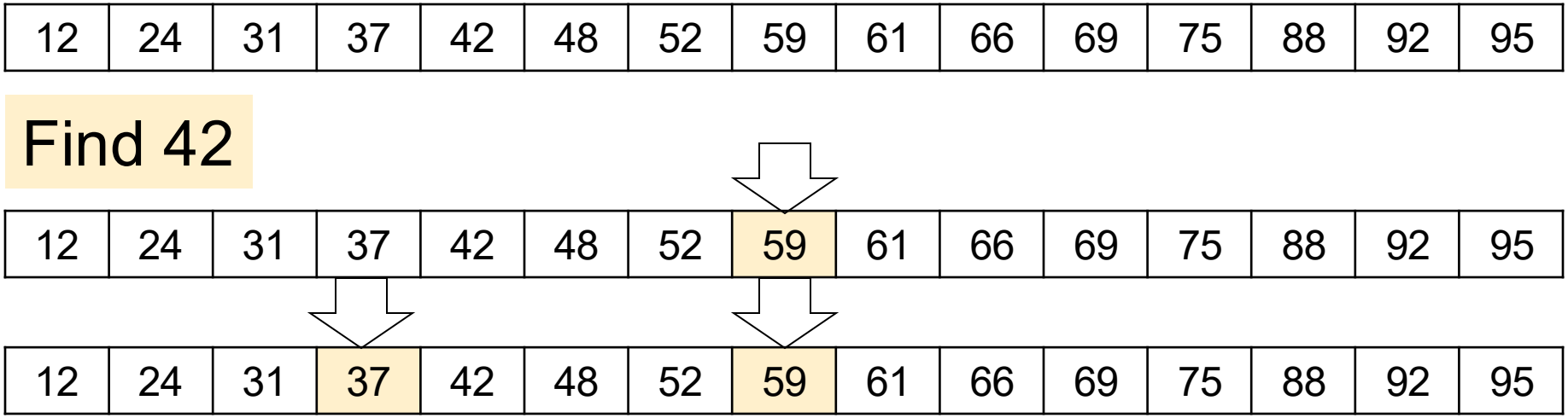
Find 42



12	24	31	37	42	48	52	59	61	66	69	75	88	92	95

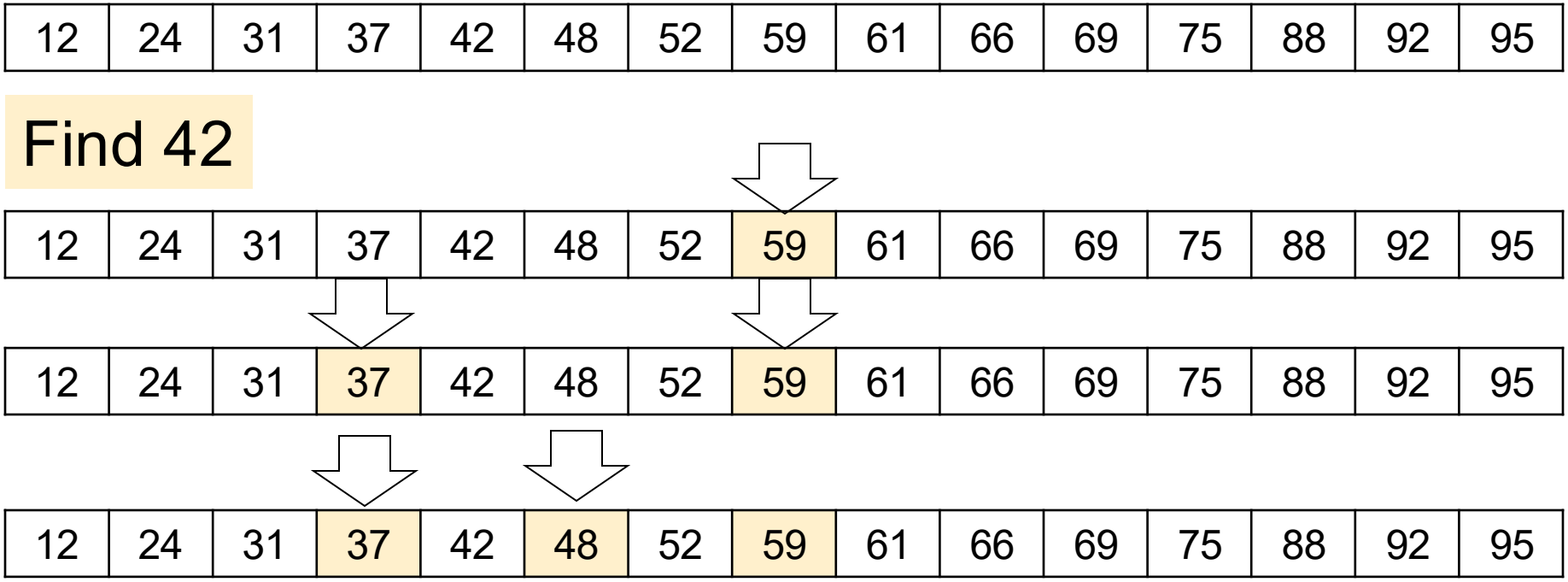
Binary Search

Sorted array:



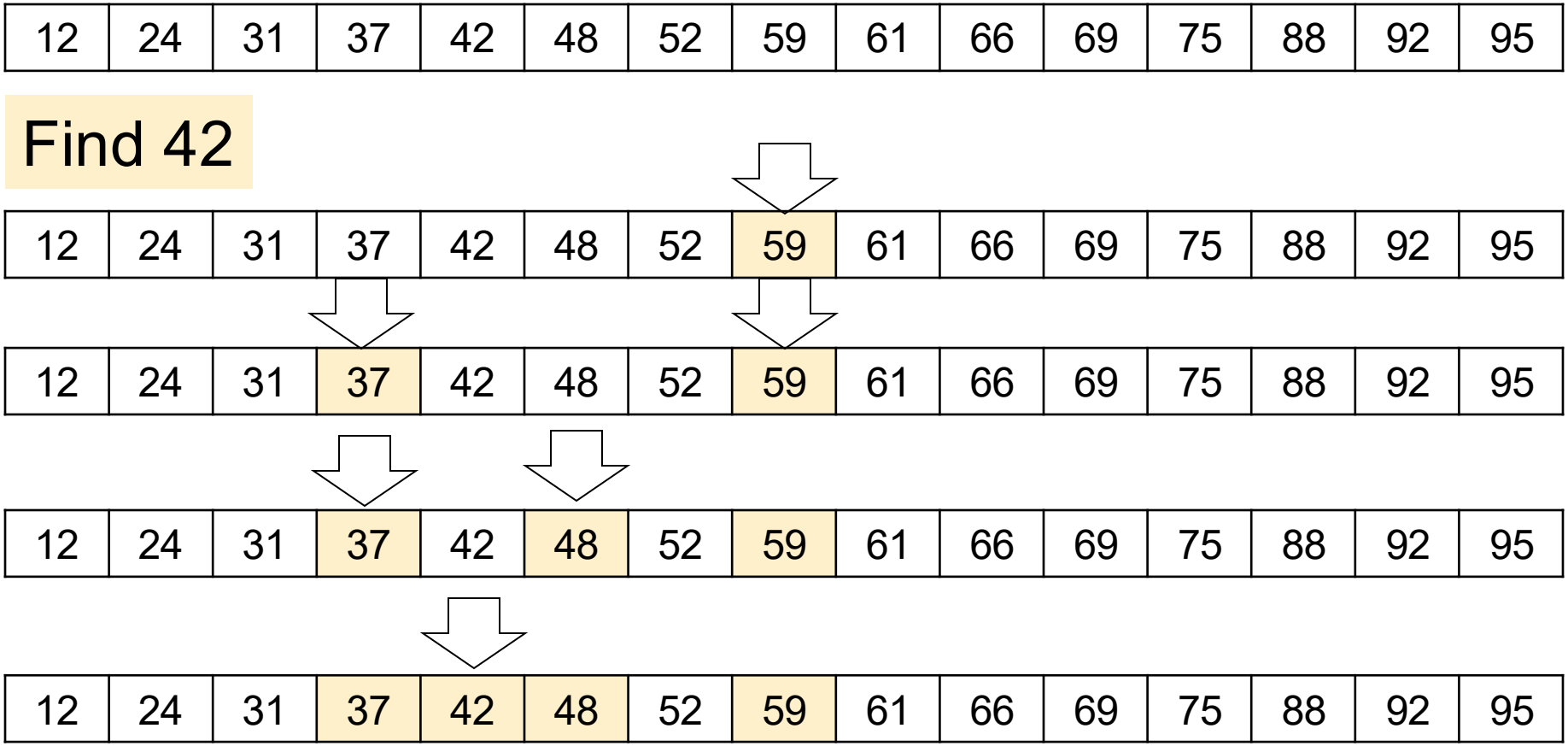
Binary Search

Sorted array:



Binary Search

Sorted array:



Binary Search

- log N comparisons
- Any algorithm needs log N

Binary Search

- log N comparisons
- Any algorithm needs log N

- Ex. N = 1,000,000,000
  - Sequential search: ≈ 500,000,000 steps
  - Binary search: 30 steps

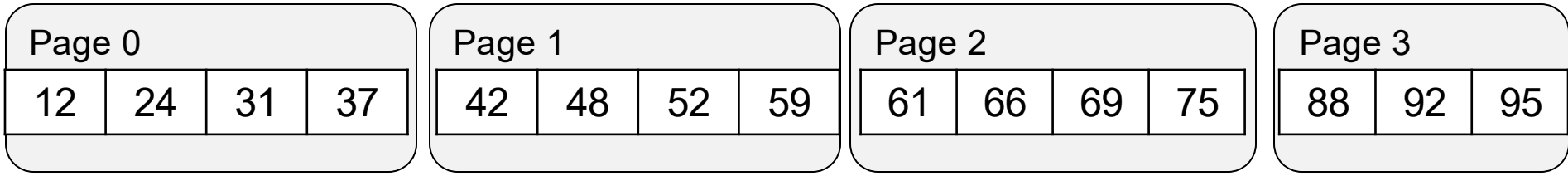
Binary Search

- log N comparisons
- Any algorithm needs log N

- Ex. N = 1,000,000,000
  - Sequential search: ≈ 500,000,000 steps
  - Binary search: 30 steps

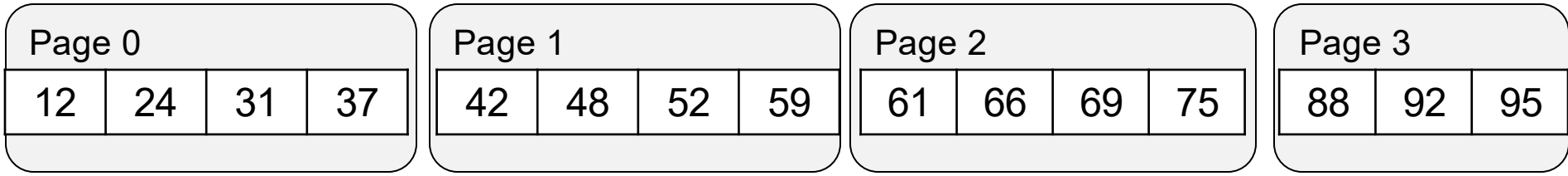
Sorted array + binary search = optimal!

Binary Search On Disk



What problem do you see?

Binary Search On Disk



What problem do you see?

Read one page, use only one key!

Solution? Wait for it!

Binary Search

Assume array is in main memory

Problem: updates with poor performance

- Insert: shift O(N) items to the right
- Delete: shift O(N) items to the left

Insert/Delete

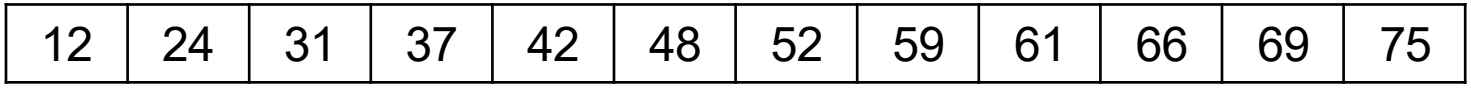
12	24	31	37	42	48	52	59	61	66	69	75

Insert/Delete

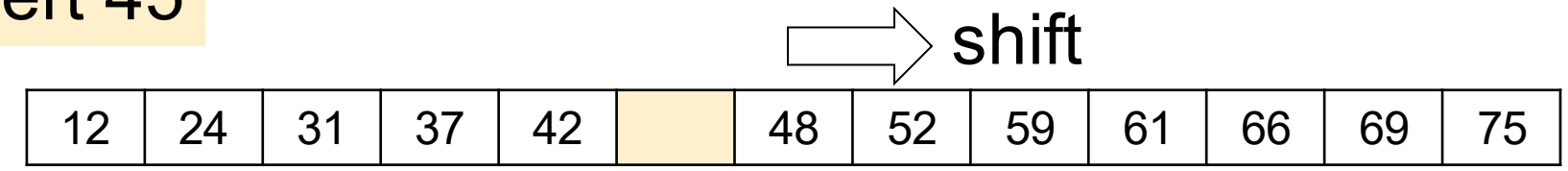
12	24	31	37	42	48	52	59	61	66	69	75

Insert 45

Insert/Delete



Insert 45



Insert/Delete

12	24	31	37	42	48	52	59	61	66	69	75

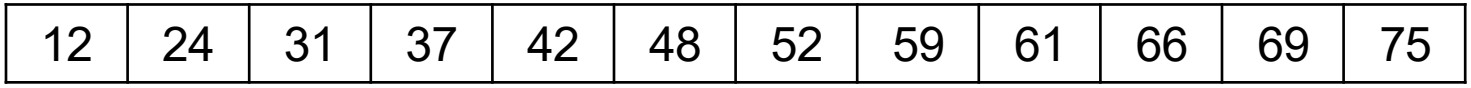
Insert 45

shift

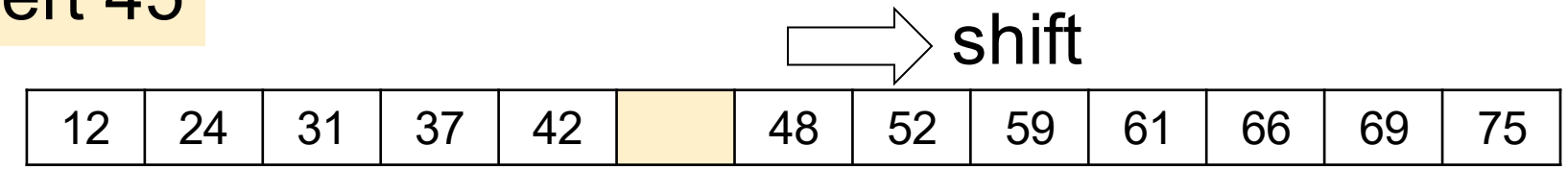
12	24	31	37	42		48	52	59	61	66	69	75

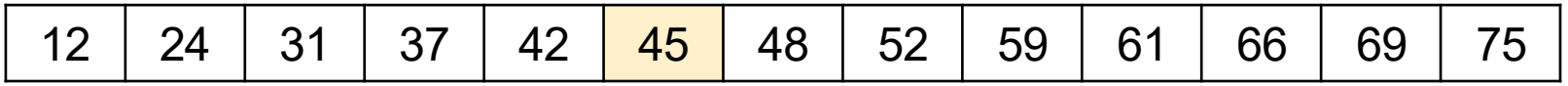
12	24	31	37	42	45	48	52	59	61	66	69	75

Insert/Delete



Insert 45





Delete 31

...

Sorted File: Summary

- Advantage: Fast lookup
- Disadvantage:
  - Updates need separate logic (B+ trees)
  - Can only sort on one attribute
- Terminology: the file is *clustered*

3. Index File

- File supports efficient lookup + update
- In class: B+ trees, Hash tables
- Other index structures: R-trees, inverted indexes, bitmaps, …

Review: Search Tree

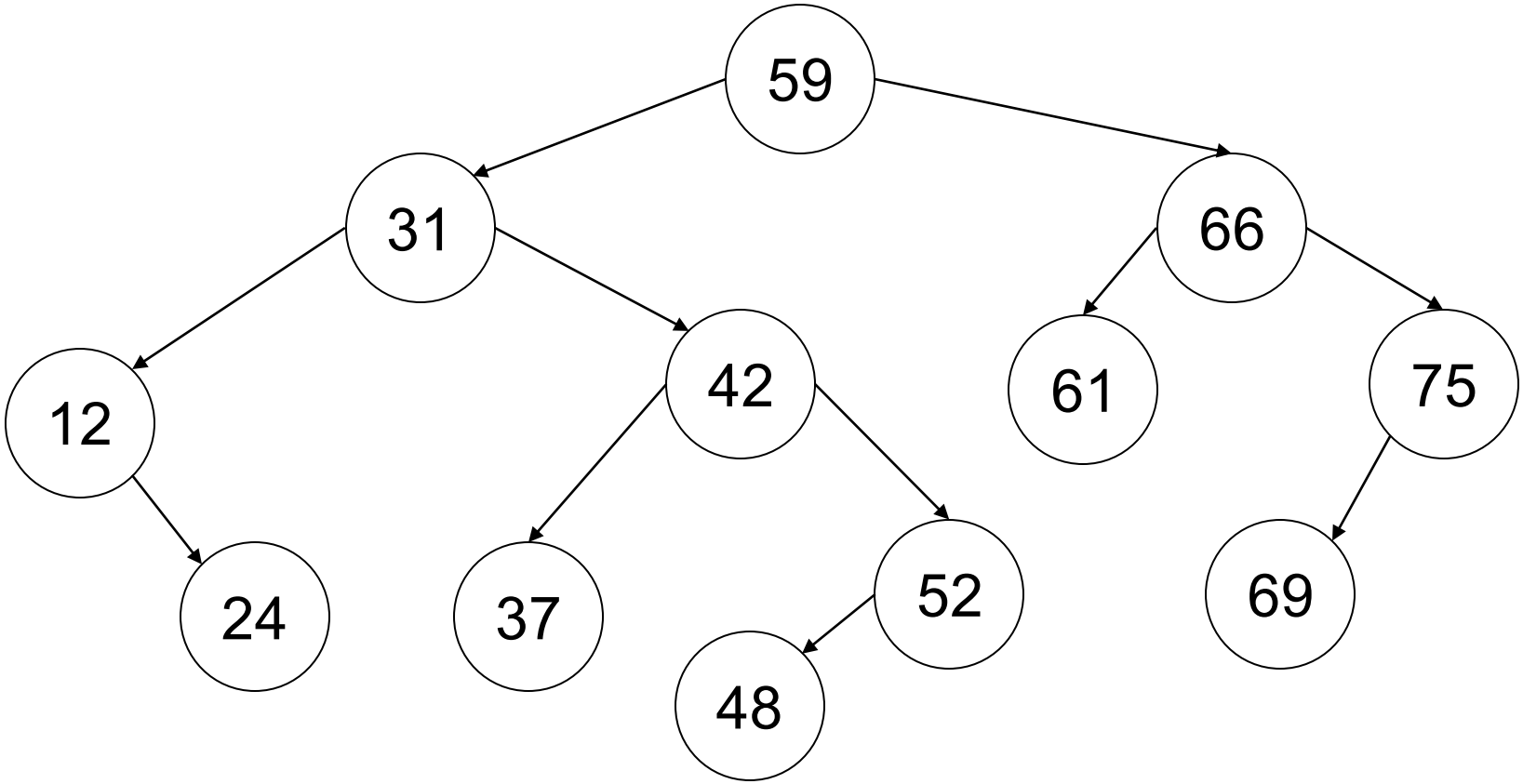
- Updating a sorted file/array is inefficient
- Standard work-around in CS is to replace the sorted file with a search tree
- Next: brief review of binary search trees

Review: Binary Search Tree

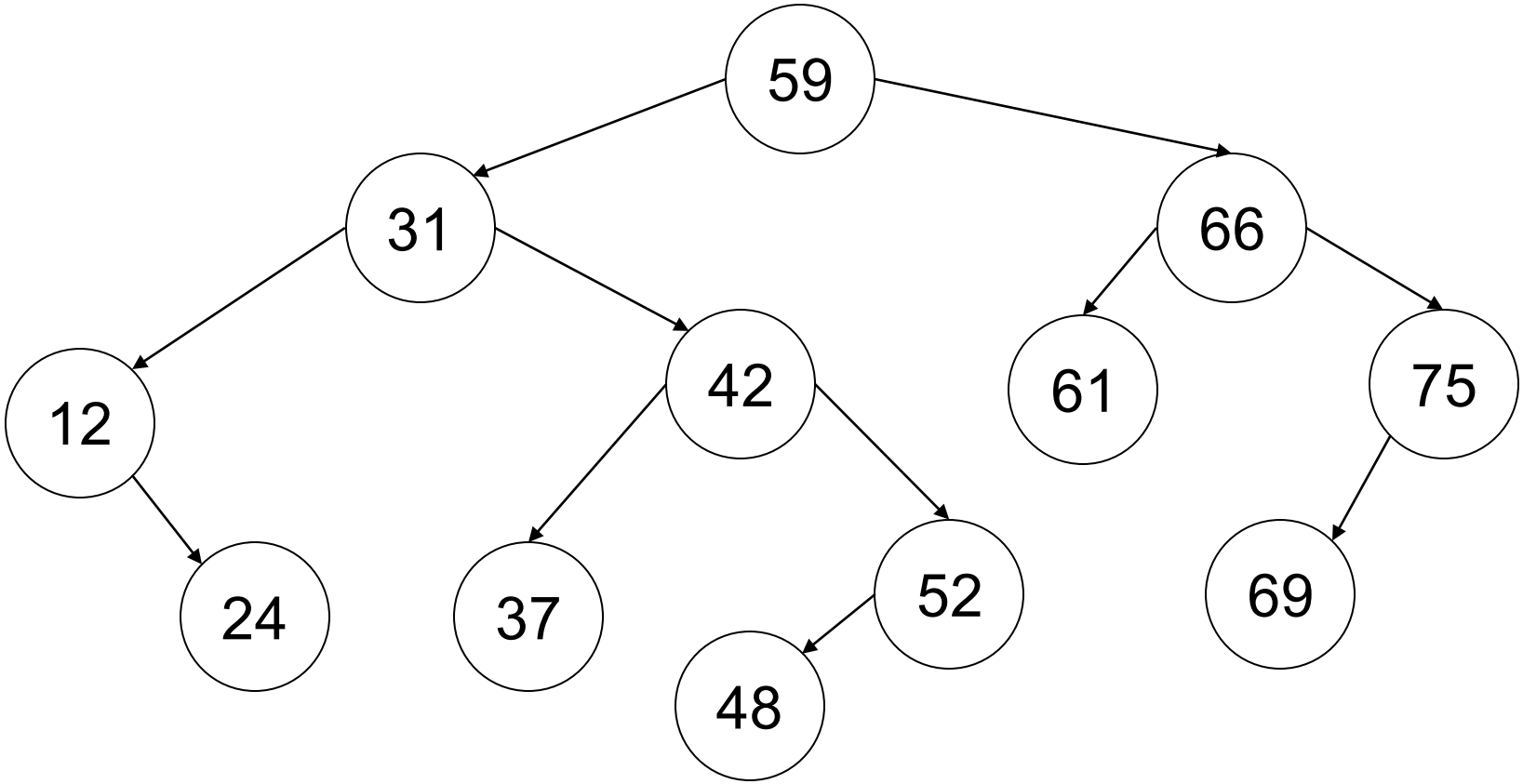
12	24	31	37	42	48	52	59	61	66	69	75

Review: Binary Search Tree

12	24	31	37	42	48	52	59	61	66	69	75

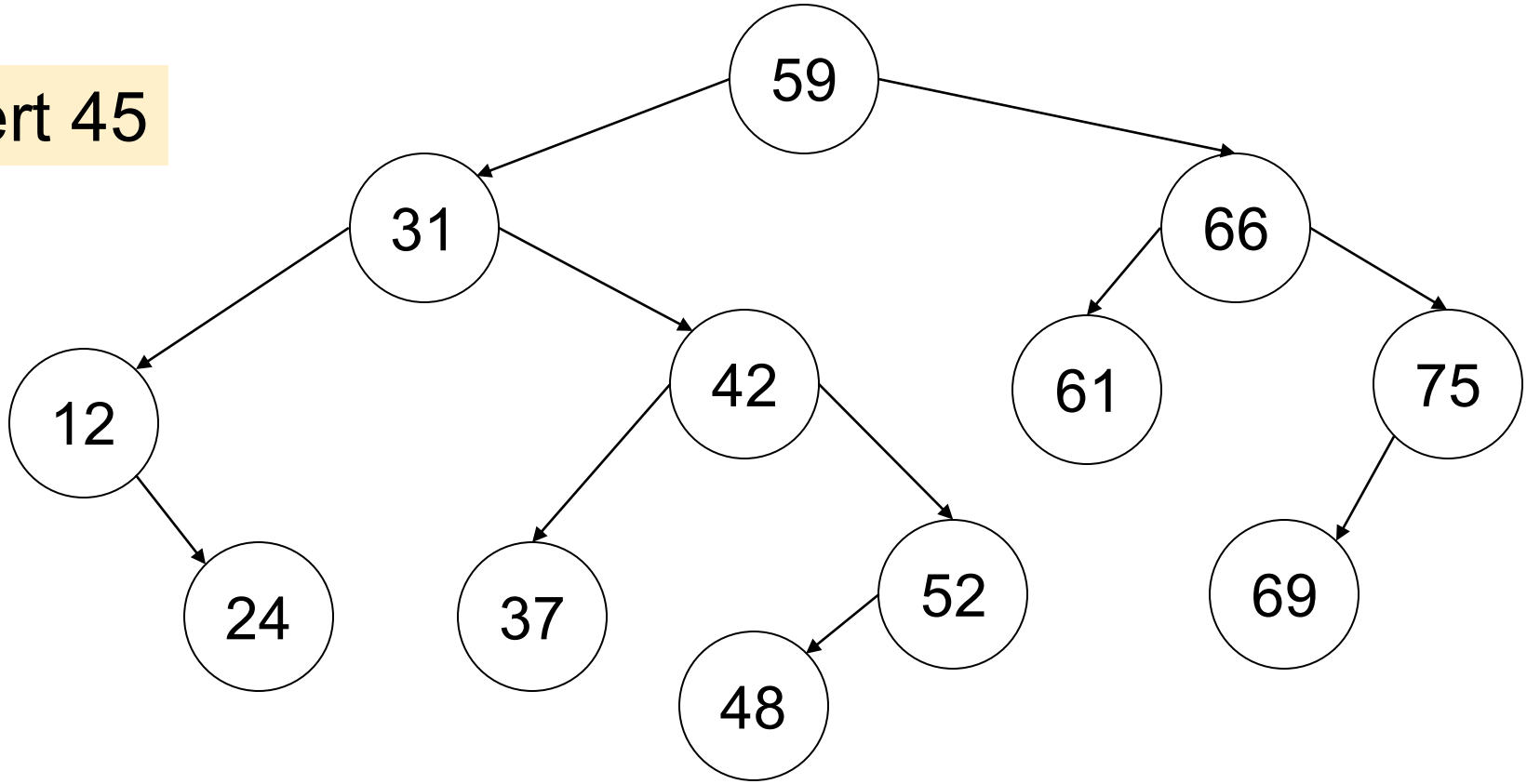


Review: Binary Search Tree



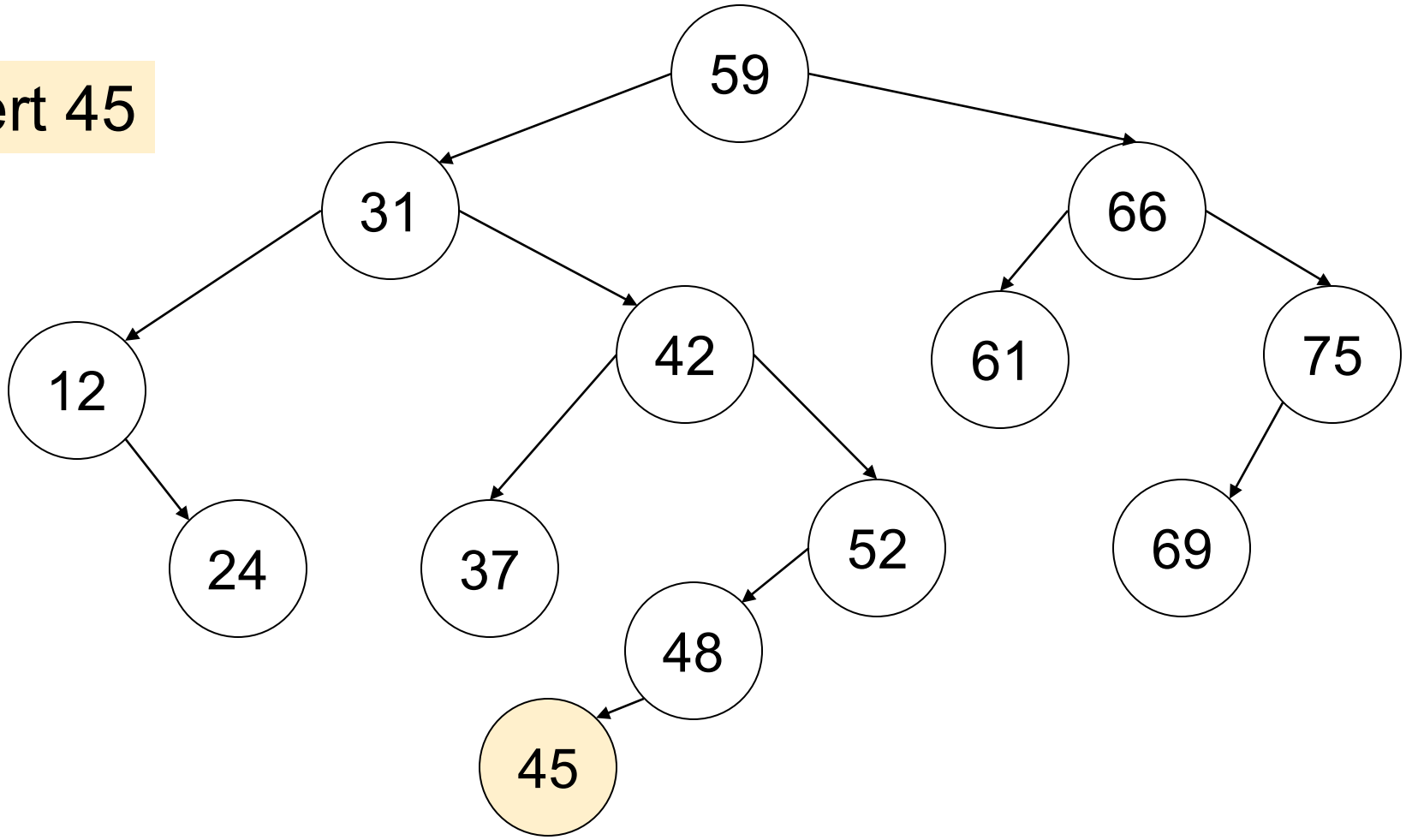
Review: Binary Search Tree

Insert 45



Review: Binary Search Tree

Insert 45

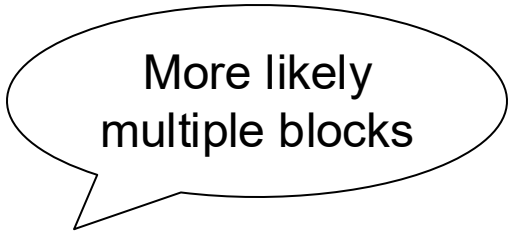


Review: Binary Search Tree

- Need to be balanced: depth=O(log N)
- Various methods to rebalance:
  - Red/black trees, splay trees, …
- Time for search/insert/delete = O(log N)

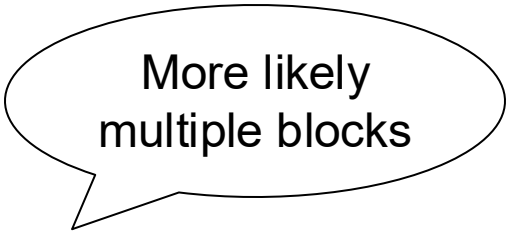
But not good for disk -- why??

B+ Trees



- Idea in B Trees
  - Make 1 node = 1 page (= 1 block)
  - Store multiple keys and children
  - Read 1 page, use many keys

B+ Trees

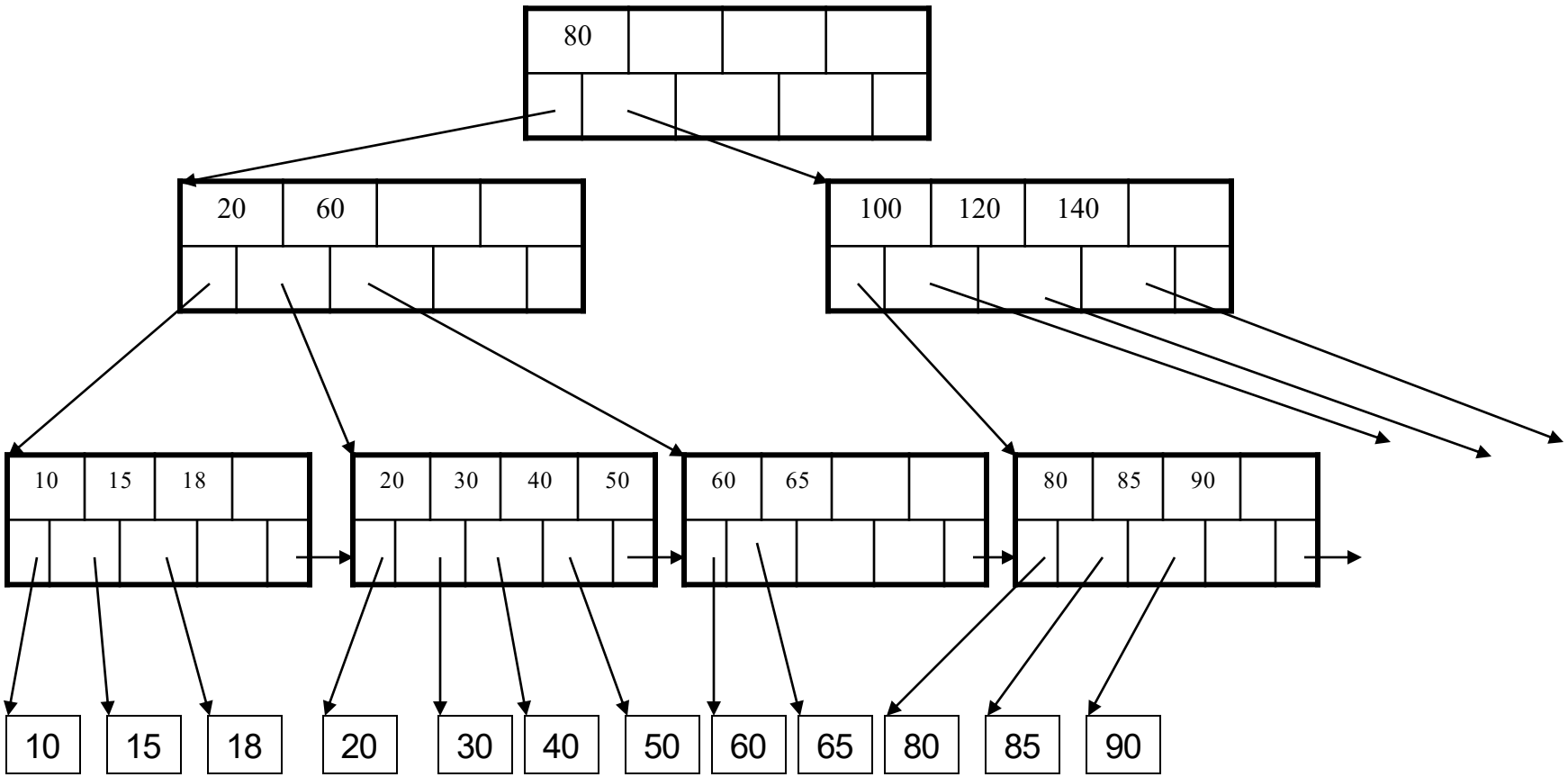


- Idea in B Trees
  - Make 1 node = 1 page (= 1 block)
  - Store multiple keys and children
  - Read 1 page, use many keys

- Idea in B+ Trees
  - Keys are stored only on leaves
  - Internal nodes used only for guiding
  - Leaves are linked in a list, for range queries

B+ Tree Example

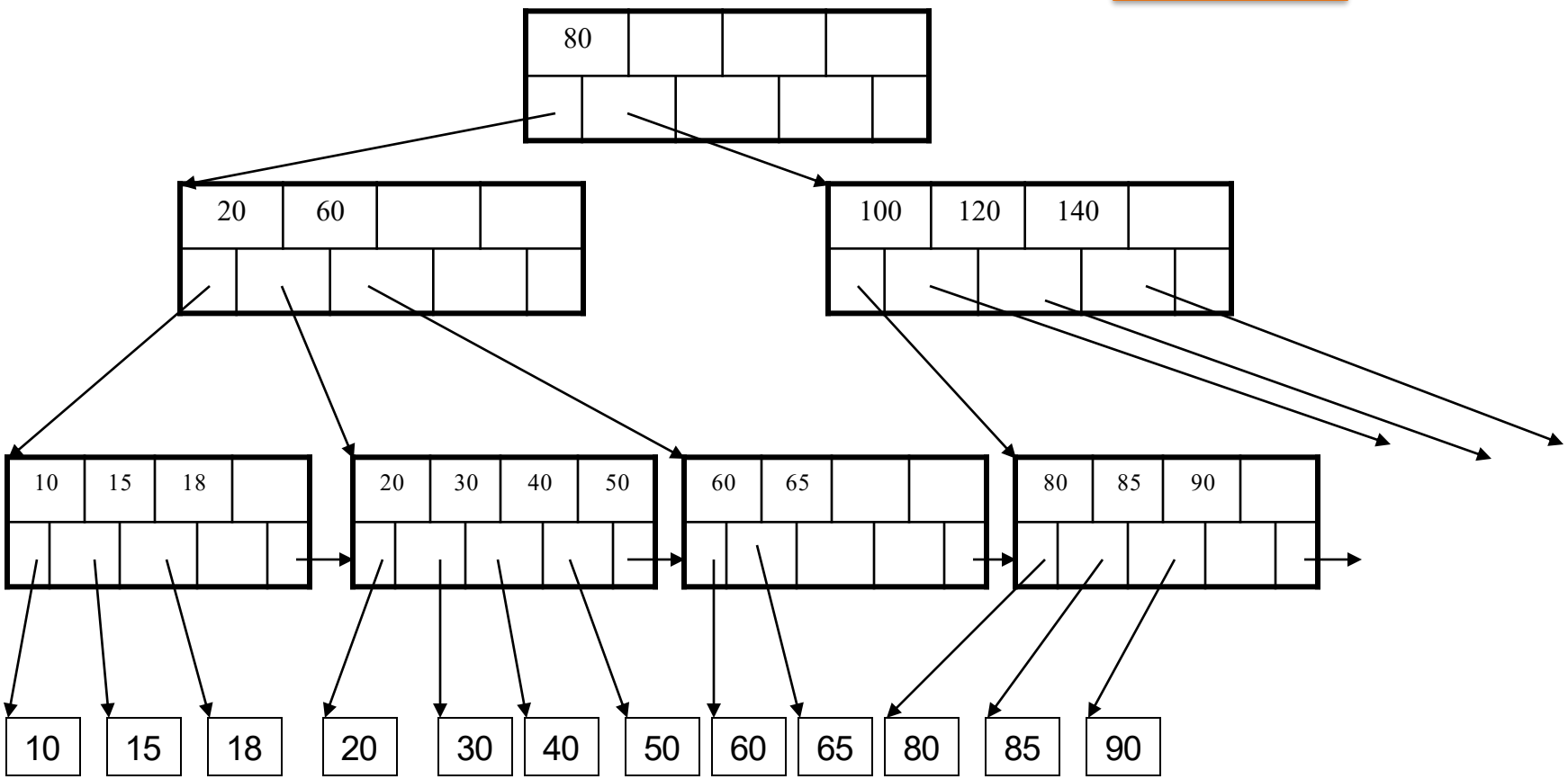
d = 2



B+ Tree Example

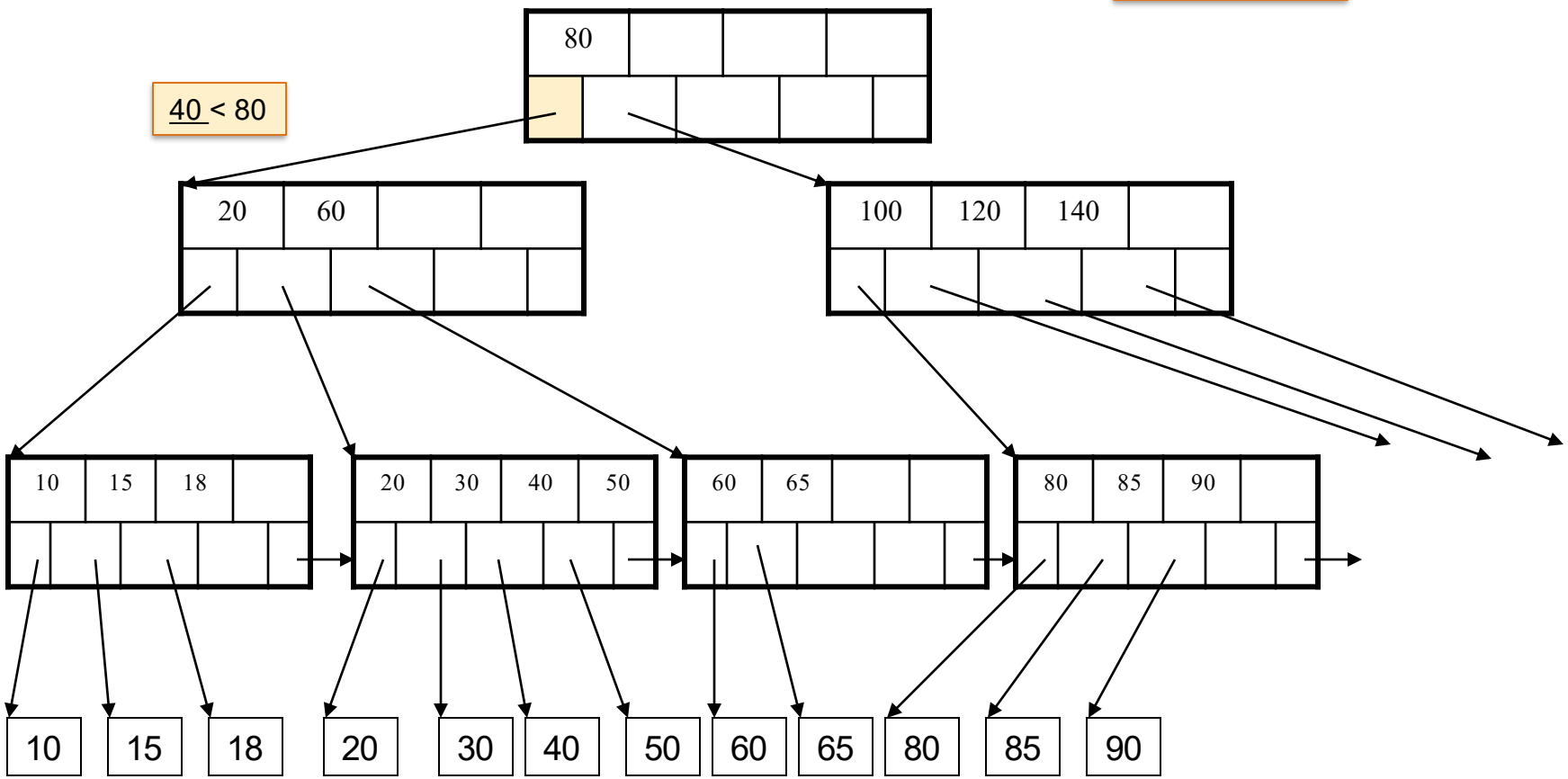
d = 2

Find the key 40



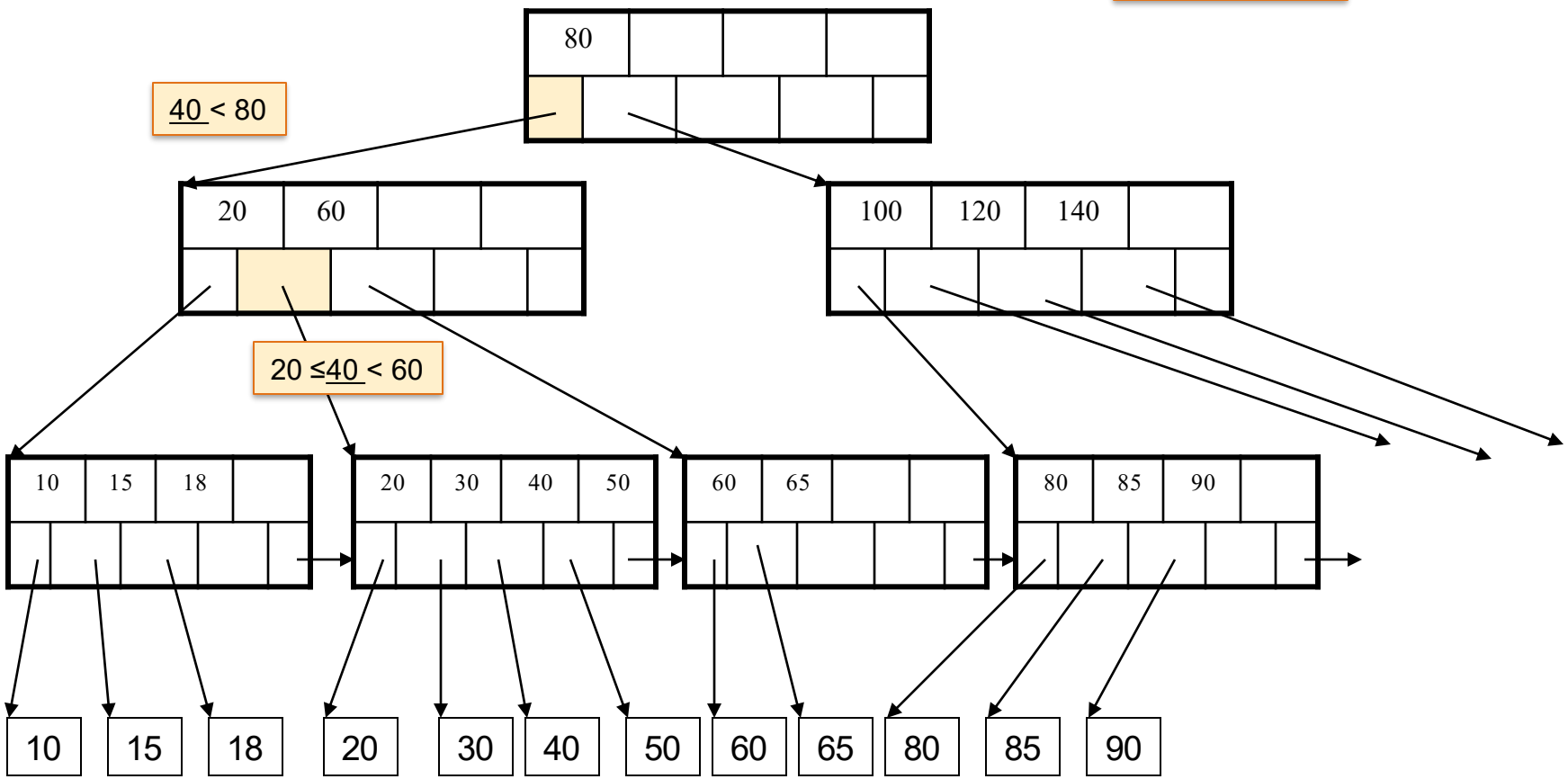
B+ Tree Example

d = 2



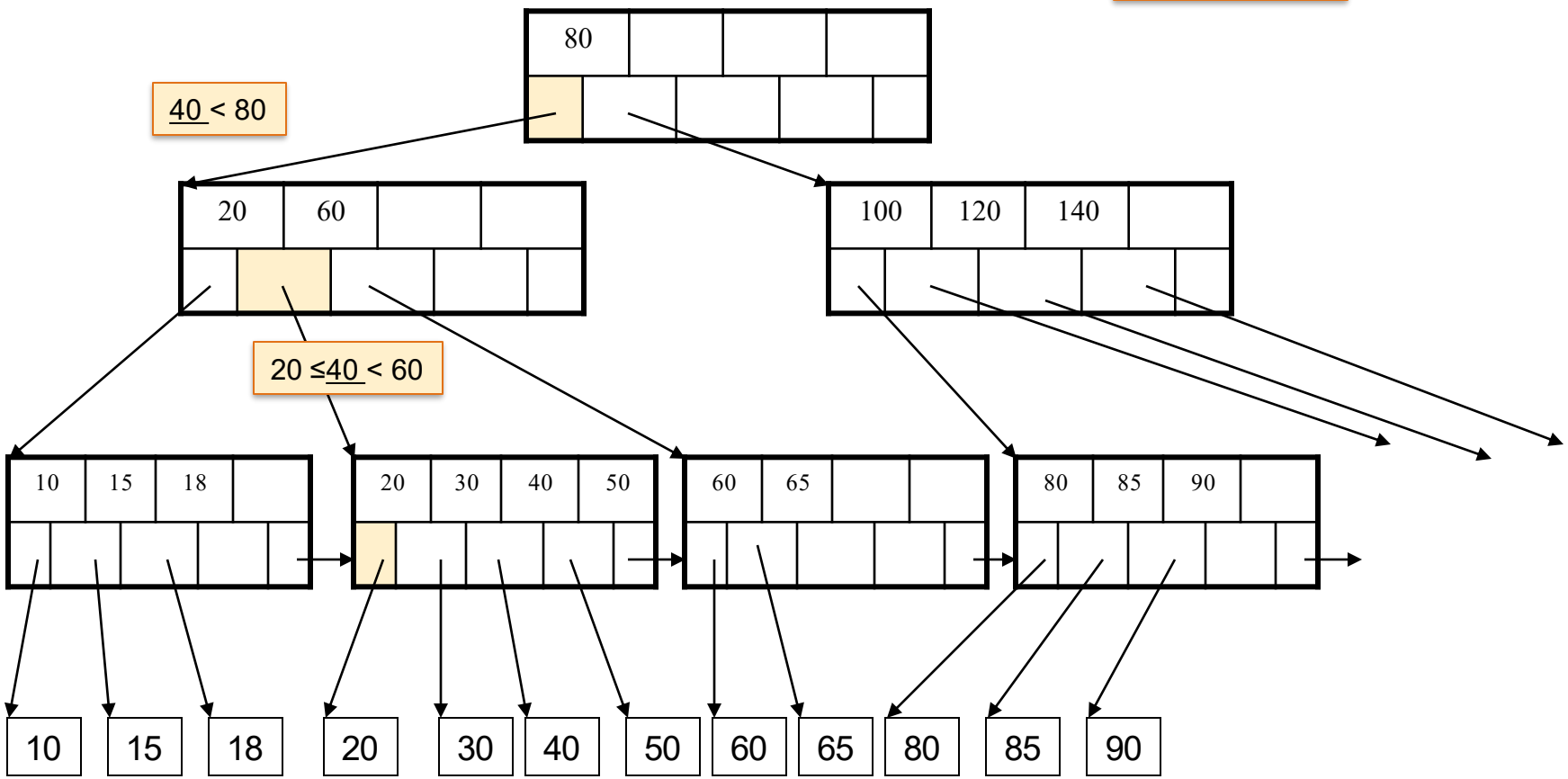
B+ Tree Example

d = 2



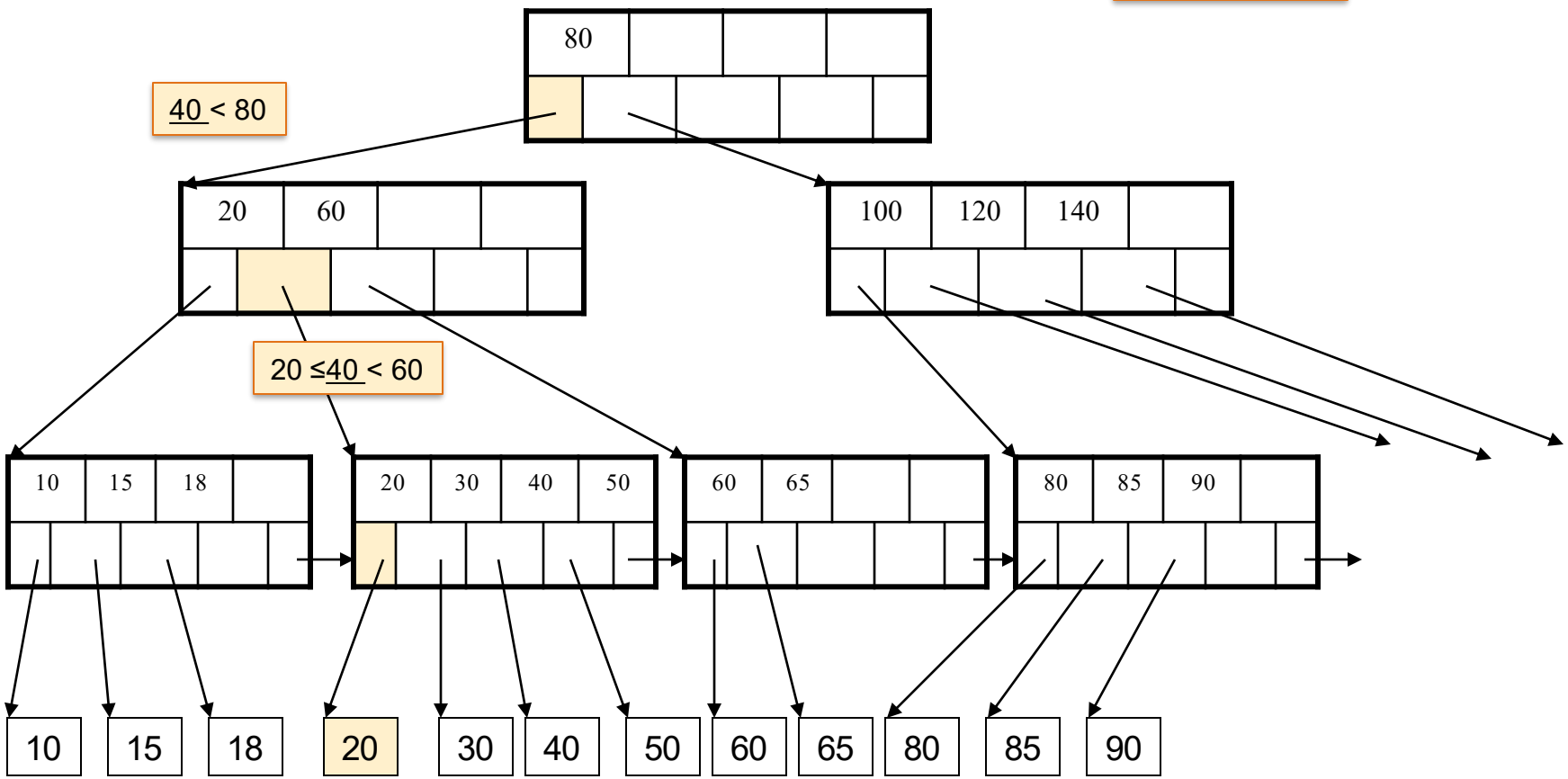
B+ Tree Example

d = 2



B+ Tree Example

d = 2



Recap: File Formats

1. Sequential
2. Sorted
3. Index: will discuss more next lecture

Summary of Storage Manager

- Storage manager and buffer manager are a significant component of DBMS
- Key for good performance
- They also need to handle transactions, which we will not cover in 544