CSE 444: Database Internals

Section 3:

Operator Algorithms

Notations

- B(R) = # of blocks (i.e. pages) for relation R
- T(R) = # of tuples in relation R
- V(R, a) = # of distinct values of attribute a
- Memory M

Algorithms for Group By and Aggregate Operators

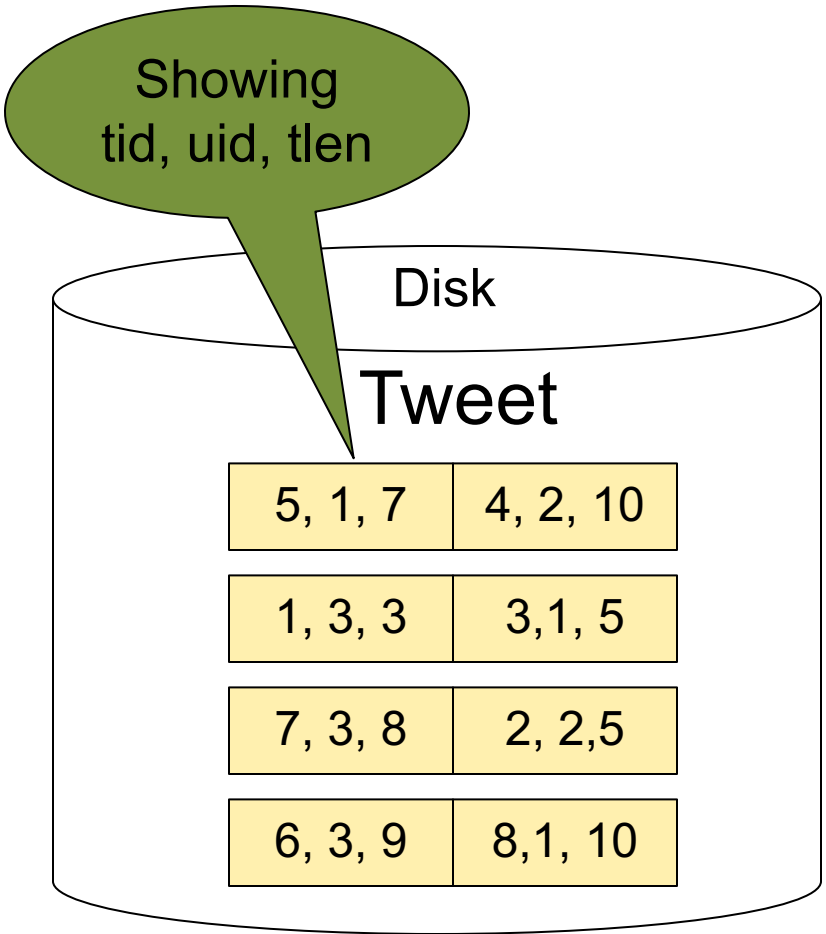
- Modified Tweet Example:

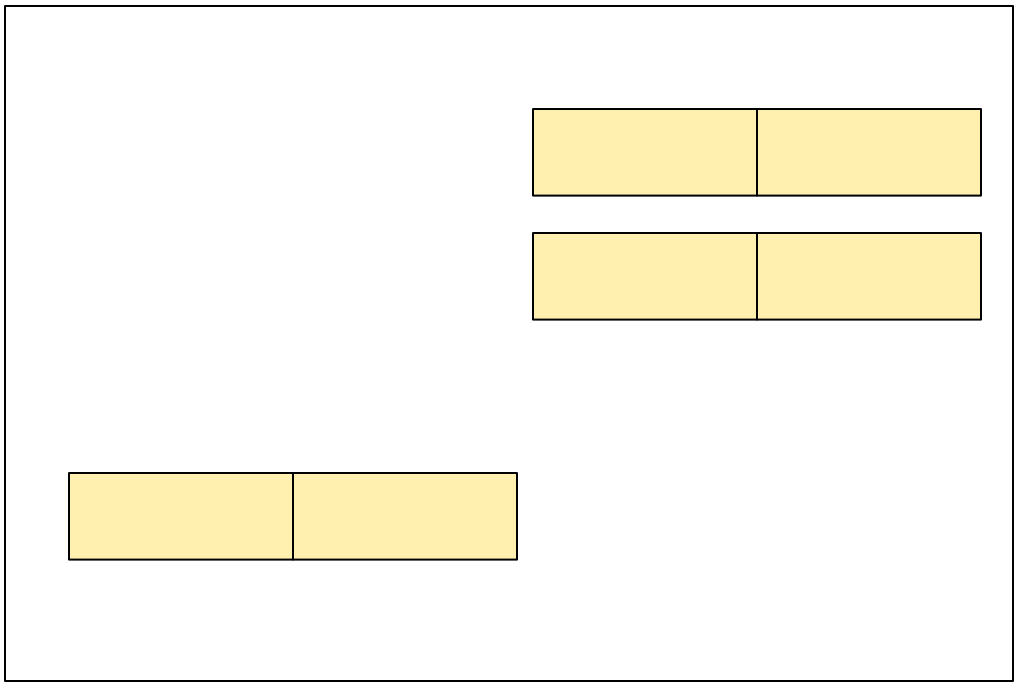
Tweet(tid, uid, tlen) tlen = tweet length

```
SELECT uid, MIN(tlen)
FROM Tweet
GROUP BY uid
```

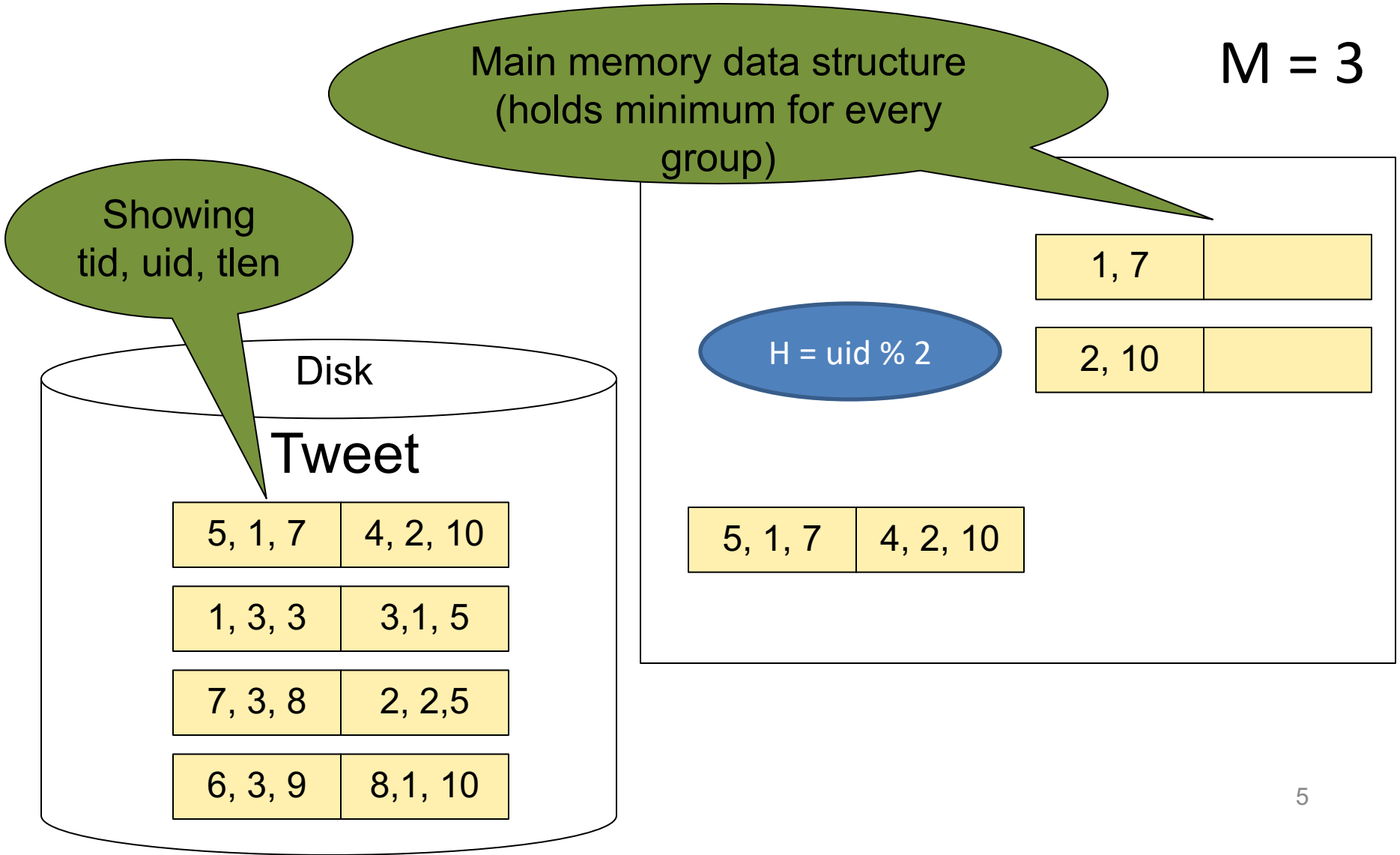
One pass, hash-based grouping

M = 3



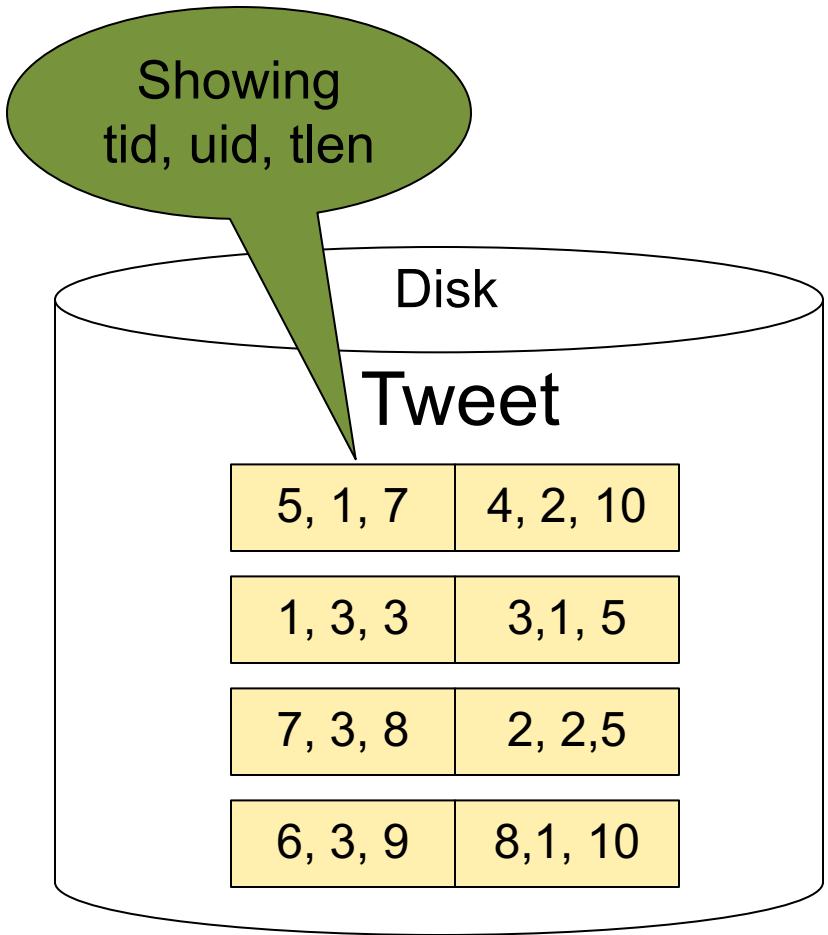


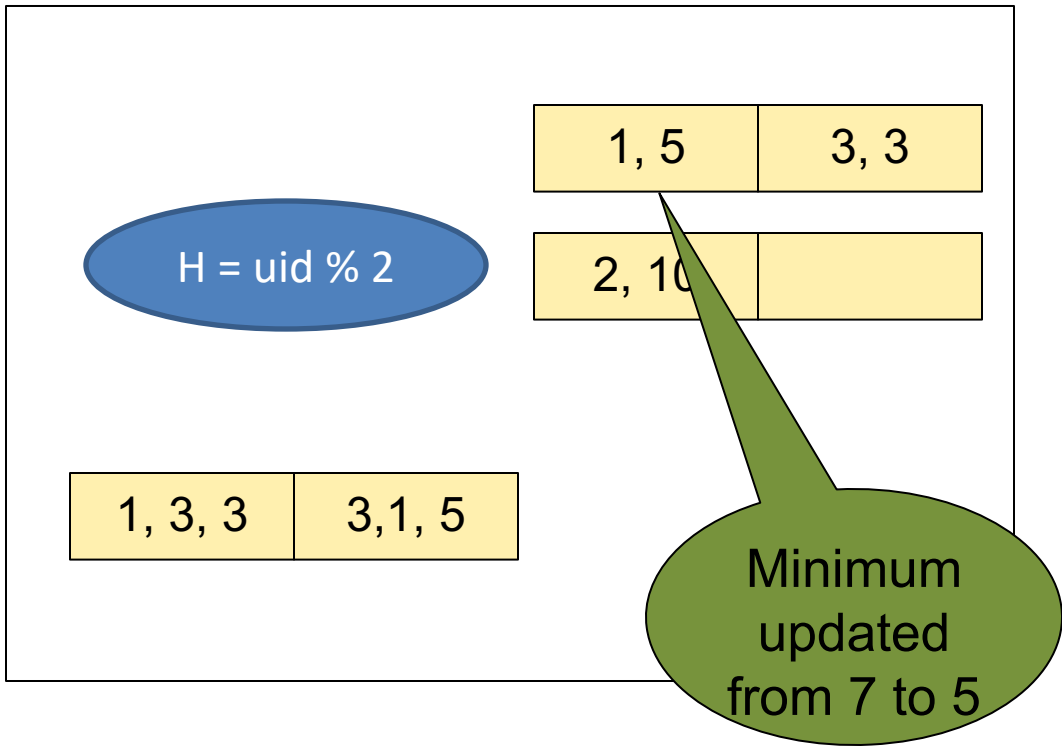
One pass, hash-based grouping



One pass, hash-based grouping

M = 3





Discussion

Cost:

- Clustered?
- Unclustered?

Which operator method does the grouping?

open(), next(), or close()?

What to do for AVG(tlen)?

Discussion

Cost:

- Clustered?

  - B(R): assuming M – 1 pages can hold all groups – tuples for groups can be shorter or larger than original tuples

- Unclustered?

  - Also B(R)

Which method does the grouping:

open(), next(), or close()?

- Cannot return anything until the entire data is read. Open() needs to do grouping

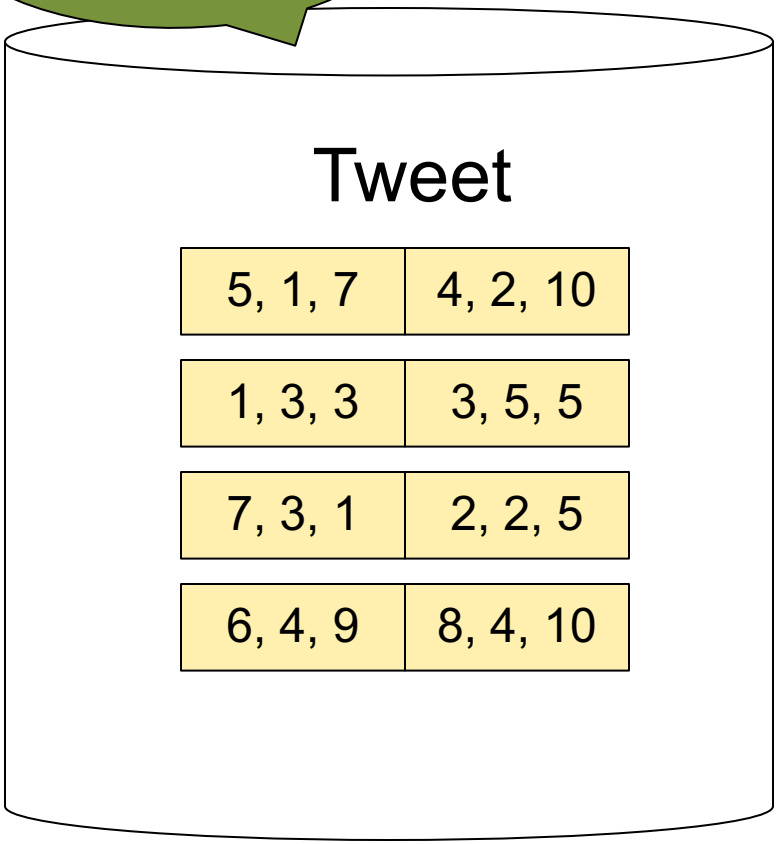
What to do for AVG(tlen)?

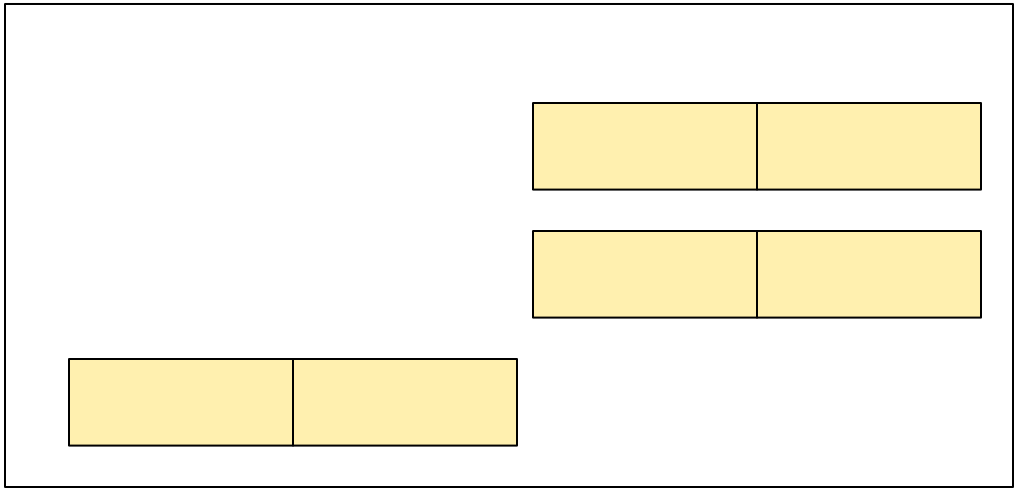
- Keep both SUM(tlen) and COUNT(*) for each group in memory

Two pass, hash-based grouping

Showing
tid, uid, tlen

M = 3





Two pass, hash-based grouping

Showing
tid, uid, tlen

No aggregation is performed in the first pass

M = 3

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

H = uid % 2

5, 1, 7	4, 2, 10

5, 1, 7	
4, 2, 10	

Two pass, hash-based grouping

Showing
tid, uid, tlen

No aggregation is performed in the first pass

M = 3

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

H = uid % 2

1, 3, 3	3, 5, 5

5, 1, 7	1, 3, 3
4, 2, 10	

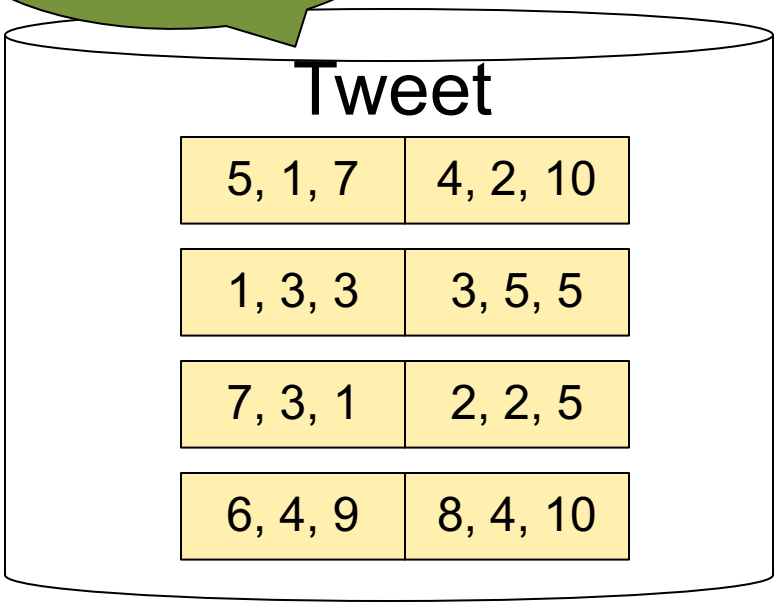
Flush!

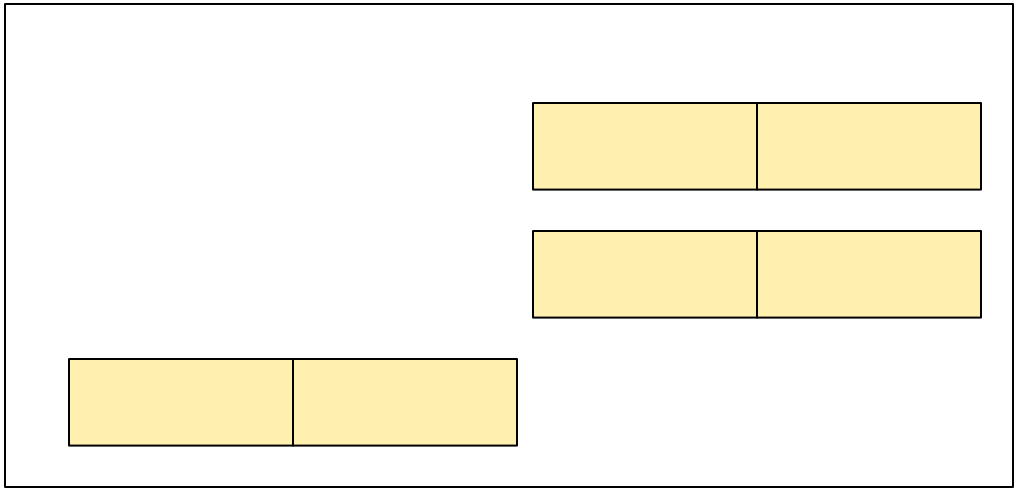
Two pass, hash-based grouping

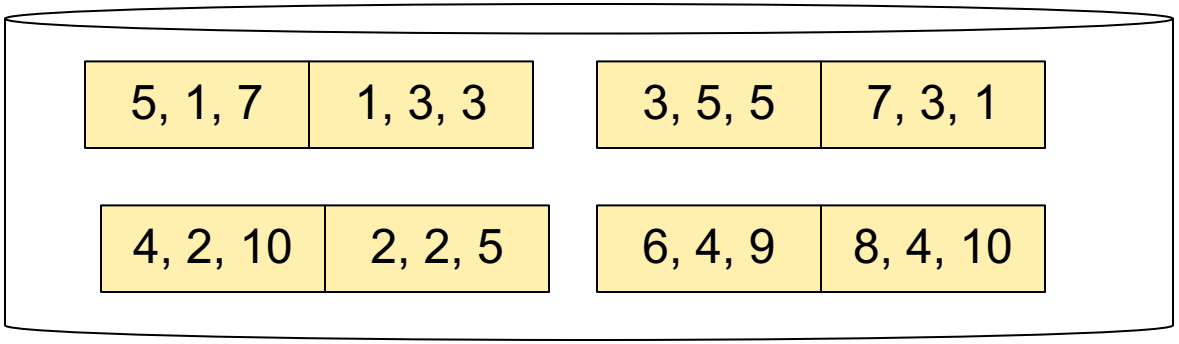
Showing
tid, uid, tlen

Final buffer and disk after pass 1

M = 3





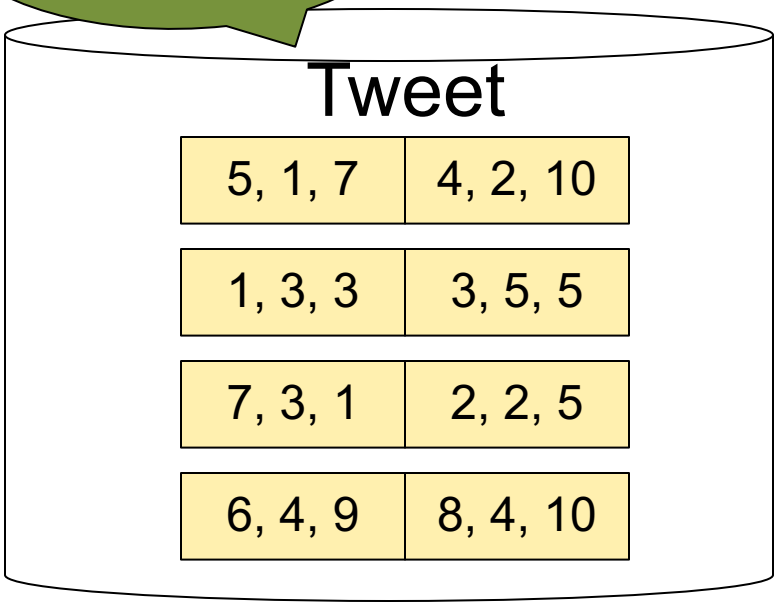


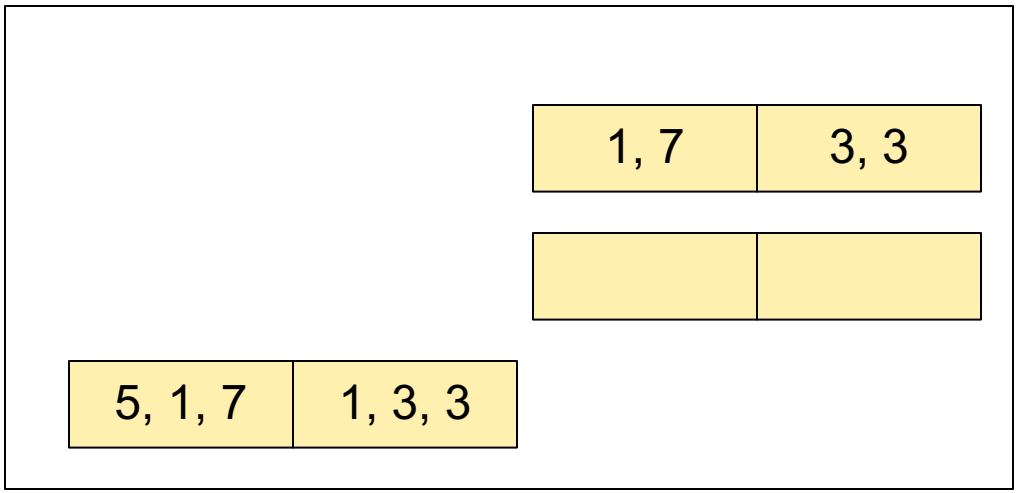
Two pass, hash-based grouping

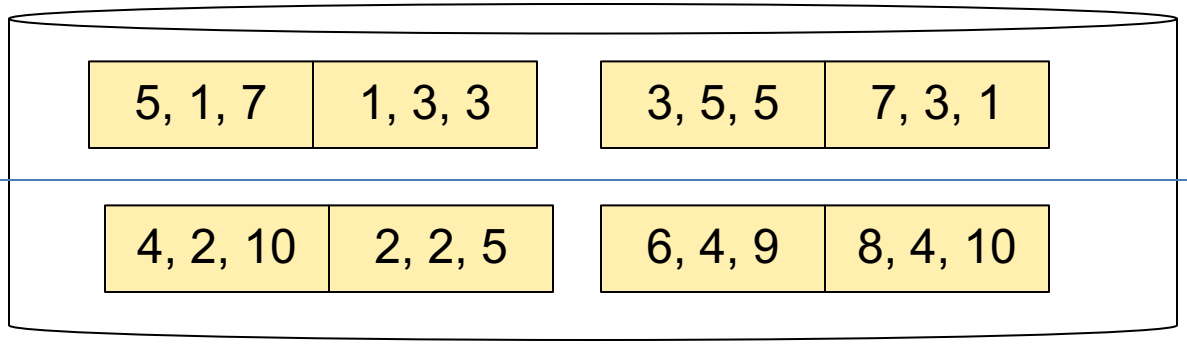
Showing
tid, uid, tlen

Second pass: compute aggregate in each bucket
Need to keep only one record per group

M = 3





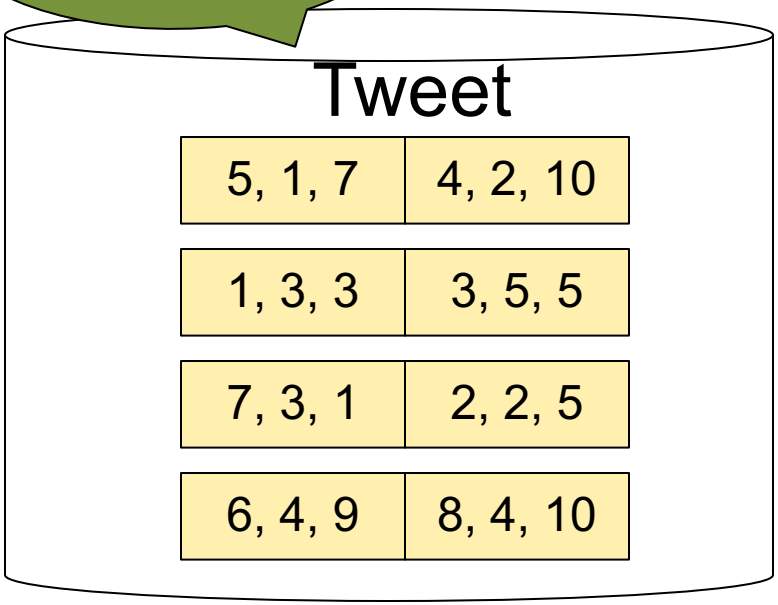


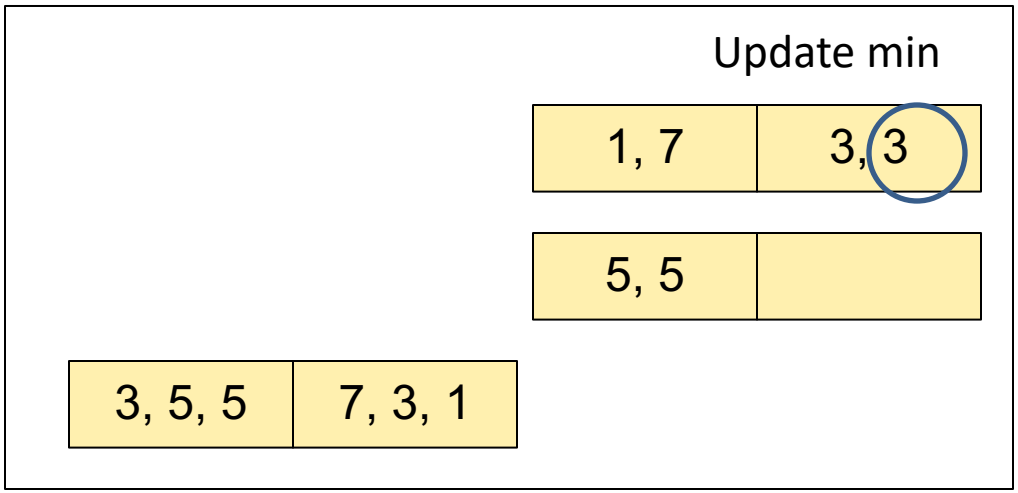
Two pass, hash-based grouping

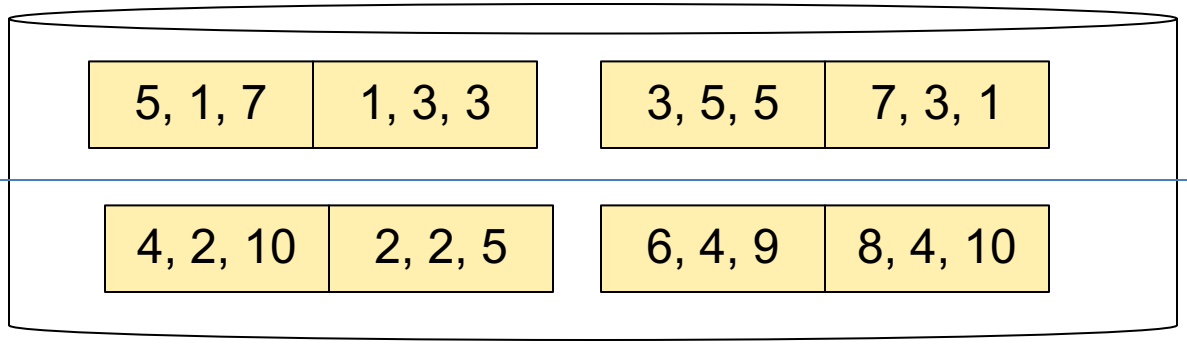
Showing
tid, uid, tlen

Second pass: compute aggregate in each bucket
Need to keep only one record per group

M = 3







Discussion

Cost?

- 3B(R)

Assumptions?

- Need to hold all distinct values in the same bucket in M-1
- Assuming uniformity, B(R) <= M^2 is safe to assume
  - i.e. B(R)/M <= M

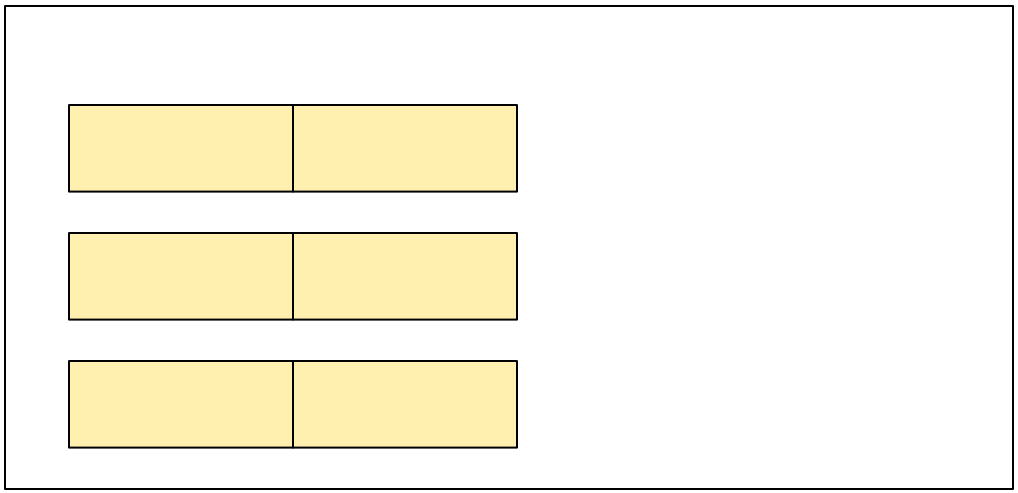
Two pass, sort-merge-based grouping

Showing
tid, uid, tlen

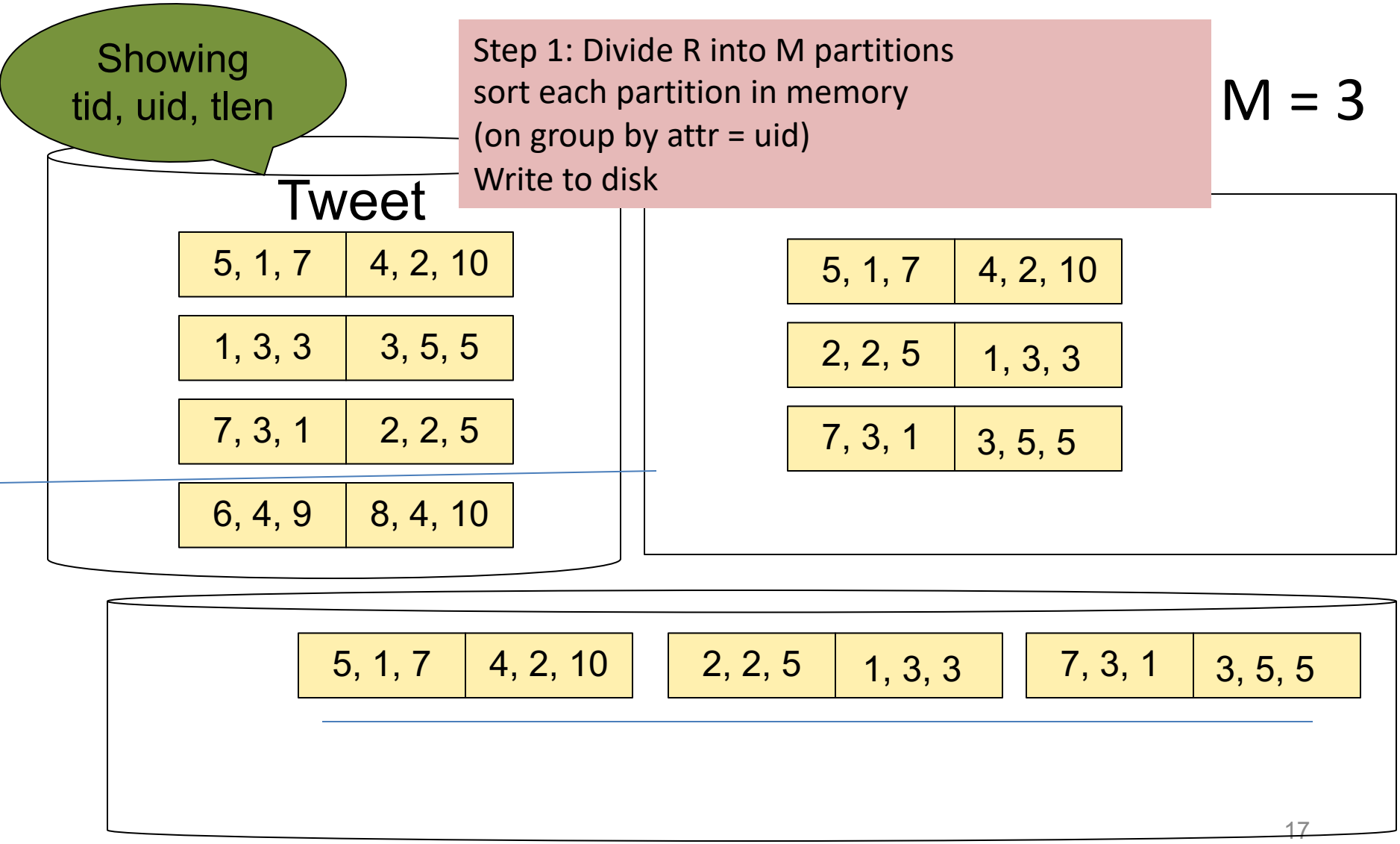
Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3 ,5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

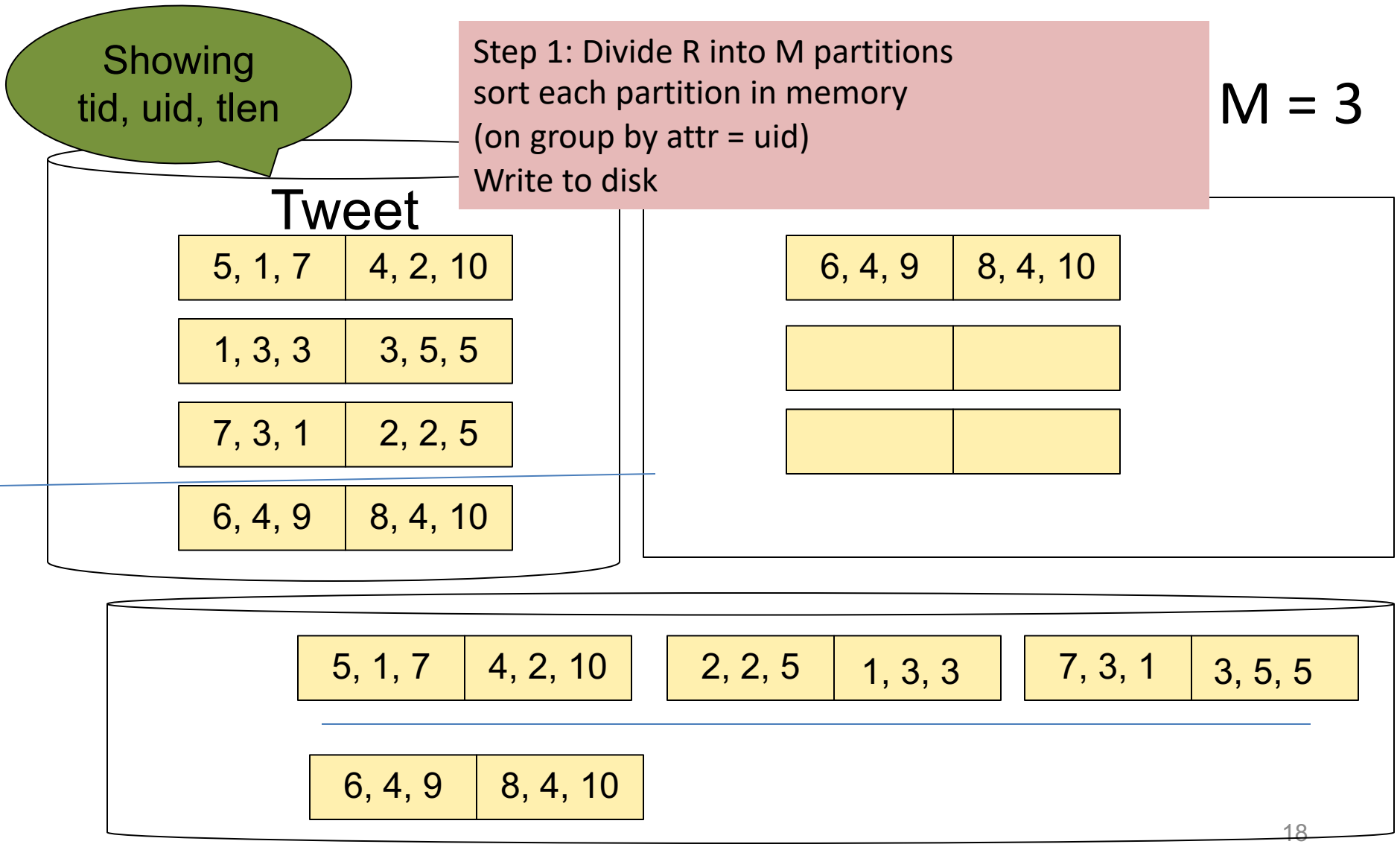
M = 3



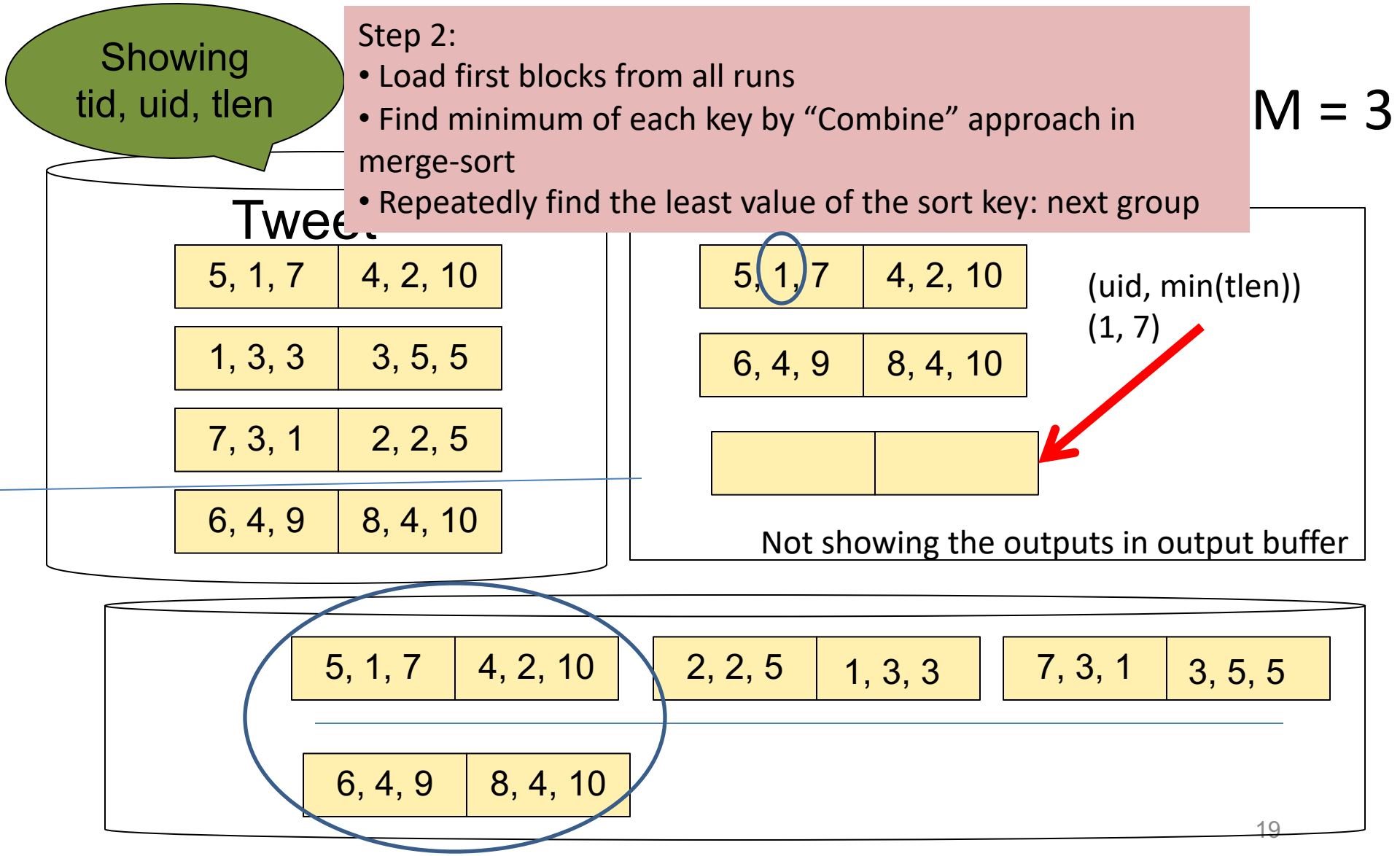
Two pass, sort-merge-based grouping



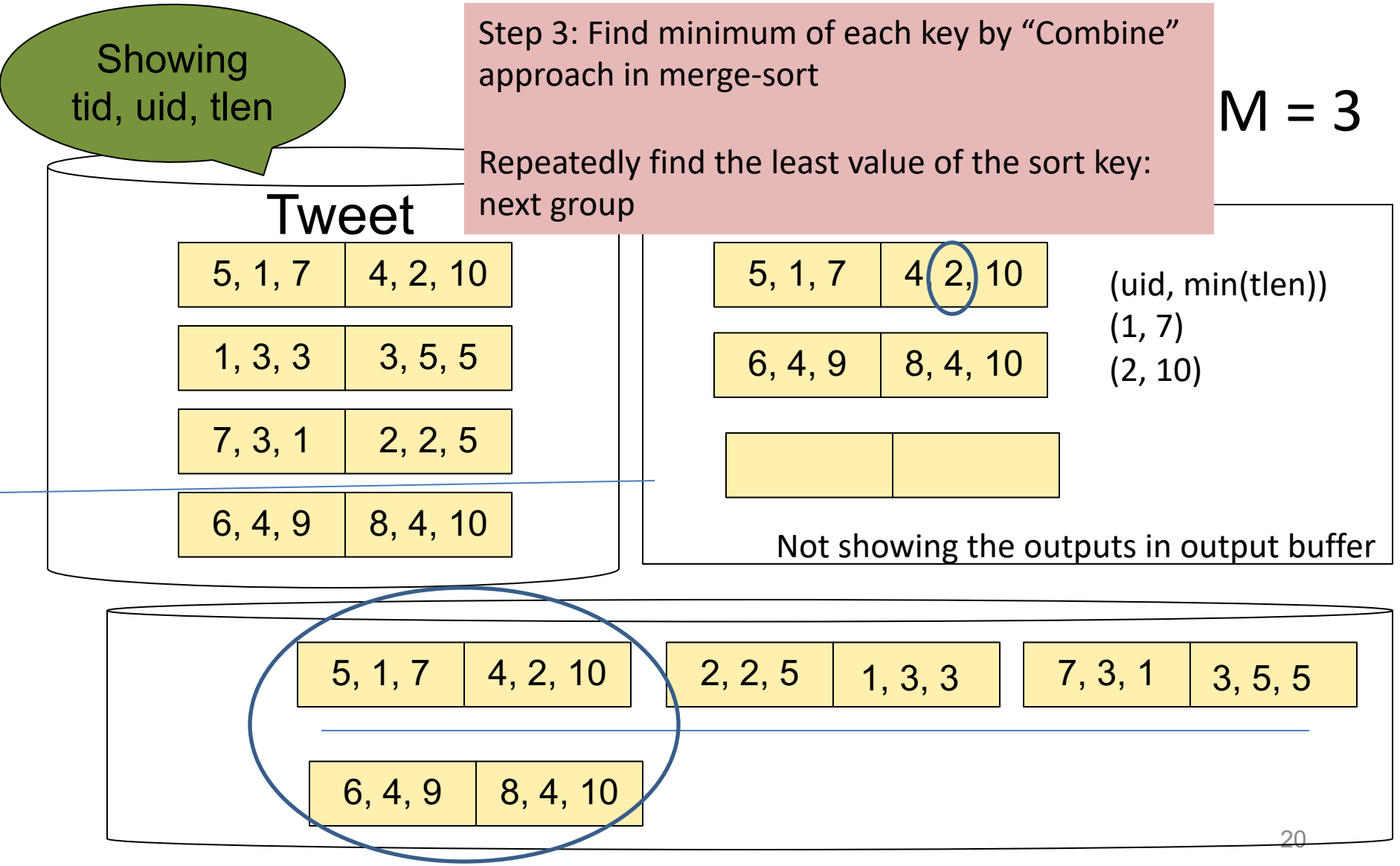
Two pass, sort-merge-based grouping



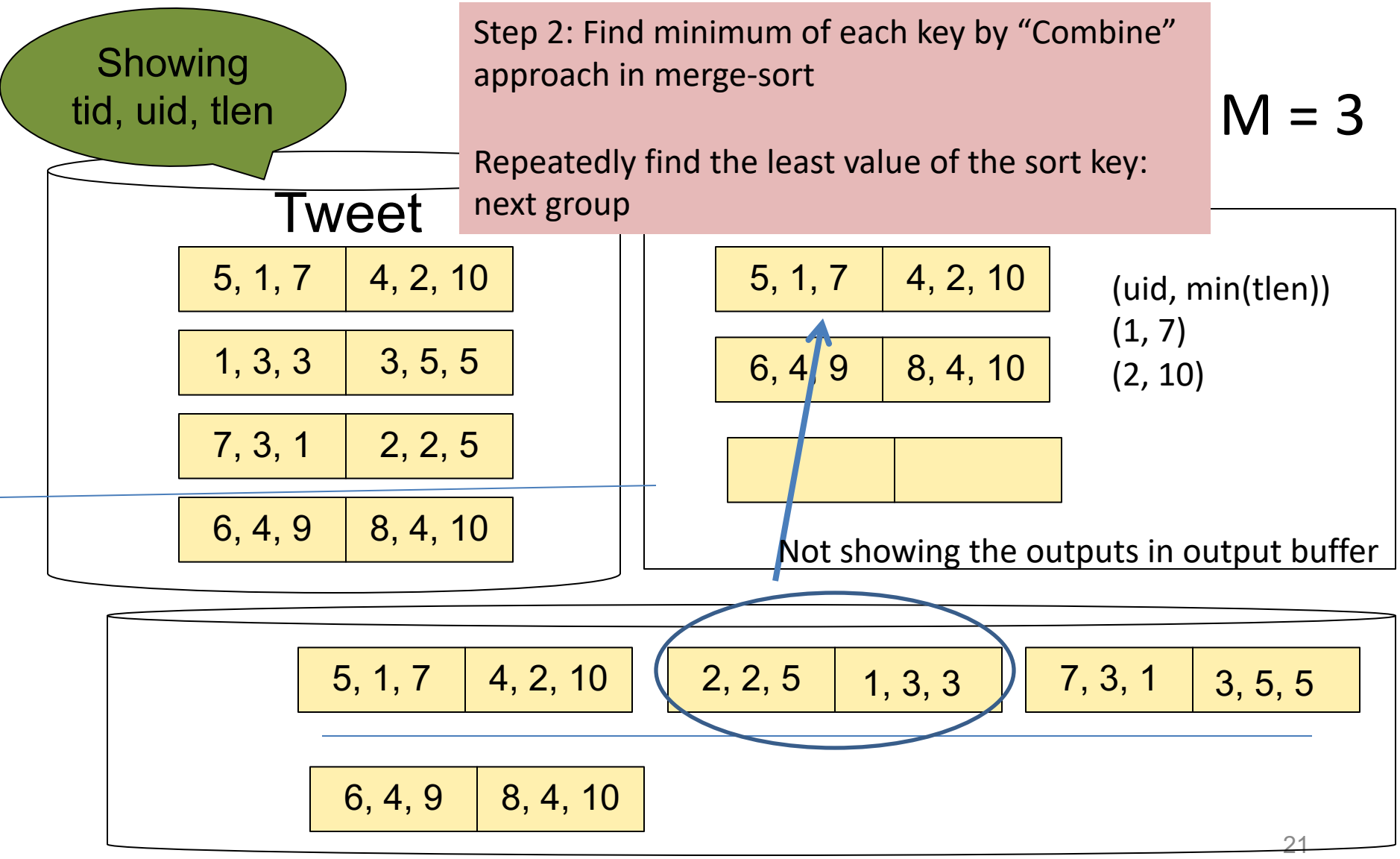
Two pass, sort-merge-based grouping



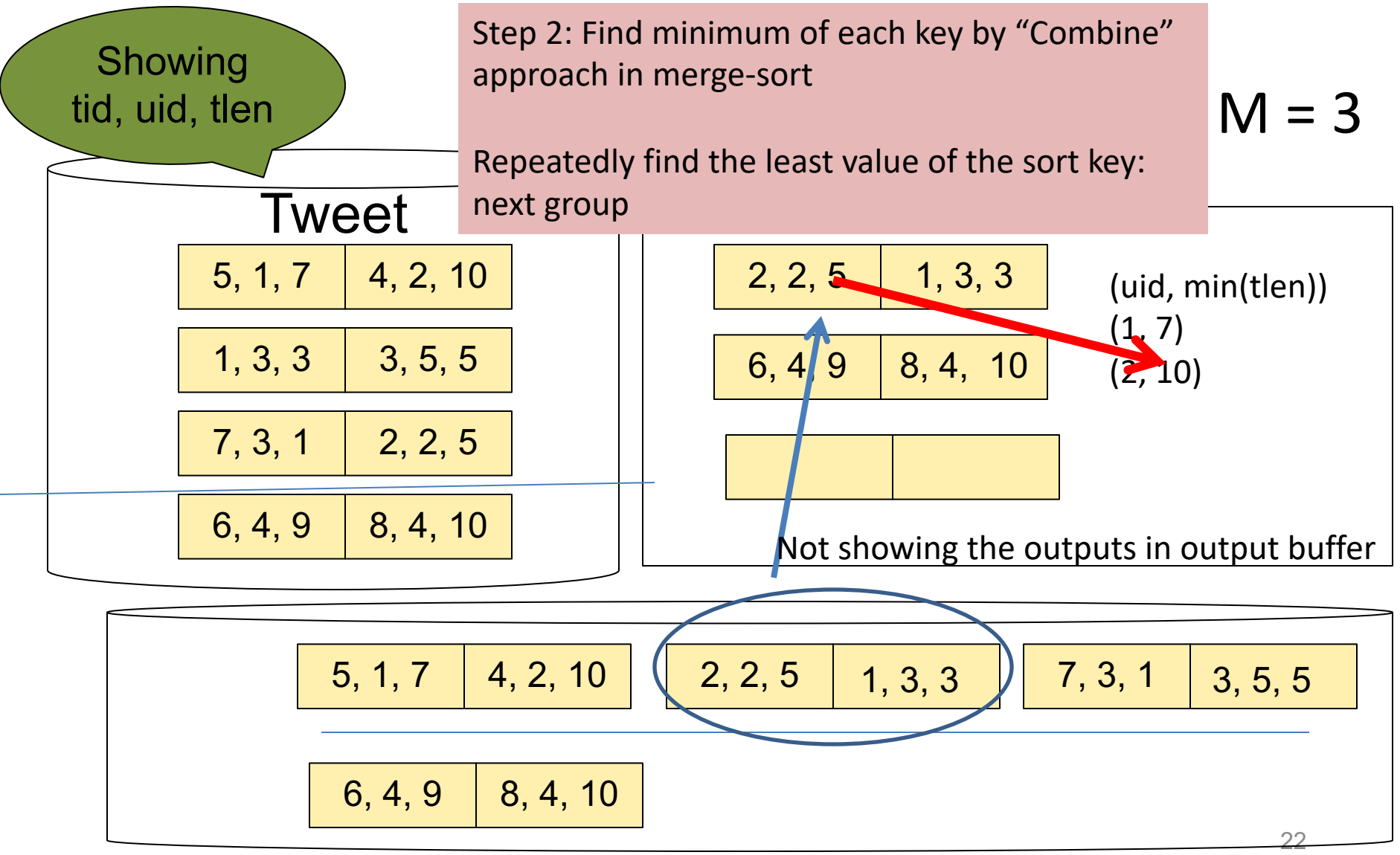
Two pass, sort-merge-based grouping



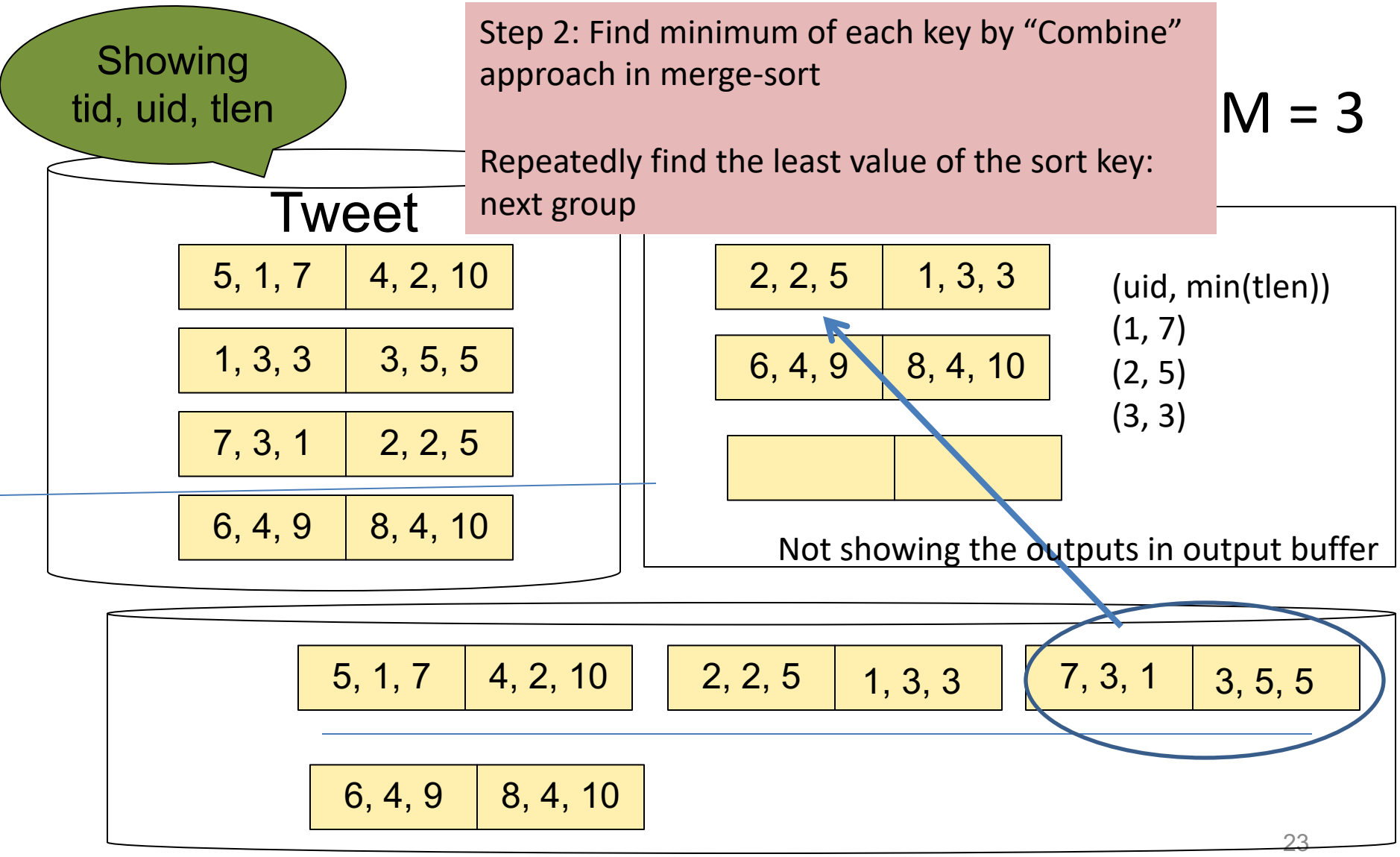
Two pass, sort-merge-based grouping



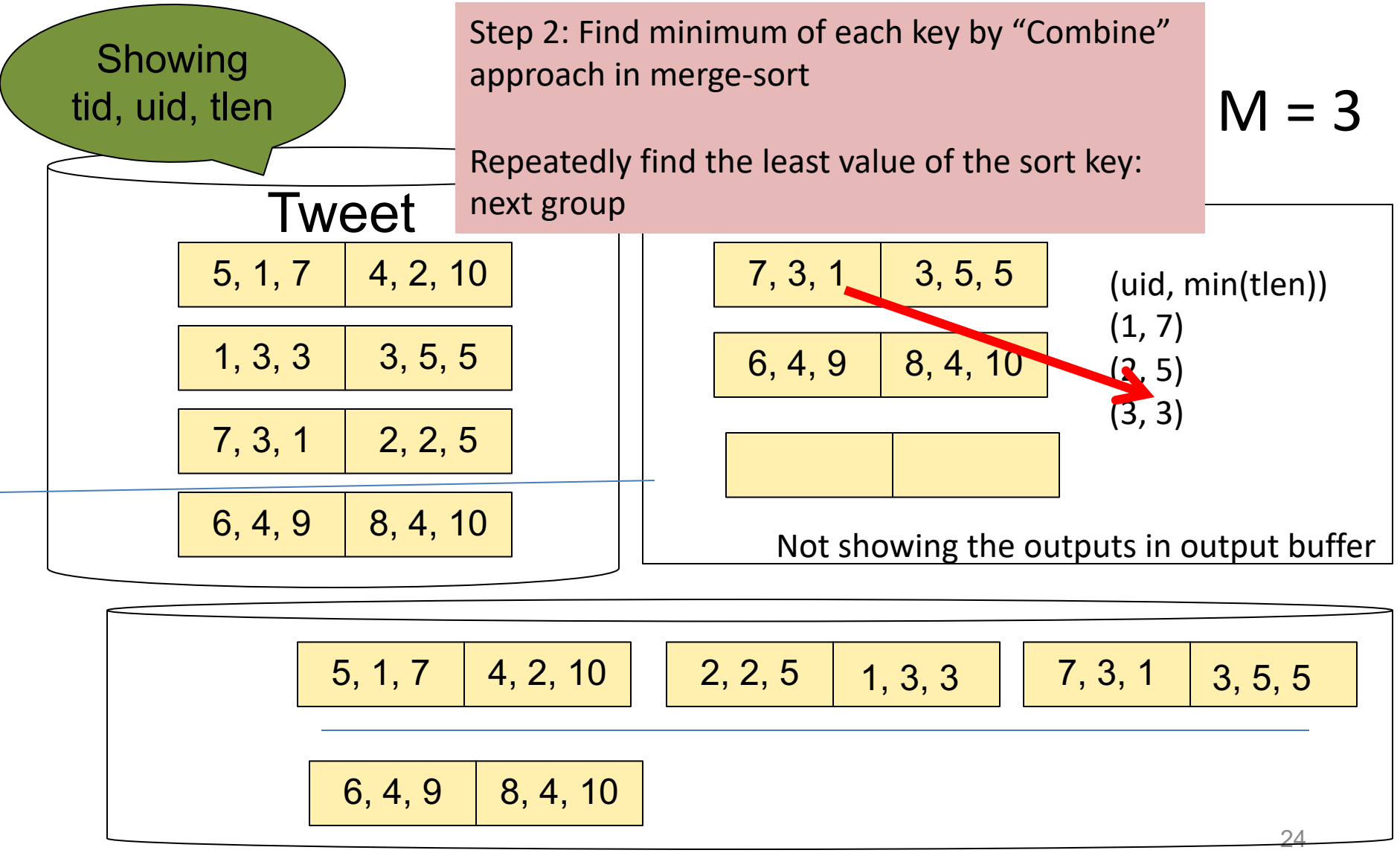
Two pass, sort-merge-based grouping



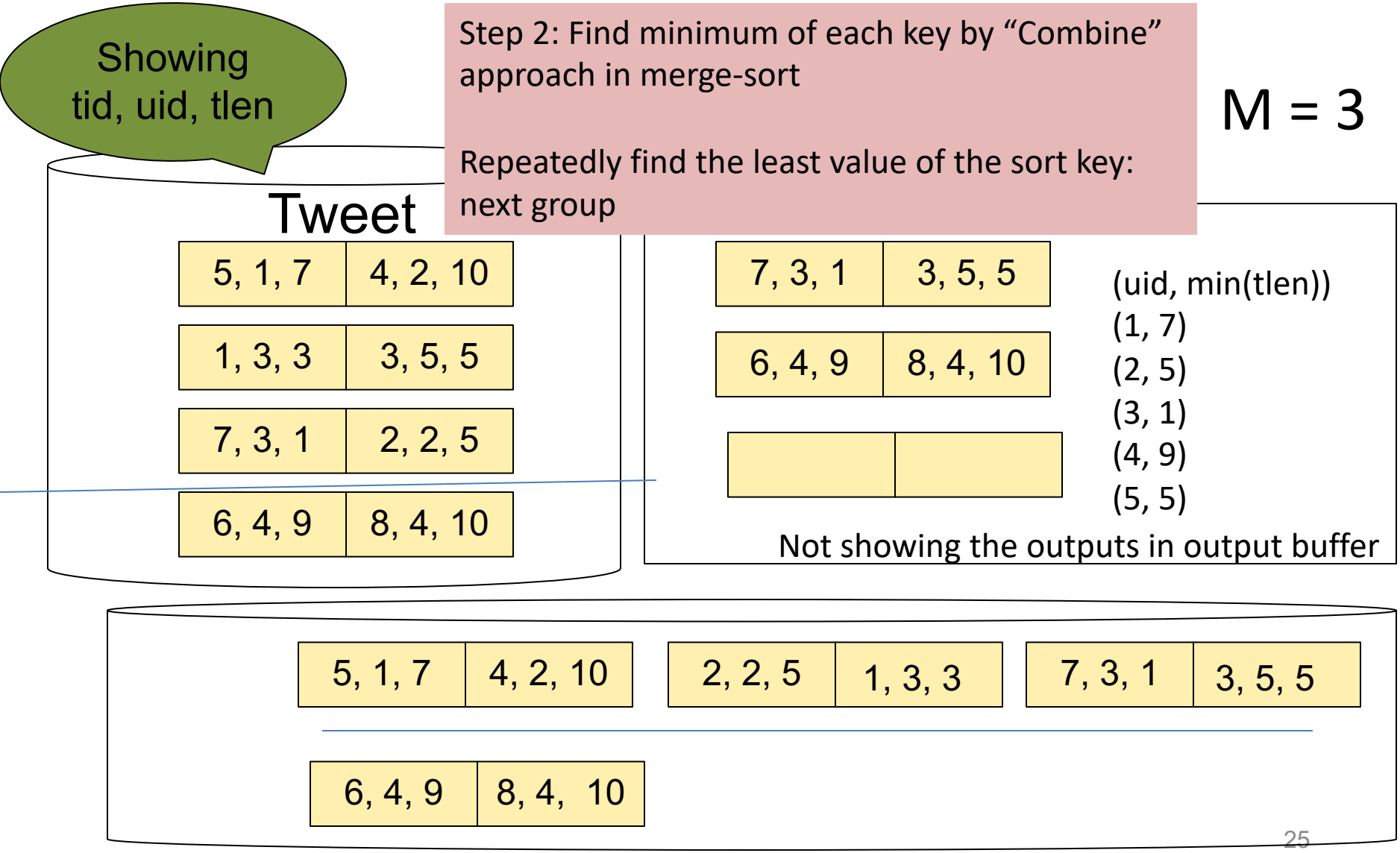
Two pass, sort-merge-based grouping



Two pass, sort-merge-based grouping



Two pass, sort-merge-based grouping



Discussion

Cost?

- 3B(R)

Assumptions?

- Need to hold one block from each run in M pages
- B(R) <= M^2

One pass vs. Two pass

- One pass:
  - smaller disk I/O cost
    - e.g. B(R) for one-pass hash-based aggregation
  - Handles smaller relations
    - e.g. B(R) <= M
- Two/Multi pass:
  - Larger disk I/O cost
    - e.g. 3B(R) for two-pass hash-based aggregation
  - Can handle larger relations
    - e.g. B(R) <= M^2

Review for Joins

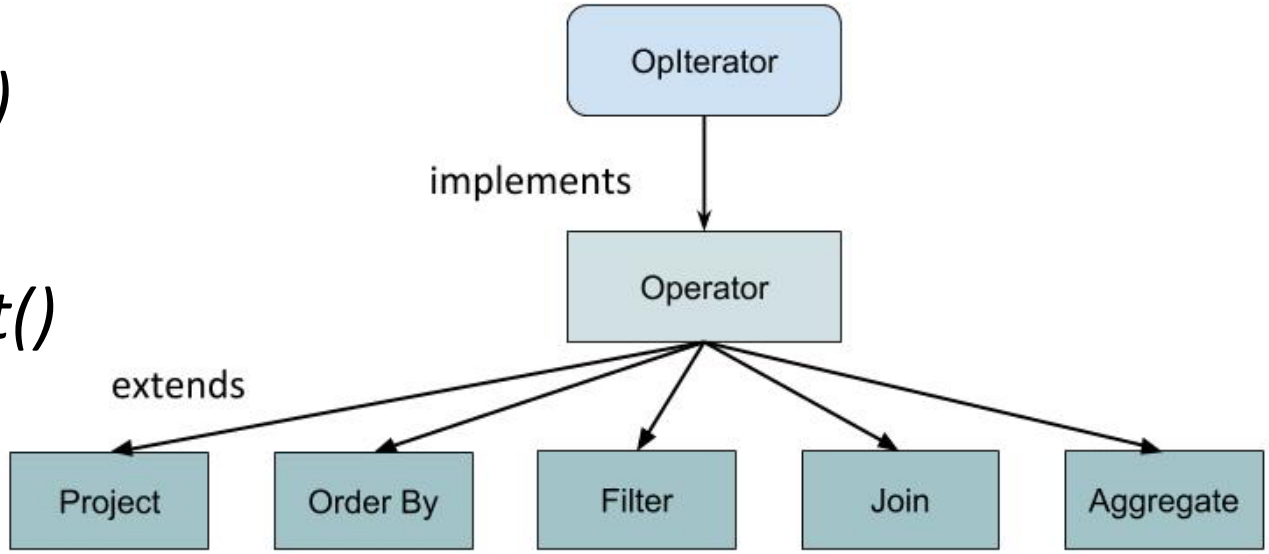
- Two-pass Hash-based Join
  - Cost: 3B(R) + 3B(S)
  - Assumption: Min(B(R), B(S)) <= M^2
- Two-pass Sort-merge-based Join
  - Implementation:
    - Cost: 5B(R) + 5B(S)
      - For R, S: sort runs/sublists (2 I/O, read + write)
      - Merge sublists to have entire R, S sorted individually (2 I/O, read + write)
      - Join by combining R and S (only read, write not counted - 1 I/O)

Homework 2

- Problem 1
  - B+ Trees (inserting/deleting/lookups)
- Problem 2
  - Operator Algorithms
- Problem 3
  - Multi-Pass Algorithms

Lab 2: Operator

- TODO: Implement Operator *Filter*, *Join* and *Aggregate*
  - *open()*
  - *close()*
  - *hasNext()*
  - *next()*
  - *fetchNext()*



Lab2: Aggregator

- TODO: Implement *IntegerAggregator* and *StringAggregator*
  - mergeTupleIntoGroup(): merge a tuple into aggregate
  - iterator(): return *a OpIterator over group aggregate results*