BIG DATA MANAGEMENT

CE/CZ4123

COLUMN STORE PART II

Siqiang Luo

Assistant Professor

IN PREVIOUS LECTURES

- ❑ What is the main design of column stores?
- ❑ Why we need column stores?
- ❑ How to conduct queries with column store?

IN THIS LECTURE

- How to conduct updates with column store?
- More techniques (ideas) about optimizing column stores
  - Compression
  - Shared scans
  - Zone maps
  - Sorting
  - Indexing

STARTING QUESTION

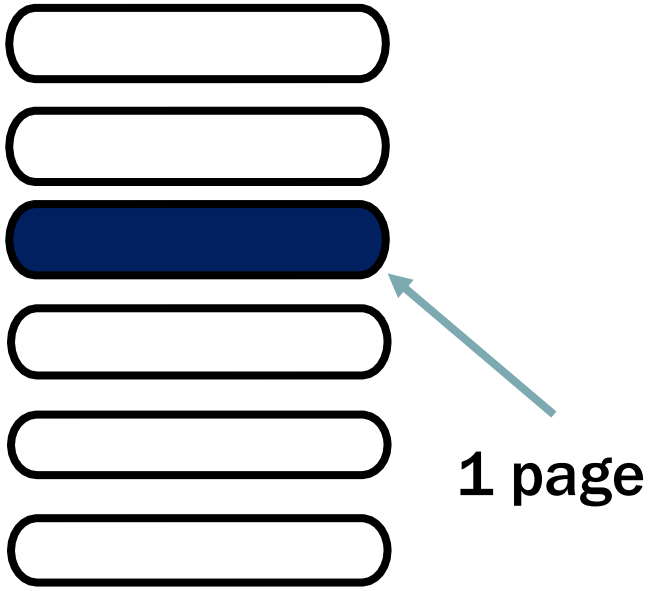
The limitations of column stores?

Updates in column stores

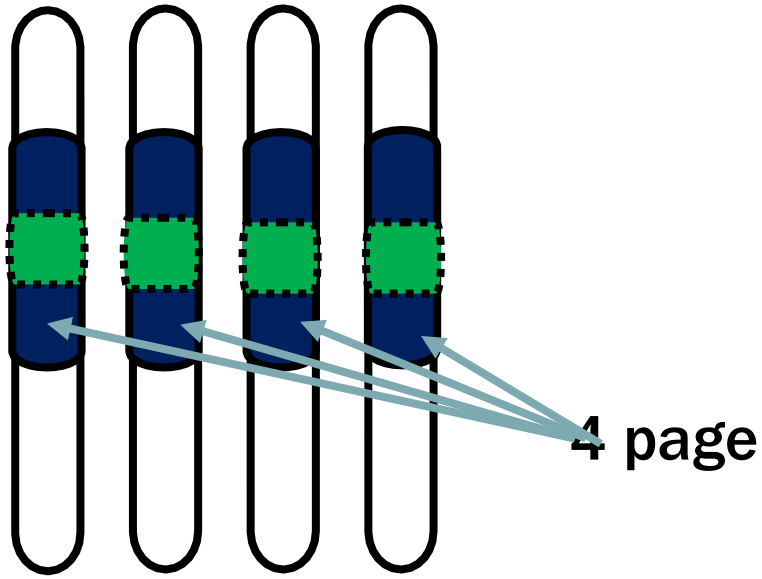
UPDATE WITH COLUMN STORE

- We assume One Row occupies a Page for ease of illustration.

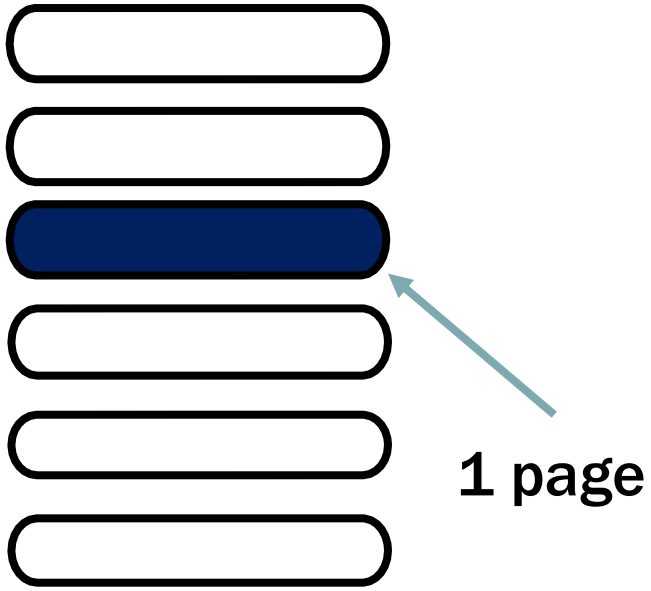
Updating the 3rd row in row store

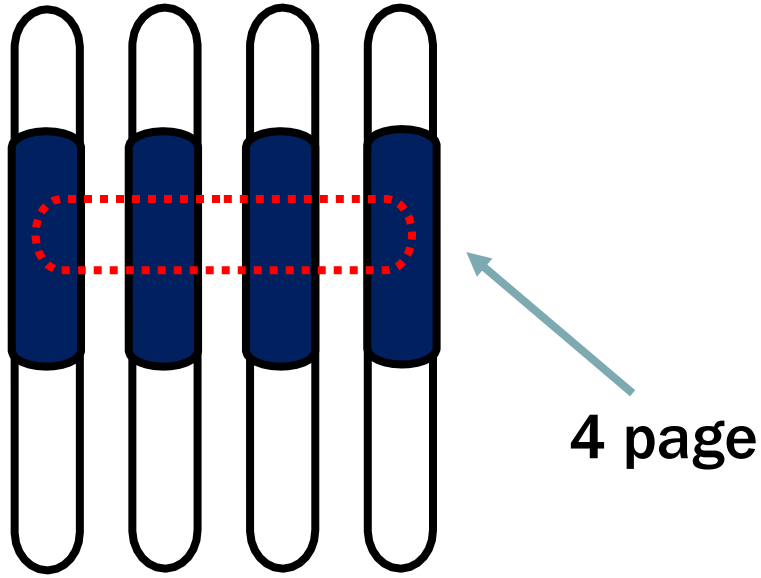


Updating the 3rd row in column store



UPDATE WITH COLUMN STORE





- Row store: when we know which row to update, we only update the page containing the tuple

- Column store: when we know which row to update, we still need to update N pages intersecting with the tuple. (note: N is the number of columns)

One issue with column store

Column store is NOT update-friendly,
if the whole row needs to be updated.

SUMMARY OF COLUMN STORE

- Column store is often great for read-heavy workloads (e.g., analytical tasks)
- Row store is often great for updates, but column store's update performance is also improving with latest technologies.
- In system design, **no** system is perfect for **every kind of workload** (e.g., analytical workloads, update-heavy workloads). Always design your data system based on the targeted workloads.

Join in column stores

```
SELECT SUM(R.a)
FROM R,S
WHERE R.c=S.b and R.a<5 and
R.b>3 and S.a>3
```

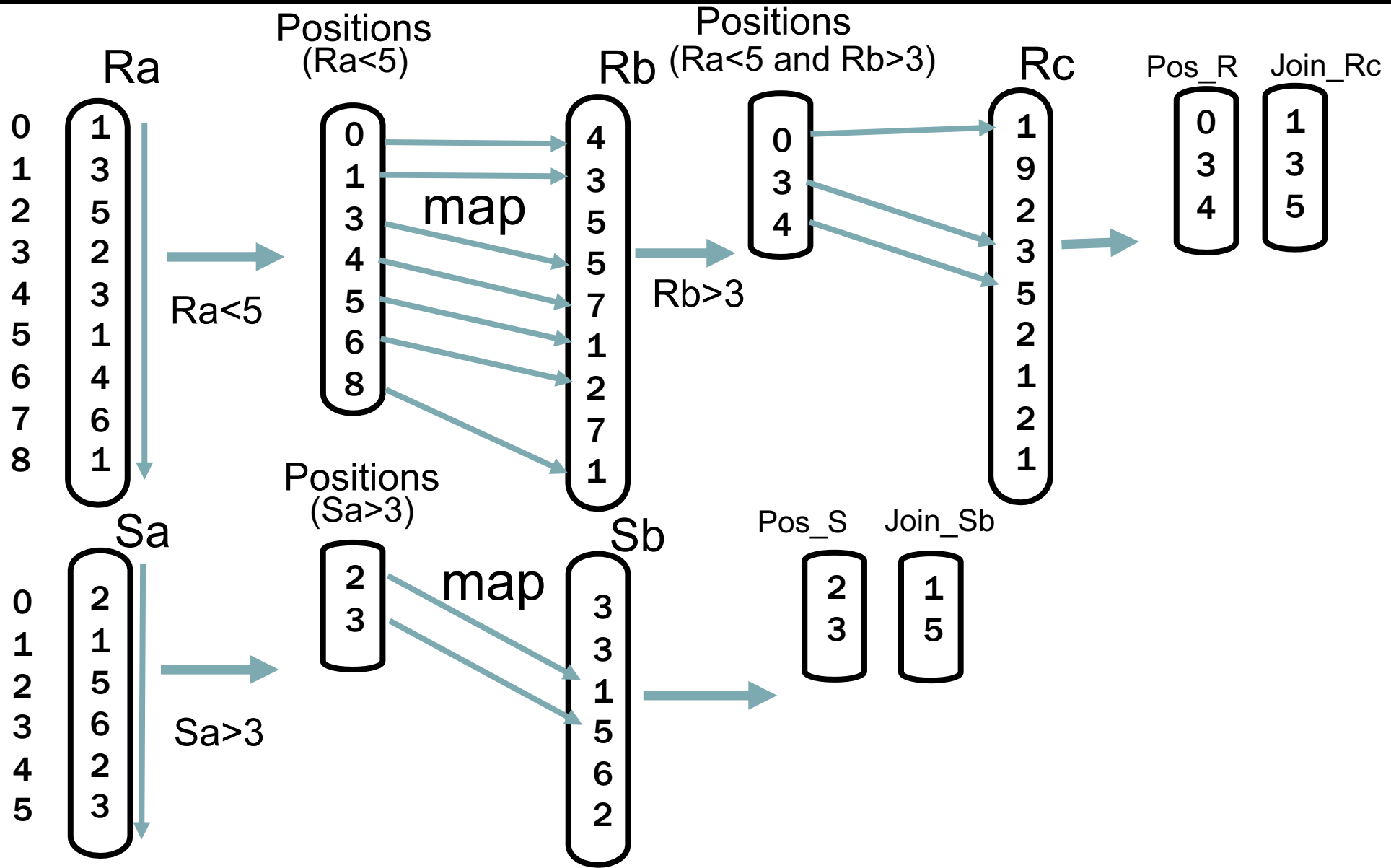
Table R

positions	Ra	Rb	Rc
0	1	4	1
1	3	3	9
2	5	5	2
3	2	5	3
4	3	7	5
5	1	1	2
6	4	2	1
7	6	7	2
8	1	1	1

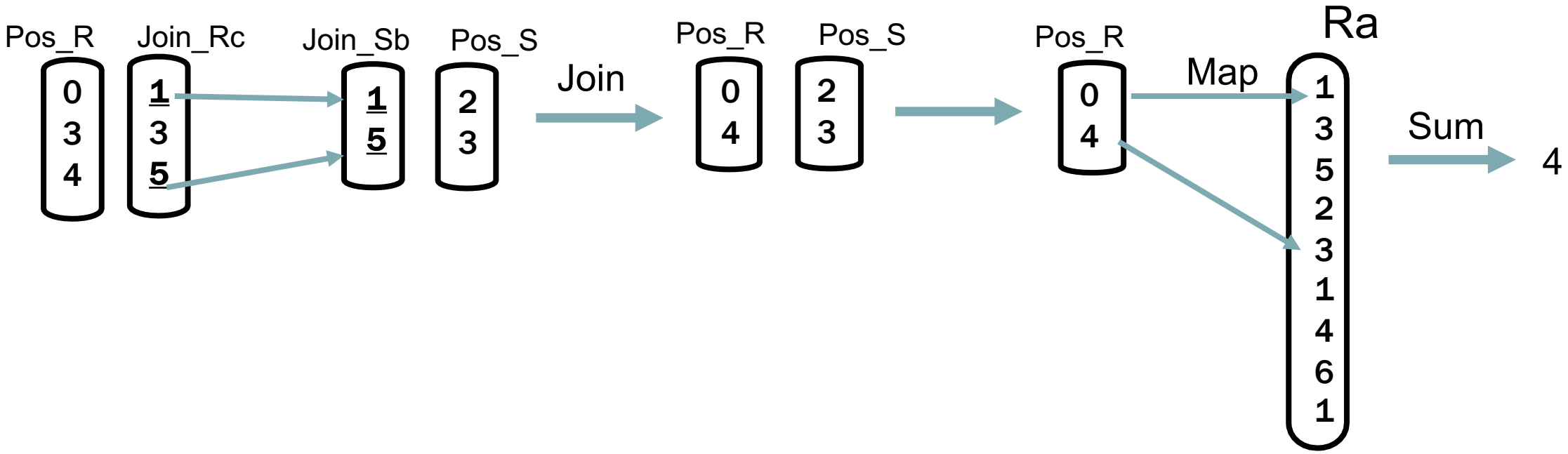
Table S

positions	Sa	Sb
0	2	3
1	1	3
2	5	1
3	6	5
4	2	6
5	3	2

COLUMN-BASED FILTERING



JOIN & AGGREGATE



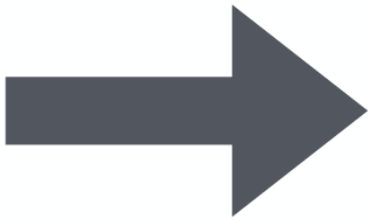
More techniques to enhance column stores

COMPRESSION

Original data

8 bytes
width

value1
value2
value3
value1
value1
value4
value2
value3
value5
...



Compressed

3 bits
width

001
010
011
001
001
100
010
011
101
...

Dictionary

8 bytes
width

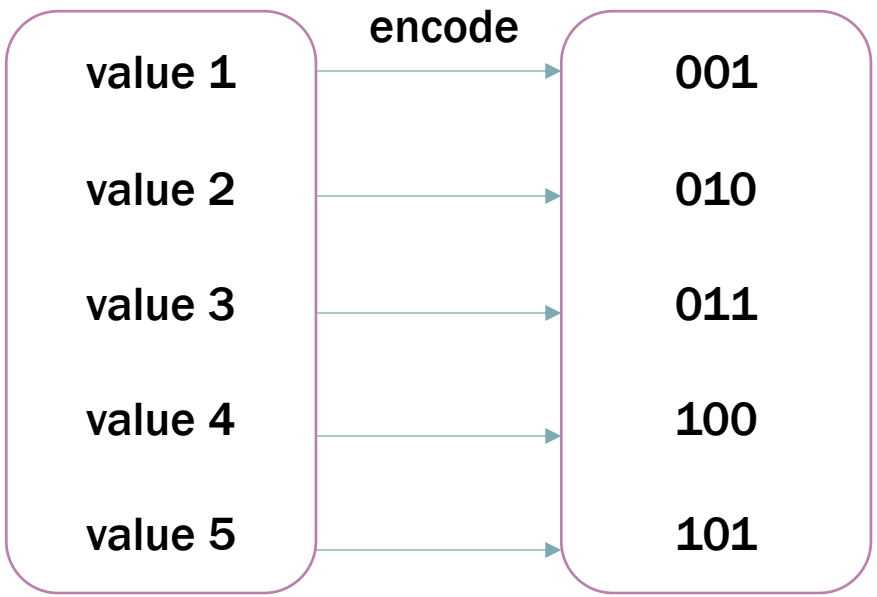
value1
value2
value3
value4
value5

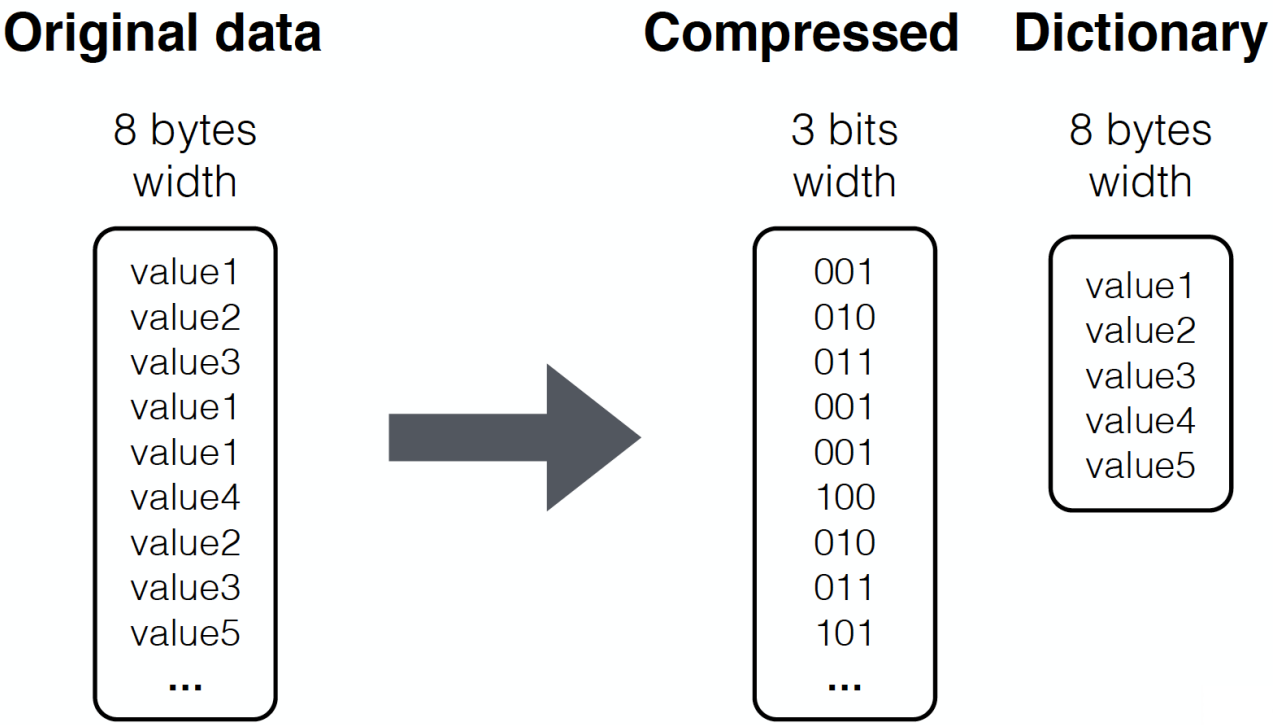
If there are only 5 possible values in a column, then 3 bits encoding can be applied.

How many bits?

COMPRESSION

If there are only 5 possible values in a column, then 3 bits encoding can be applied.





Compressed ~21x

COMPRESSION

Compressed

3 bits
width

001
010
011
001
001
100
010
011
101
...

Dictionary

8 bytes
width

value1
value2
value3
value4
value5

How do we process data?

Check “=100”?

Check “<100”?

COMPRESSION

In general, if there are N possible values in the column, we can use log N bits to encode (compress) the column values

Comparison of data can be operated on the compressed data
For “= 100”, we can process the data by:

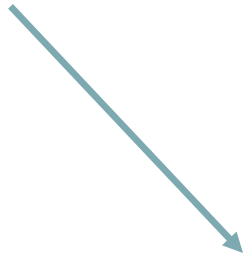
(1) find the compressed code for 100 from the dictionary; if not found, then must be “unequal”
(2) Otherwise compare with the code directly in the compressed column

COMPRESSION

In general, if there are N possible values in the column, we can use log N bits to encode (compress) the column values

Comparison of data can be operated on the compressed data

For “<100”, we can apply order-preserving encoding.



value 1 < value 2 is equal to code 1 < code 2

SHARED SCANS

loop fusion

```
for(i=0;i<n;i++)
    min = a[i]<min ? a[i] : min

for(i=0;i<n;i++)
    max = a[i]>max ? a[i] : max
```

```
for(i=0;i<n;i++)
    min = a[i]<min ? a[i] : min
    max = a[i]>max ? a[i] : max
```

Which one is better?

SHARED SCANS

loop fusion

```
for(i=0;i<n;i++)
    min = a[i]<min ? a[i] : min

for(i=0;i<n;i++)
    max = a[i]>max ? a[i] : max
```

```
for(i=0;i<n;i++)
    min = a[i]<min ? a[i] : min
    max = a[i]>max ? a[i] : max
```

Two passes of data

One pass of data

SHARED SCAN

Real-case code in C++

```
7   # define LENGTH 500000000
8   int array[LENGTH];
9   int main(int argc, char* argv[]){
10      for(int i = 0; i < LENGTH; i++){
11          array[i] = rand();
12      }
13      int min, max;
14      struct timespec t1, t2;
```

Two Pass Solution

```
16      clock_gettime(CLOCK_MONOTONIC, &t1);
17      min = RAND_MAX;
18      max = 0;
19      for(int i = 0; i < LENGTH; i++){
20          min = array[i]<min?array[i]:min;
21      }
22      for(int i = 0; i < LENGTH; i++){
23          max = array[i]>max?array[i]:max;
24      }
25      clock_gettime(CLOCK_MONOTONIC, &t2);
26      double time = (t2.tv_sec - t1.tv_sec) +
27          (double)(t2.tv_nsec - t1.tv_nsec) / 1000000000;
28      std::cout << "Two pass time: " << time << " s" << std::endl;
```

One Pass Solution

```
30      clock_gettime(CLOCK_MONOTONIC, &t1);
31      min = RAND_MAX;
32      max = 0;
33      for(int i = 0; i < LENGTH; i++){
34          min = array[i]<min?array[i]:min;
35          max = array[i]>max?array[i]:max;
36      }
37      clock_gettime(CLOCK_MONOTONIC, &t2);
38      time = (t2.tv_sec - t1.tv_sec) +
39          (double)(t2.tv_nsec - t1.tv_nsec) / 1000000000;
40      std::cout << "One pass time: " << time << " s" << std::endl;
```

SHARED SCAN

Real-case code in C++

```
7   # define LENGTH 500000000
8   int array[LENGTH];
9   int main(int argc, char* argv[]){
10      for(int i = 0; i < LENGTH; i++){
11          array[i] = rand();
12      }
13      int min, max;
14      struct timespec t1, t2;
```

Execution result

```
Two pass time: 2.20189 s
One pass time: 1.34021 s
```

Two Pass Solution

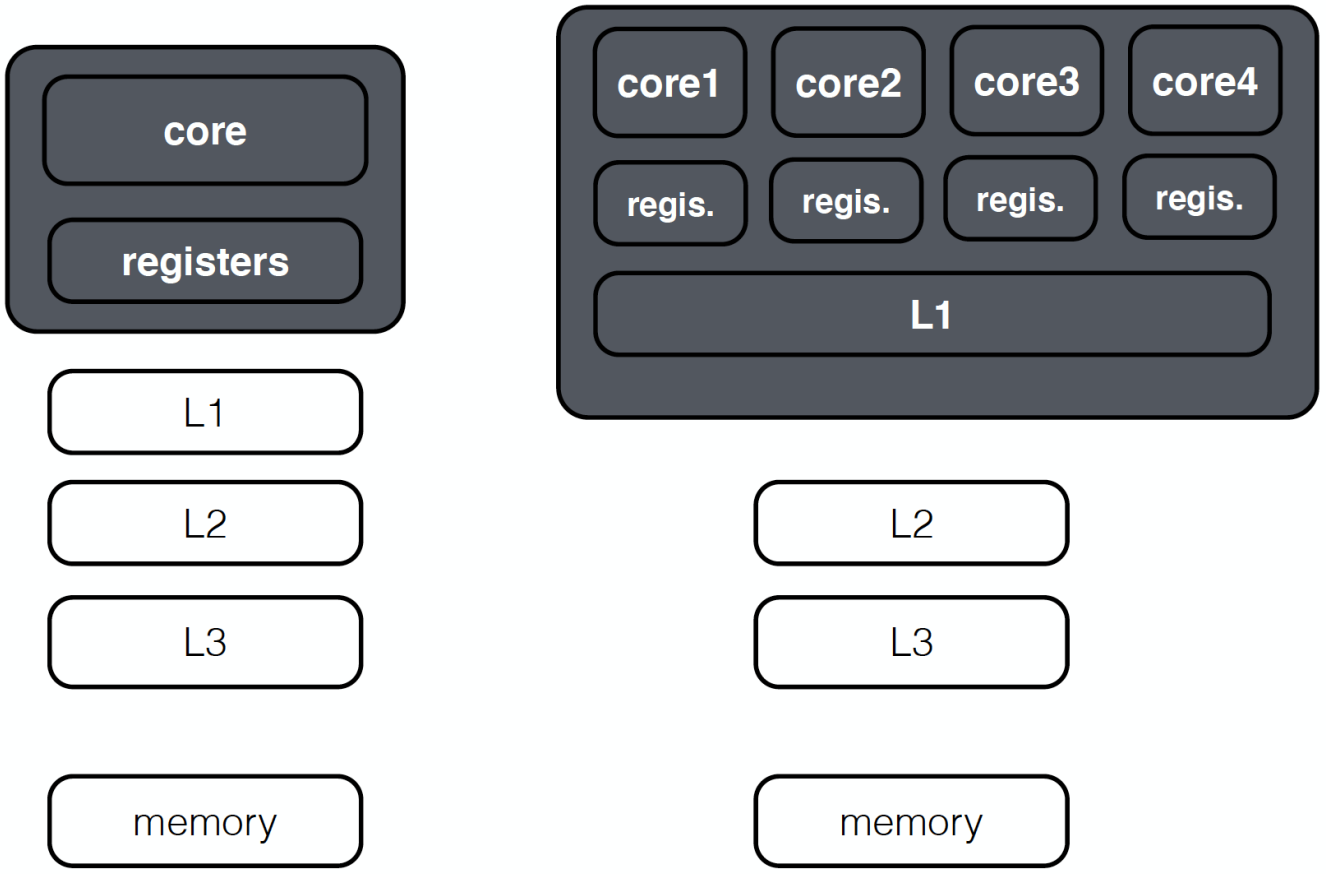
```
16      clock_gettime(CLOCK_MONOTONIC, &t1);
17      min = RAND_MAX;
18      max = 0;
19      for(int i = 0; i < LENGTH; i++){
20          min = array[i]<min?array[i]:min;
21      }
22      for(int i = 0; i < LENGTH; i++){
23          max = array[i]>max?array[i]:max;
24      }
25      clock_gettime(CLOCK_MONOTONIC, &t2);
26      double time = (t2.tv_sec - t1.tv_sec) +
27          (double)(t2.tv_nsec - t1.tv_nsec) / 1000000000;
28      std::cout << "Two pass time: " << time << " s" << std::endl;
```

One Pass Solution

```
30      clock_gettime(CLOCK_MONOTONIC, &t1);
31      min = RAND_MAX;
32      max = 0;
33      for(int i = 0; i < LENGTH; i++){
34          min = array[i]<min?array[i]:min;
35          max = array[i]>max?array[i]:max;
36      }
37      clock_gettime(CLOCK_MONOTONIC, &t2);
38      time = (t2.tv_sec - t1.tv_sec) +
39          (double)(t2.tv_nsec - t1.tv_nsec) / 1000000000;
40      std::cout << "One pass time: " << time << " s" << std::endl;
```

SHARED SCANS

Modern CPUs can do more than 1 tasks at the same time.



SHARED SCANS

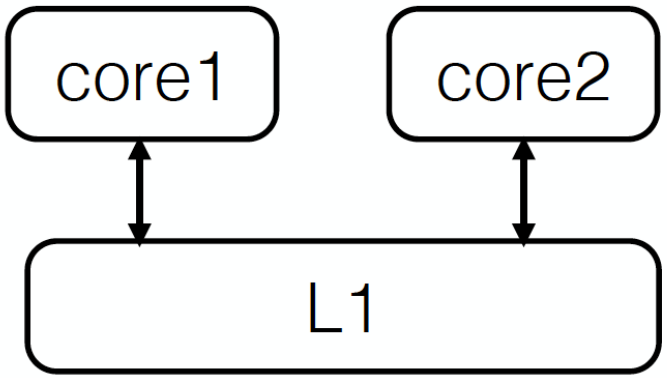
can work in parallel

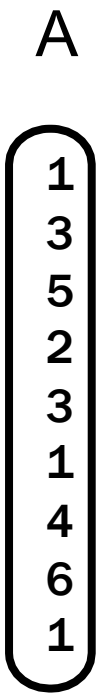
SHARED SCANS

In column store, multiple queries scanning the same column.

We can always maximize the CPU core usage and minimize the data scanned.
This idea is called shared scan.

Query 1: Select > 5 (handled by core 1)
Query 2: Select < 8 (handled by core 2)



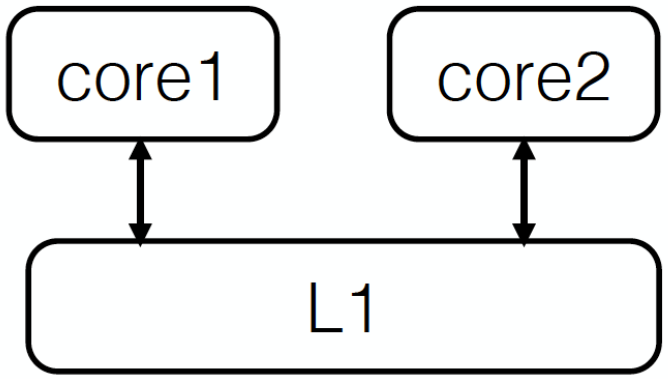


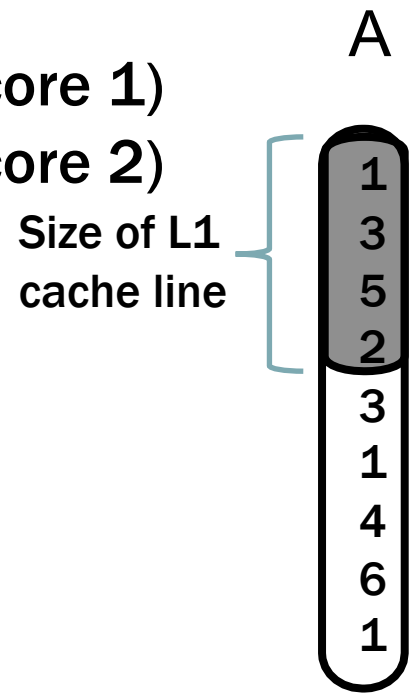
SHARED SCANS

In column store, multiple queries scanning the same column.

We can always maximize the CPU core usage and minimizing the scanning data. This idea is called shared scan.

Query 1: Select > 5 (handled by core 1)
Query 2: Select < 8 (handled by core 2)

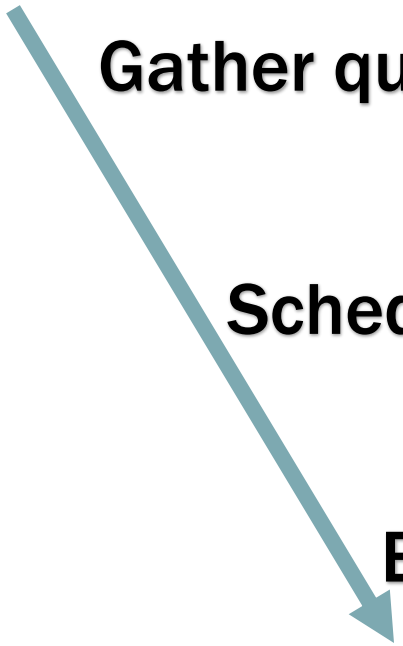




Detailed steps:
1. Sequentially loads the data into L1 cache lines.
2. Core 1 extracts the values > 5, while Core 2 extracts the values < 8

QUESTION

What if queries do not come at the same time?



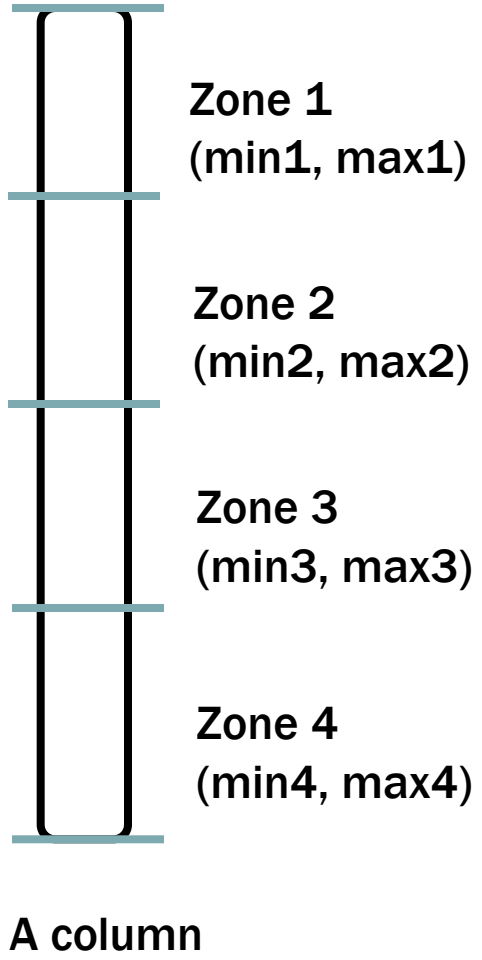
Gather queries in a batch

Schedule the queries on same data to run in parallel

Each query gets a thread/core

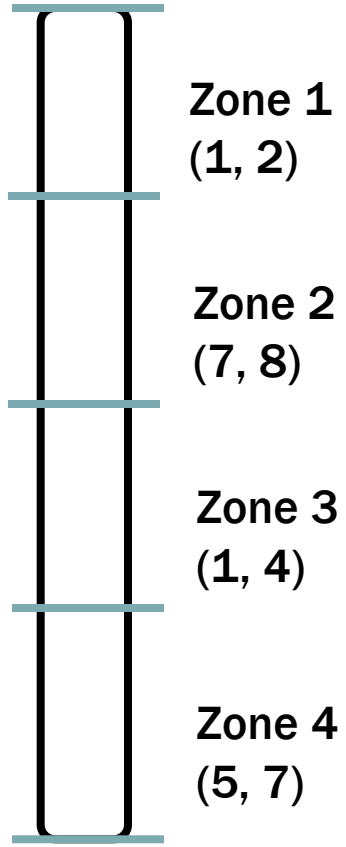
ZONE MAP

- A common way to help column scan is the zone map.
- It separates a column into “zones”, each is computed with max and min
- In filtering, some zones can be skipped.



ZONE MAP

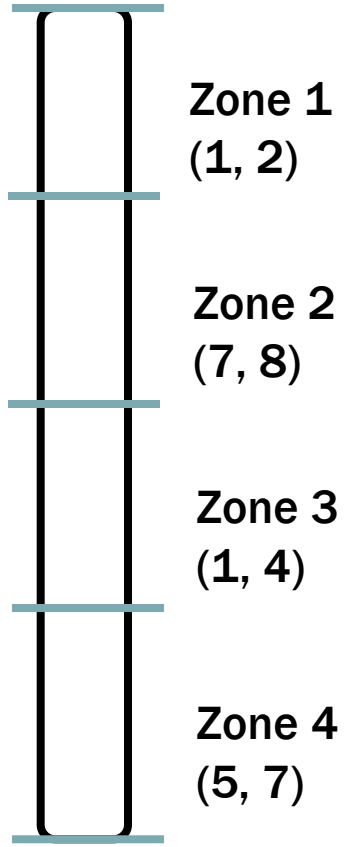
- Select from T where X>3 and X<6
- Which zones need to be scanned?



A column X

ZONE MAP

- Select from T where X>3 and X<6
- Which zones need to be scanned?
- We only need to scan Zone3 and Zone4 because
- [1, 4] overlaps with (3,6)
- [5, 7] overlaps with (3,6)
- [1, 2] does not overlap with (3,6)
- [7, 8] does not overlap with (3,6)

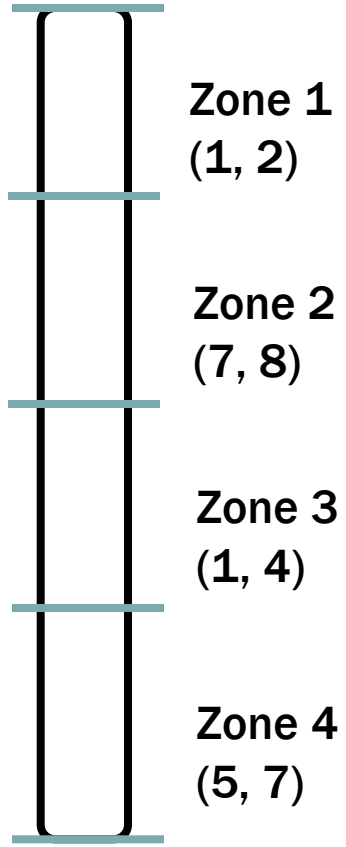


A column X

ZONE MAP

- Select from T where X>3 and X<6
- Which zones need to be scanned?
- We only need to scan Zone3 and Zone4 because
- [1, 4] overlaps with (3,6)
- [5, 7] overlaps with (3,6)
- [1, 2] does not overlap with (3,6)
- [7, 8] does not overlap with (3,6)

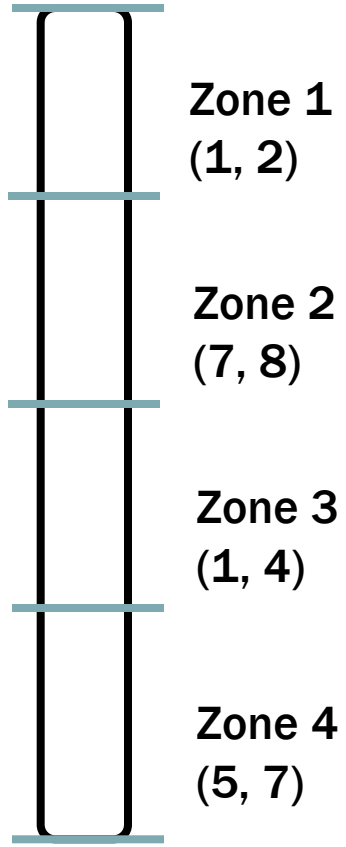
- Usually, a zone is of a page size
- We skip the scanning of two pages



A column X

ZONE MAP

- How to store min and max from each zones?
- They are stored together, probably in a few disk pages.
- These zone map pages can be loaded into memory when system is started
- So the additional cost of reading zone maps is very low



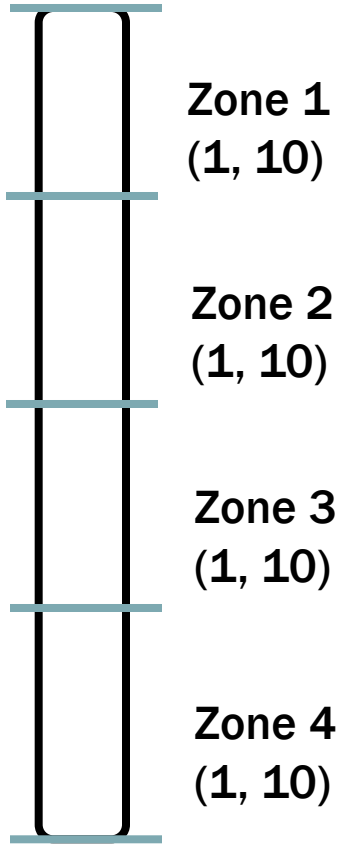
A column X

ZONE MAP

- Is zone map always effective? Why?

ZONE MAP

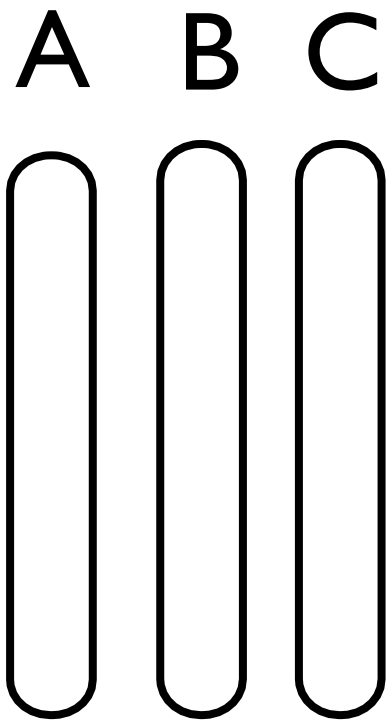
- Is zone map always effective? Why?
- Data can be uniformly distributed.



A column X

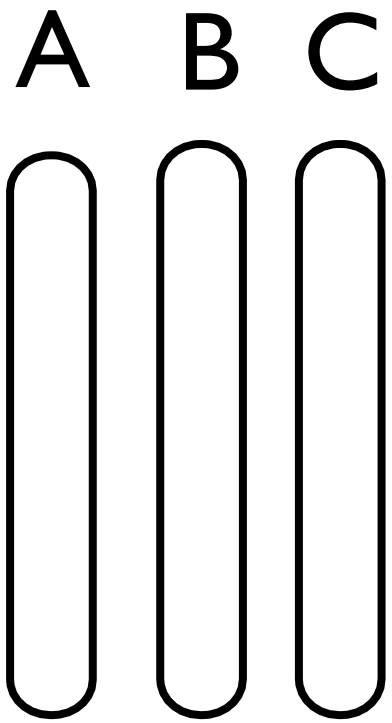
ENHANCED WITH SORTING

SELECT max(C) FROM T WHERE A>10 and A<40 and B>20



ENHANCED WITH SORTING

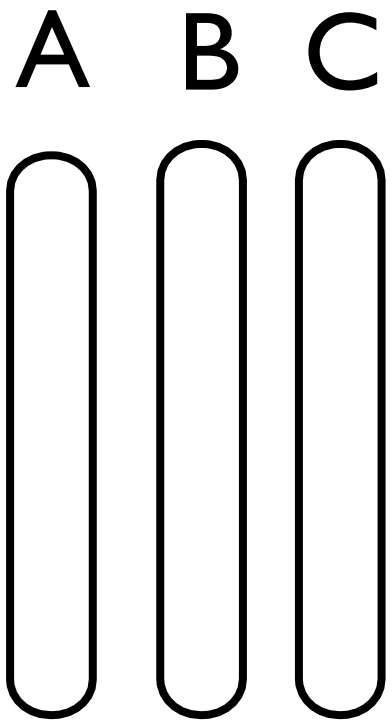
SELECT max(C) FROM T WHERE A>10 and A<40 and B>20



1) Scan A and select A in (10, 40)
2) Among those positions returned in 1), scan B and select B>20
3) Among those positions returned in 2), select max(C)

ENHANCED WITH SORTING

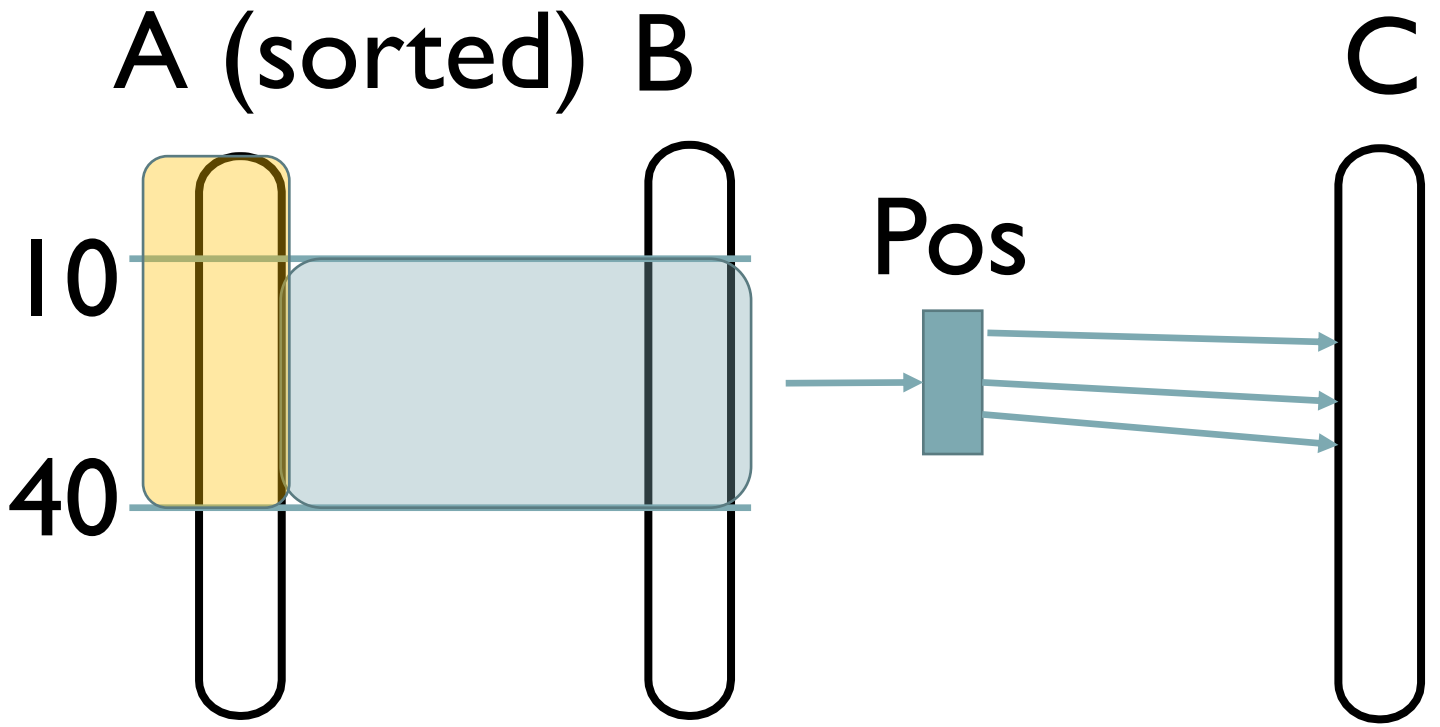
SELECT max(C) FROM T WHERE A>10 and A<40 and B>20



What if A is sorted?

ENHANCED WITH SORTING

SELECT max(C) FROM T WHERE A>10 and A<40 and B>20



First scan A up to 40 (yellow)

Then scan B in blue range and filter for B>20

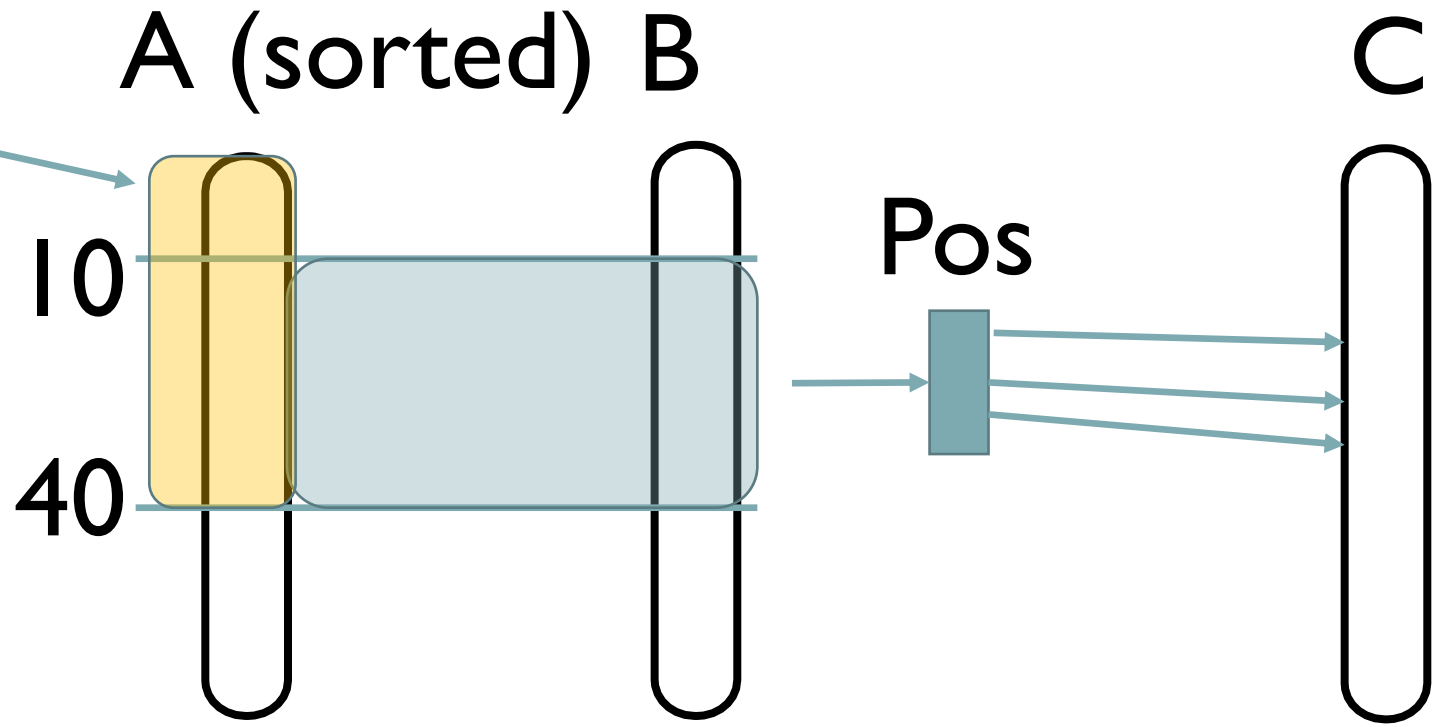
Store positions for tuples having A>10, A<40 and B>20

Select max(C) based on the positions

ENHANCED WITH SORTING

SELECT max(C) FROM T WHERE A>10 and A<40 and B>20

Optimization: No need to scan from the beginning.



First scan A up to 40 (yellow)

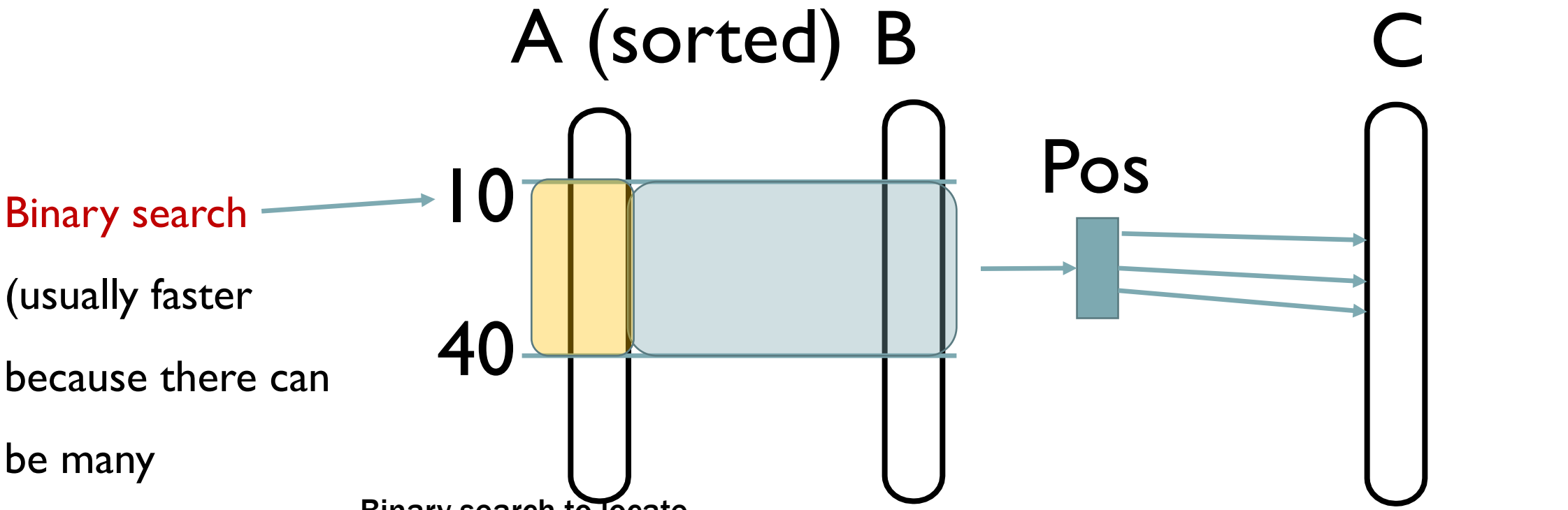
Then scan B in blue range and filter for B>20

Store positions for tuples having A>10, A<40 and B>20

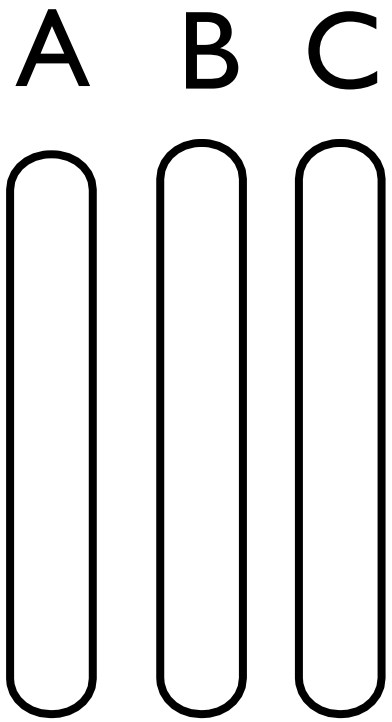
Select max(C) based on the positions

ENHANCED WITH SORTING

SELECT max(C) FROM T WHERE A>10 and A<40 and B>20

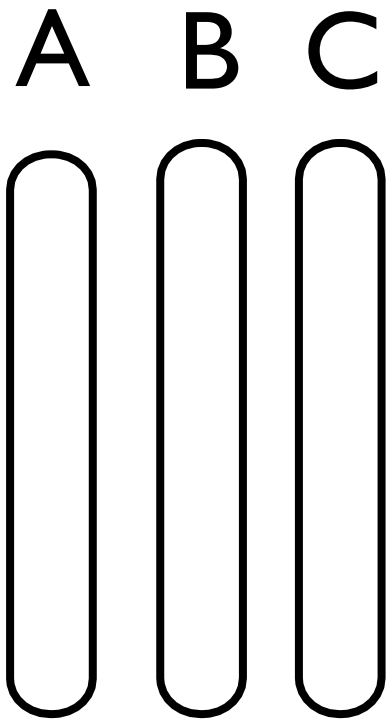


QUESTIONS FOR YOU



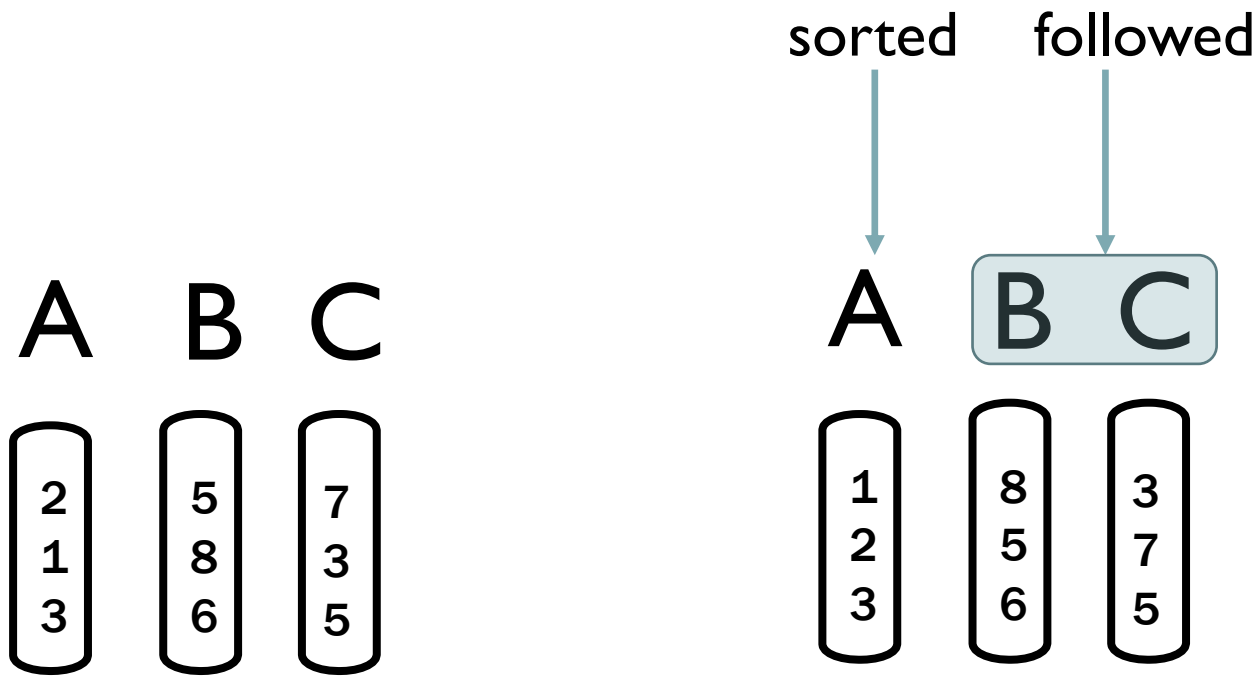
When sorting A,
do the other columns change?

QUESTIONS FOR YOU



To apply the above query scheme, the other columns need to be rearranged to follow the order of A

QUESTIONS FOR YOU



LIMITATION OF SORTING

- Can only sort one column (because the other columns need to follow the order of the sorted column)
- Filtering based on unsorted column is still as costly.
- Better solution: use index

ENHANCED WITH INDEX

What is an index?

ENHANCED WITH INDEX

What is an index?

Roughly speaking, in column store, an index is an affiliated data structure built on a column, and it can efficiently help locate the position of a value in the column. It is a widely used technique in data management.

ENHANCED WITH INDEX

Why do we need index?

Imaging how do you find a book in the library

Business Library
Available , Level B1: A-HG; Level B4: HJ-Z HM131.S431 2003

1. Find the bookrack corresponding to HJ-Z will help you find the book
2. Find the range for HM131

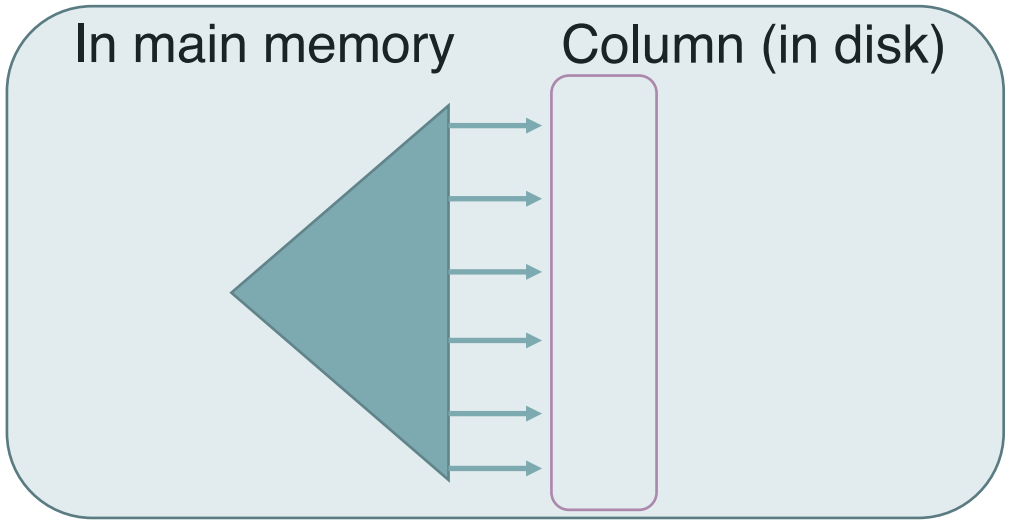
This is a kind of index

ENHANCED WITH INDEX

In column store, we can build index for a column

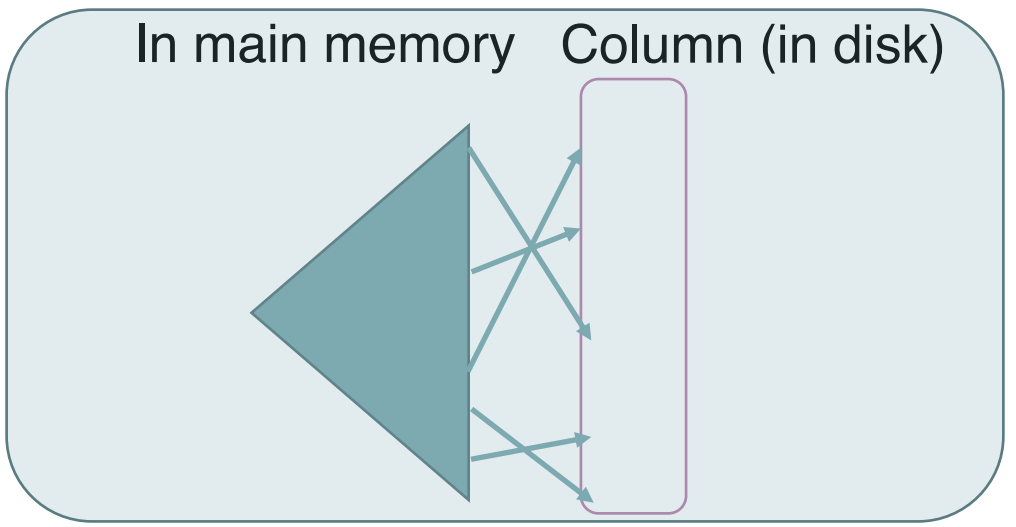
If the column is sorted

Clustered index



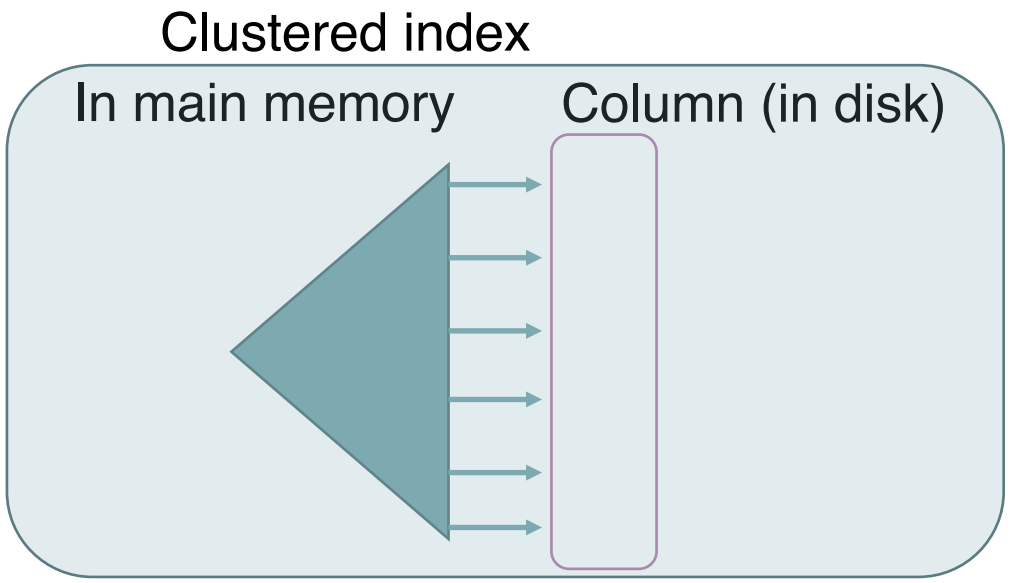
If the column is unsorted

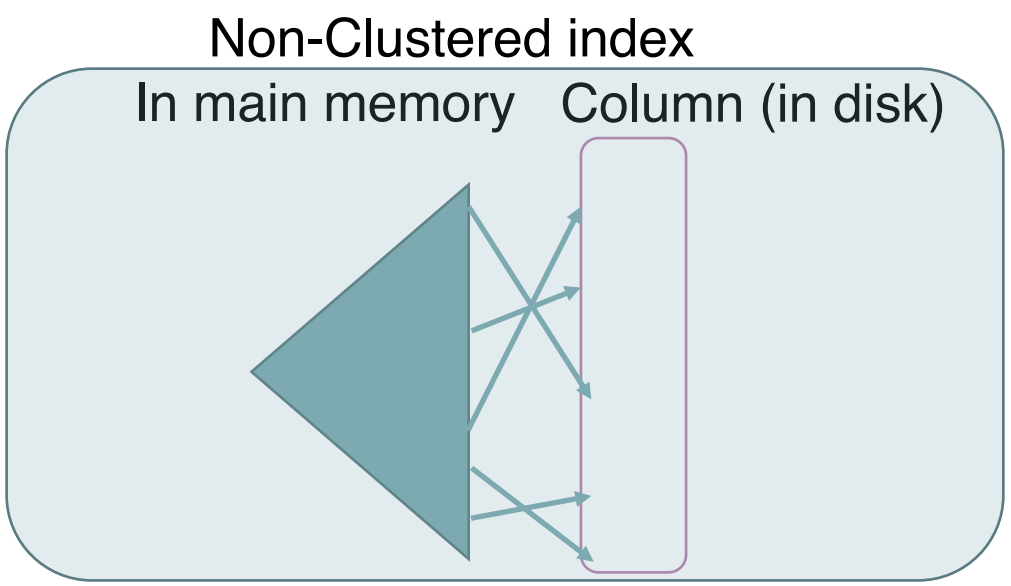
Non-Clustered index



ENHANCED WITH INDEX

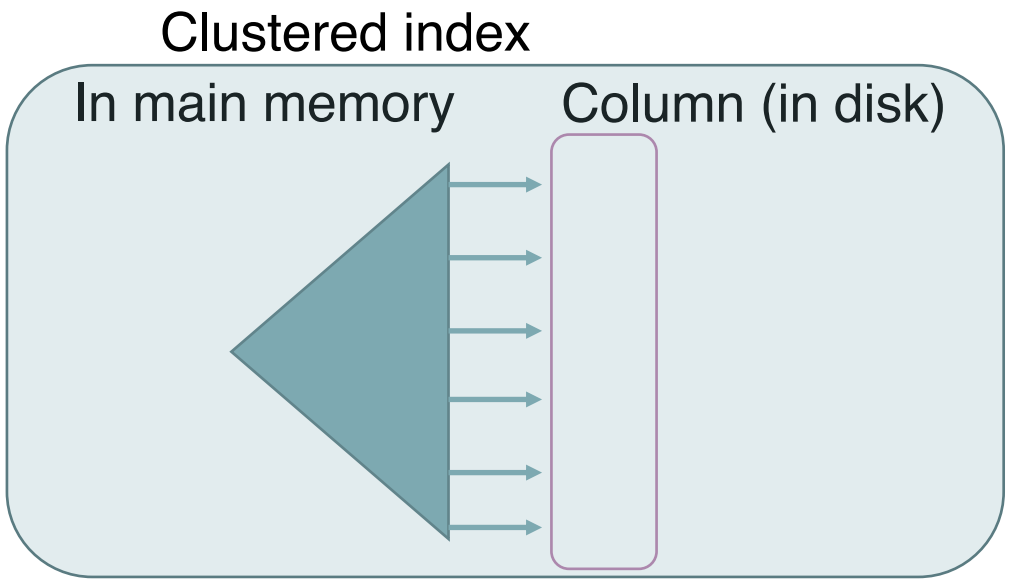
- In one table, we can apply both clustered index and non-clustered index.
- One clustered index for a table (why?)
- Can have multiple non-clustered index for a table
- Clustered index is typically faster in extracting values compared with non-clustered index (why?)

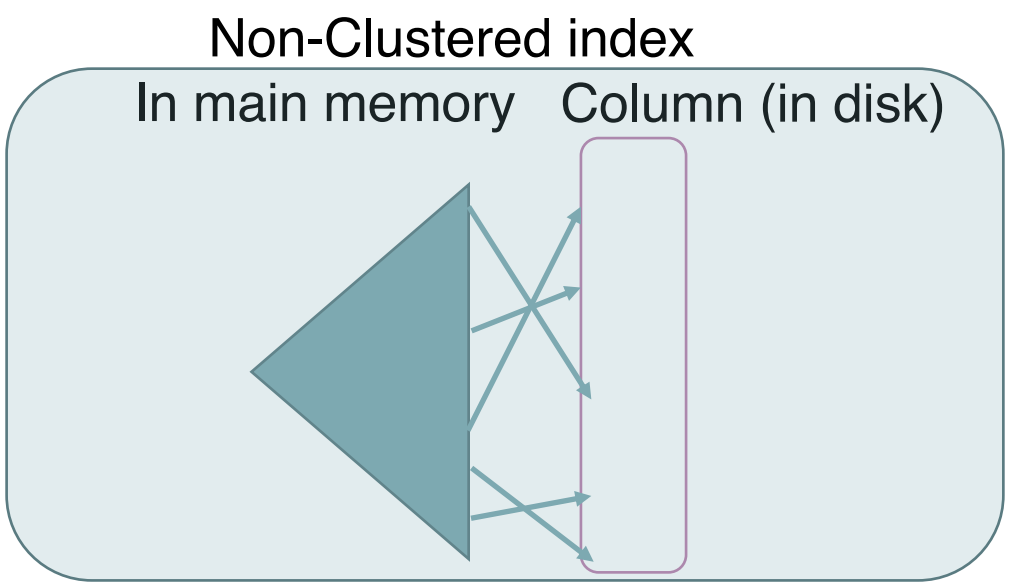




ENHANCED WITH INDEX

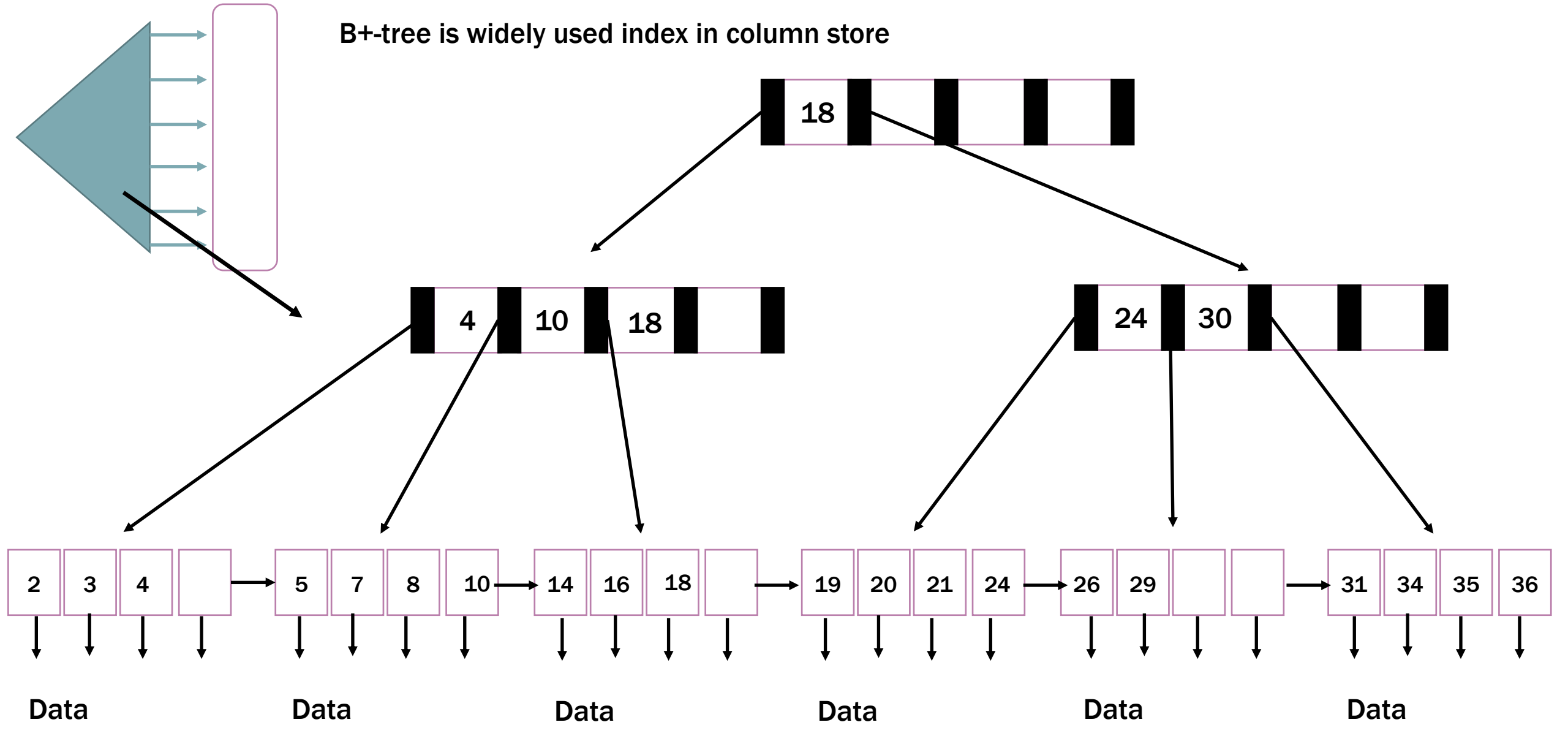
- Index access is much faster than data access, because index is stored in Main Memory, while data is stored in disk.
- So it is beneficial to have complicated index access to locate the exact pages where stores the data.





ENHANCED WITH INDEX

B+-tree is widely used index in column store



More discussions

Hybrid layouts

Anastasia Ailamaki, EPFL

ACM SIGMOD Innovations award, 2019

PAX: store all data about a row in a single disk page but organize data in a column way inside each page

Weaving Relations for Cache Performance, VLDB 2001

More practice for Column Stores

QUESTION1

Use **shared scan** to find the 2^{nd} largest value and 2^{nd} smallest value in an integer array A of size t. **Assume that the integers in A are different.**

RECAP-WHAT IS SHARED SCAN

Shared scan minimizes the scan cost. Typically, one pass of scan is preferable.

Think about the question… How to write the code?

LET'S START WITH WHATEVER WE HAVE

First, we can use the following code to compute the max and second max of array A.

```
max= - ∞;

second_max= - ∞;

for (i=0;i<t;i++){

        if(A[i]>max) {second_max=max; max=A[i]; }

        if(A[i]<= max && A[i]>second_max) second_max=A[i];

}
```

LET'S START WITH WHATEVER WE HAVE

Second, we can use the following code to compute the min and second min of array A.

```
min= ∞;

second_min= ∞;

for (i=0;i<t;i++){

        if(A[i]<min) {second_min=min; min=A[i]; }

        if(A[i]>= min && A[i]<second_min) second_min=A[i];

}
```

SHARED SCAN

Solution:

The idea of shared scan is to do multiple operations in one scan of the data.

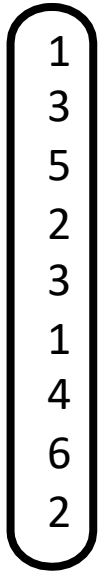
```
max= - ∞;
second_max= - ∞;
min= ∞;
second_min= ∞;
for (i=0;i<t;i++){
        if(A[i]>max) {second_max=max; max=A[i]; }
        if(A[i]<= max && A[i]>second_max) second_max=A[i];
        if(A[i]<min) {second_min=min; min=A[i]; }
        if(A[i]>= min && A[i]<second_min) second_min=A[i];
}
```

QUESTION 2

Suppose we have the following column applied with zone map of size 3.

1) Illustrate the information recorded for each zone.
2) Give the steps of using zone map to answer queries “A>4”
3) Give the steps of using zone map to answer queries “A>7 or A<2

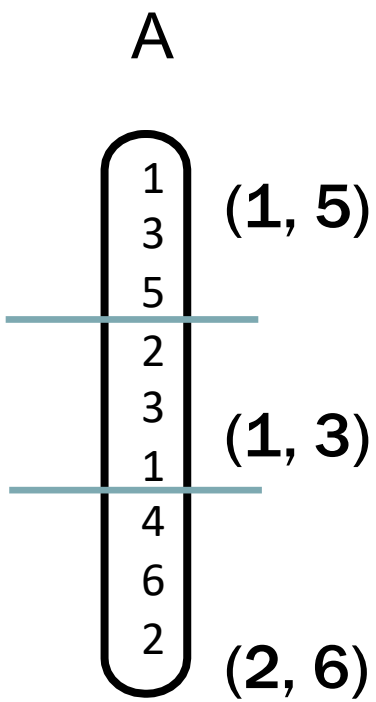
A



QUESTION 2

Suppose we have the following column applied with zone map of size 3.

1) Illustrate the information recorded for each zone.
2) Give the steps of using zone map to answer queries “A>4”
3) Give the steps of using zone map to answer queries “A>7 or A<2

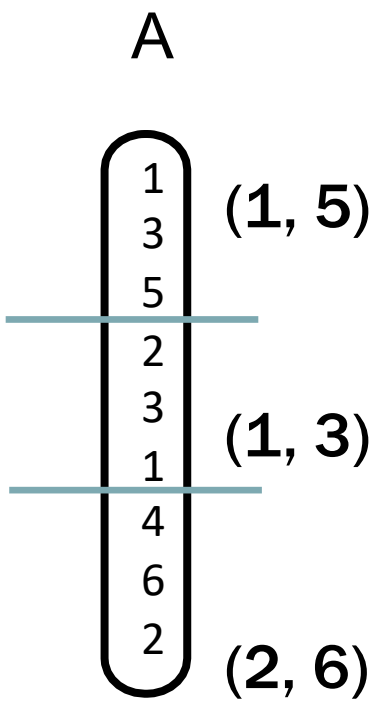


Compute the min, max for each zone and record them in memory.

QUESTION 2

Suppose we have the following column applied with zone map of size 3.

1) Illustrate the information recorded for each zone.
2) Give the steps of using zone map to answer queries “A>4”
3) Give the steps of using zone map to answer queries “A>7 or A<2

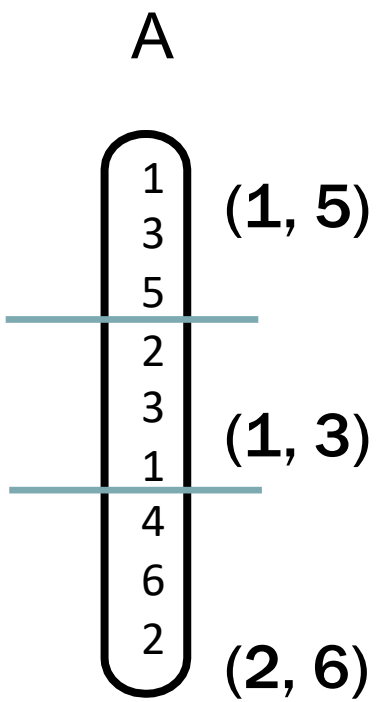


Querying A>4:
Check the min, max of each zone, and skip those do not contain value > 4.
The 2nd zone does not contain value > 4, and hence no need to access that zone.

QUESTION 2

Suppose we have the following column applied with zone map of size 3.

1) Illustrate the information recorded for each zone.
2) Give the steps of using zone map to answer queries “A>4”
3) Give the steps of using zone map to answer queries “A>7 or A<2”



Querying A>7 or A<2:
Check the min, max of each zone, and skip those do not contain value > 7 or <2.
The 3rd zone does not contain value > 7 or <2, and hence no need to access that zone.

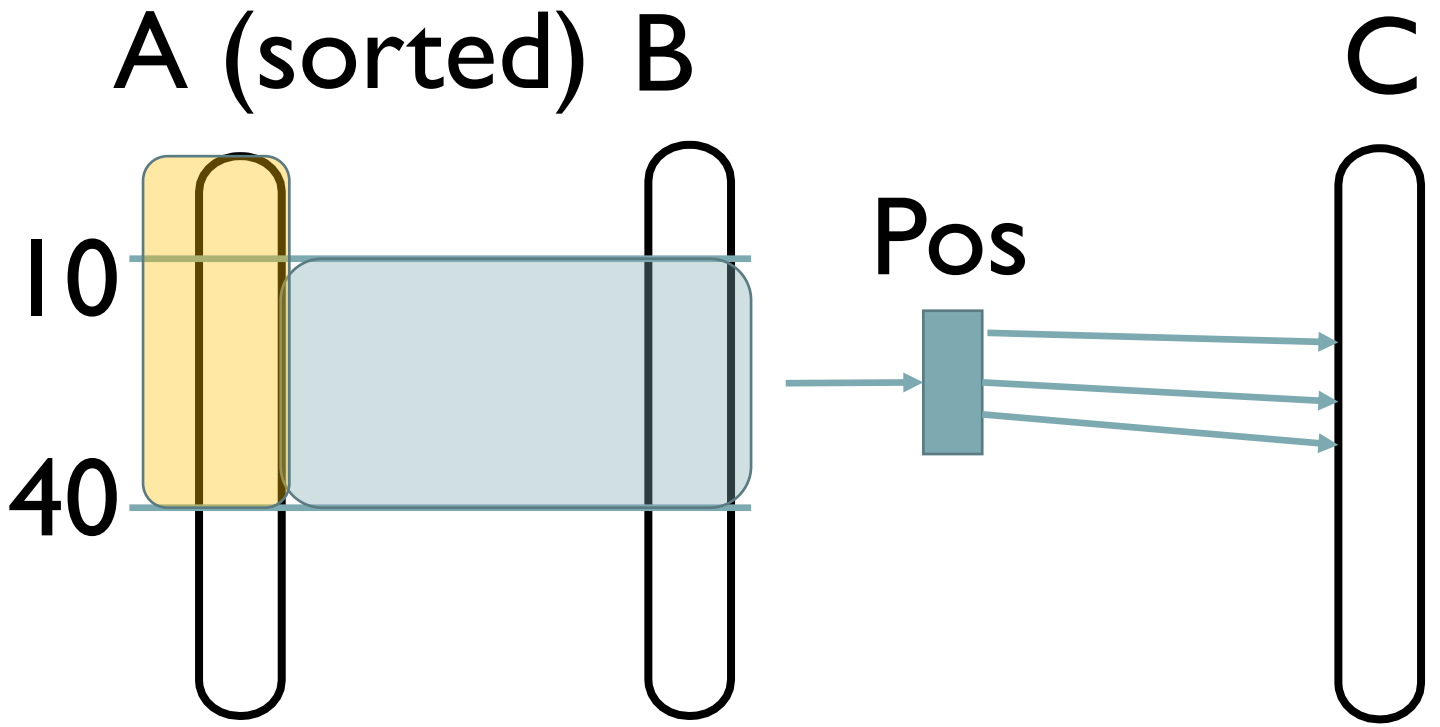
QUESTION 3

In lecture slides, we discuss two ways of scanning a sorted column: scanning from the start and scanning using binary search.

Give one extreme scenario where the first method is better.

RECAP – SCANNING FROM THE START

SELECT max(C) FROM T WHERE A>10 and A<40 and B>20



First scan A up to 40 (yellow)

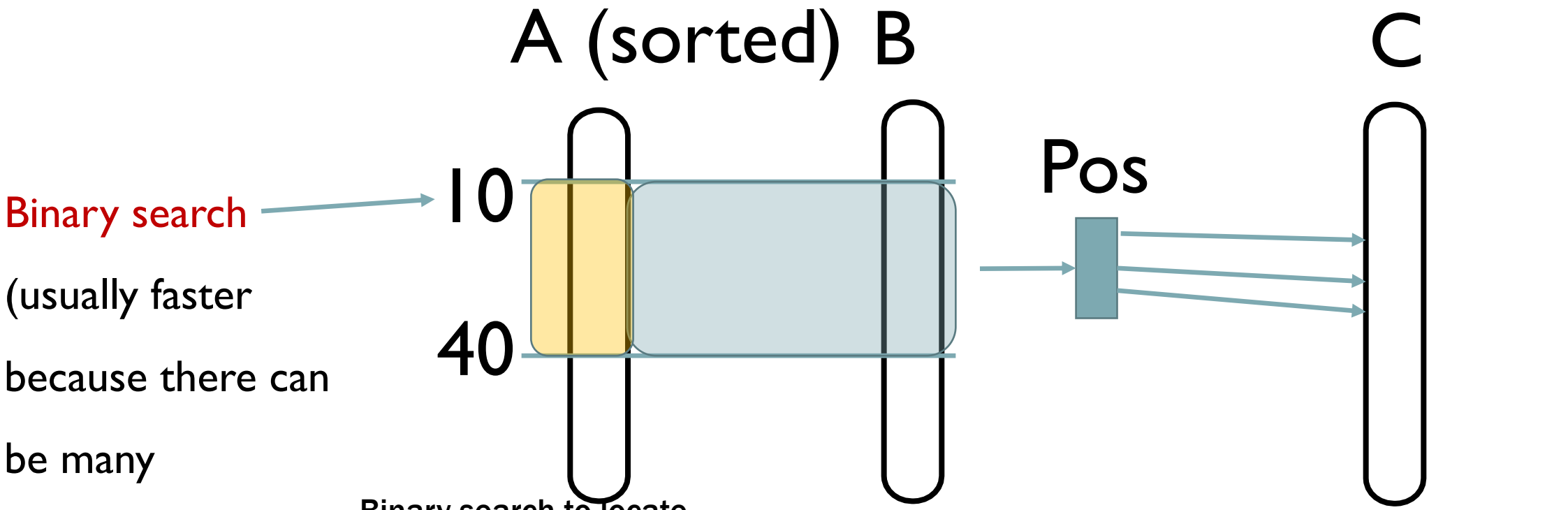
Then scan B in blue range and filter for B>20

Store positions for tuples having A>10, A<40 and B>20

Select max(C) based on the positions

RECAP– USE BINARY SEARCH

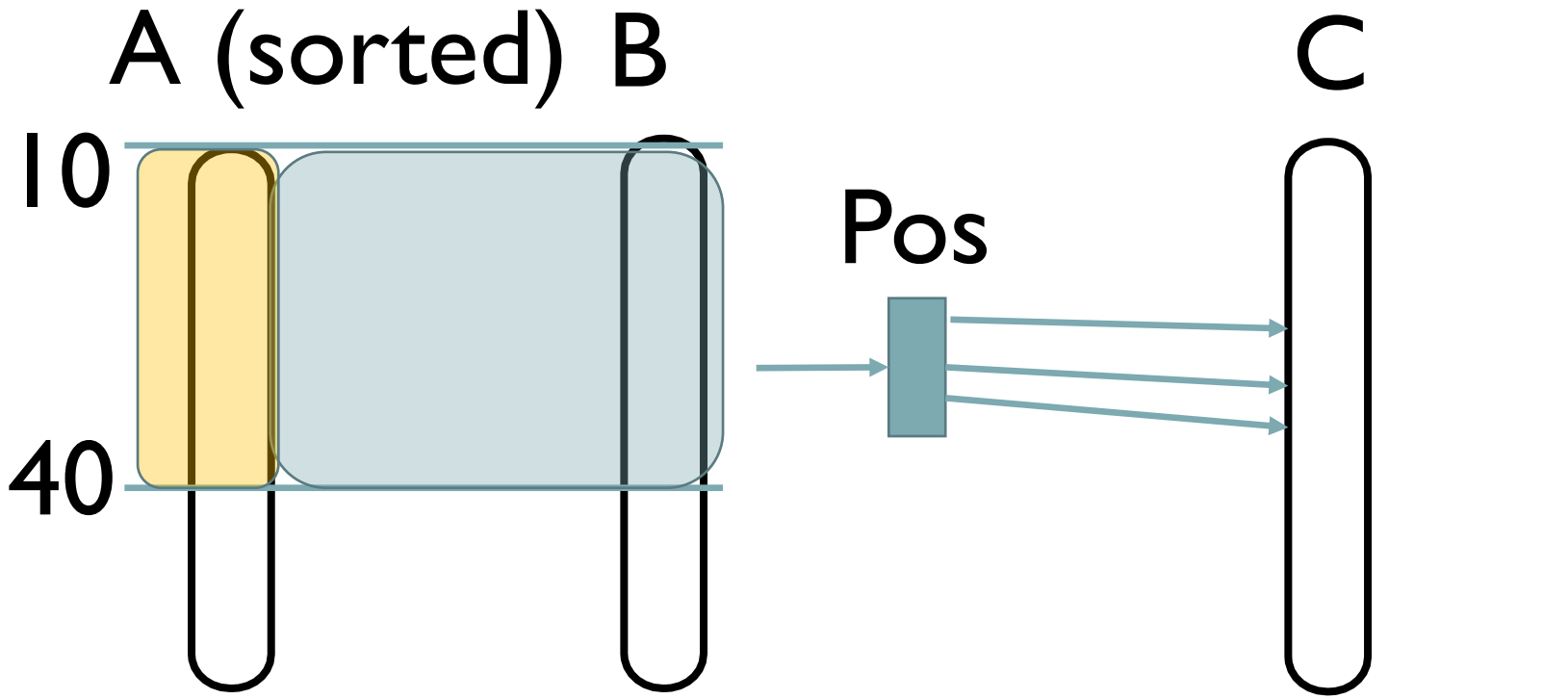
SELECT max(C) FROM T WHERE A>10 and A<40 and B>20



POSSIBLE CASE 1 – FIRST DATA ITEM IN A IS ALREADY IN [10, 40]

SELECT max(C) FROM T WHERE A>10 and A<40 and B>20

Binary search cost is unnecessary



First scan A up to 40 (yellow)

Then scan B in blue range and filter for B>20

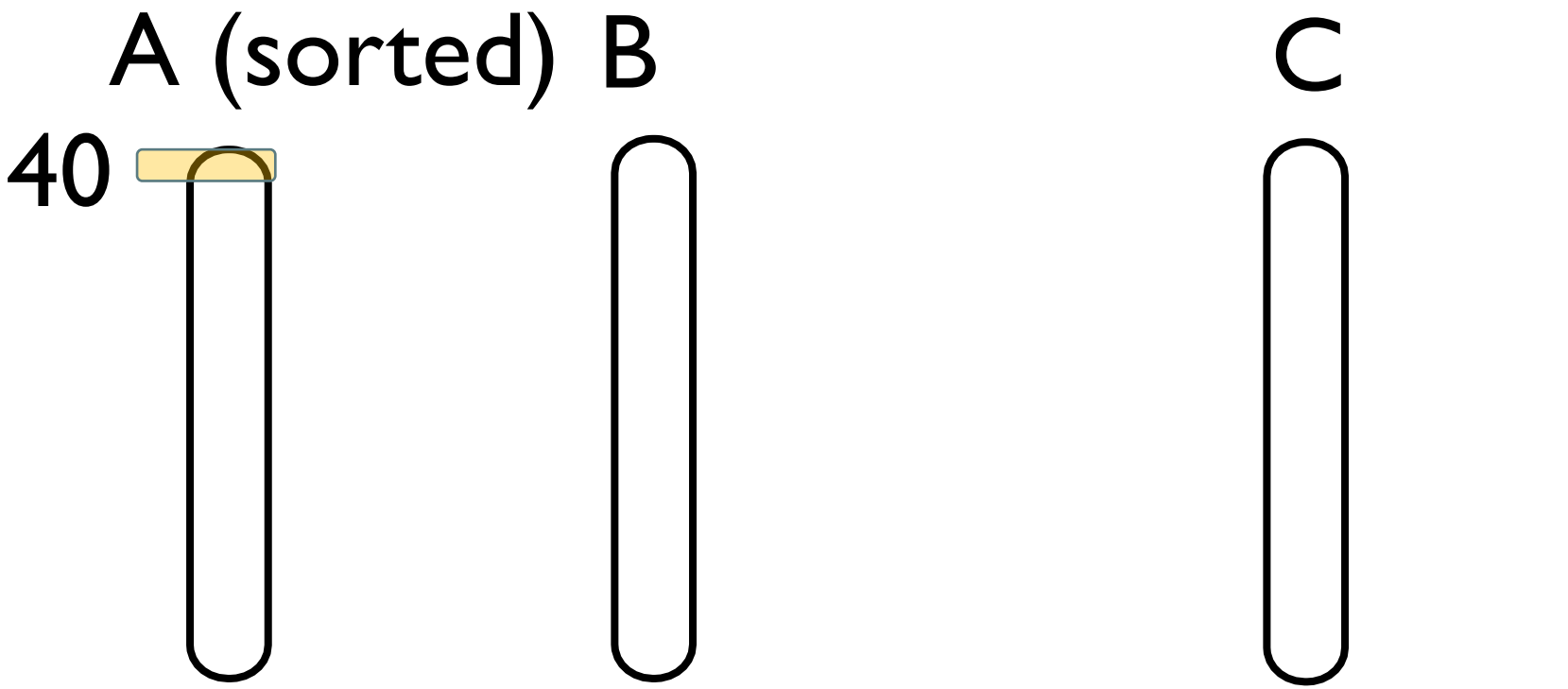
Store positions for tuples having A>10, A<40 and B>20

Select max(C) based on the positions

POSSIBLE CASE 2 – FIRST DATA ITEM IN A IS ALREADY >= 40

SELECT max(C) FROM T WHERE A>10 and A<40 and B>20

Binary search cost is unnecessary



First scan A up to 40 (yellow)

Then scan B in blue range and filter for B>20

Store positions for tuples having A>10, A<40 and B>20

Select max(C) based on the positions

QUESTION 4

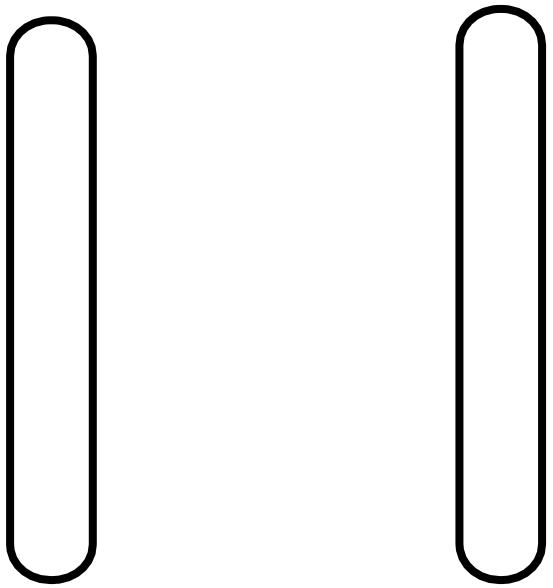
Given a table T of two columns A and B. Assume that the table is stored in a column store, and it has two copies: the first copy has column A sorted (while column B follows the order of column A); the second copy has column B sorted (while column A follows the order of column B).

(1)For query “select max(B) from T where A>3 and A<6”, which copy should be used for lower cost?
(2)For query “select max(A) from T where B>3 and B<6”, which copy should be used for lower cost?
(3)We can also use multiple copies simultaneously. Please describe possible issues of using multiple copies.

SOLUTIONS (1) (2)

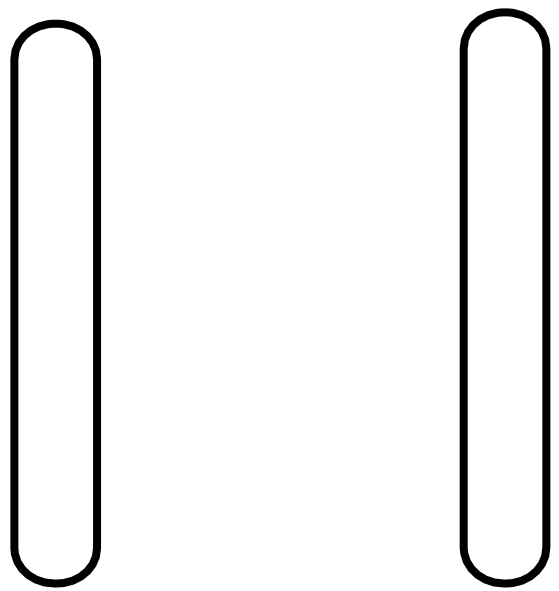
Queries on sorted columns are faster!

A (sorted) B



For query “select max(B) from T where A>3 and A<6”, which copy should be used for lower cost?

A B (sorted)



For query “select max(A) from T where B>3 and B<6”, which copy should be used for lower cost?

QUESTION(3)-OPEN DISCUSSION

Describe possible issues of using multiple copies.

Think…

QUESTION(3)-OPEN DISCUSSION

Describe possible issues of using multiple copies.

(1) Large space

(2) Updates have to be conducted on each copy

We finish lectures for Column Store!

Next lecture:

Distributed Systems and MapReduce