

CSE 444: Database Internals

Section 4: 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

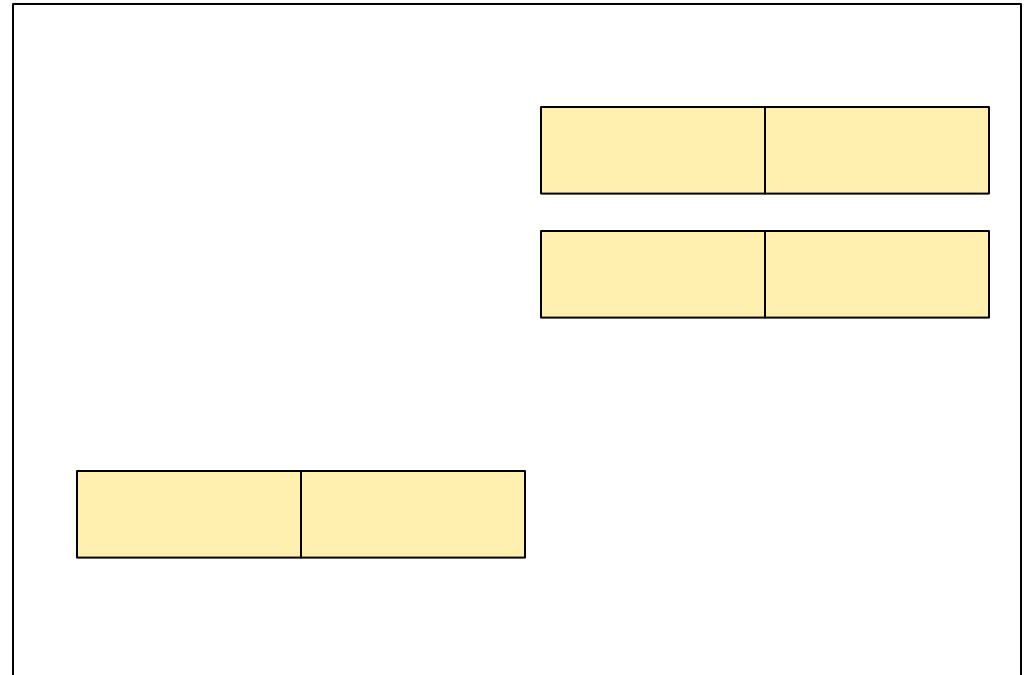
M = 3

Showing
tid, uid, tlen

Disk

Tweet

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

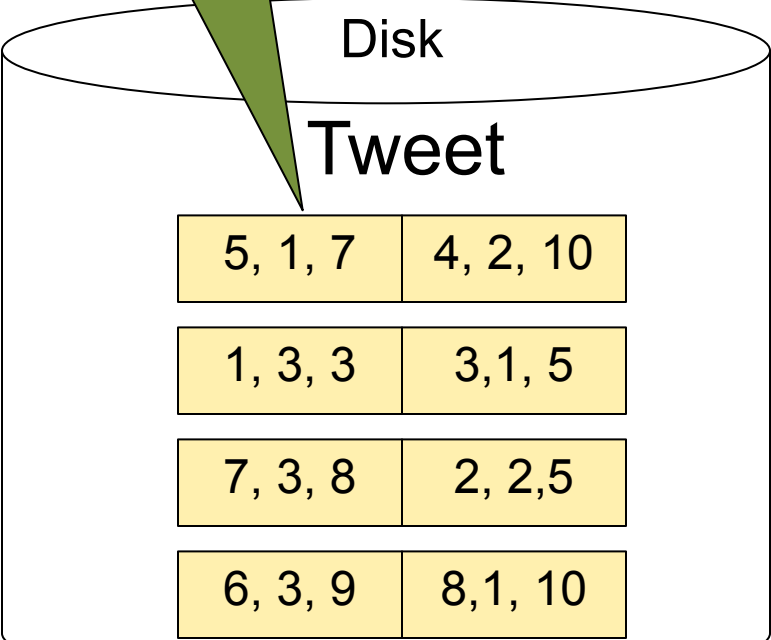


One pass, hash-based grouping

M = 3

Main memory data structure
(holds minimum for every group)

Showing
tid, uid, tlen



H = uid % 2

1, 7	
2, 10	

5, 1, 7	4, 2, 10
---------	----------

One pass, hash-based grouping

M = 3

Showing
tid, uid, tlen

Disk

Tweet

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

$H = \text{uid} \% 2$

1, 3, 3	3, 1, 5
---------	---------

1, 5	3, 3
2, 10	

Minimum
updated
from 7 to 5

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

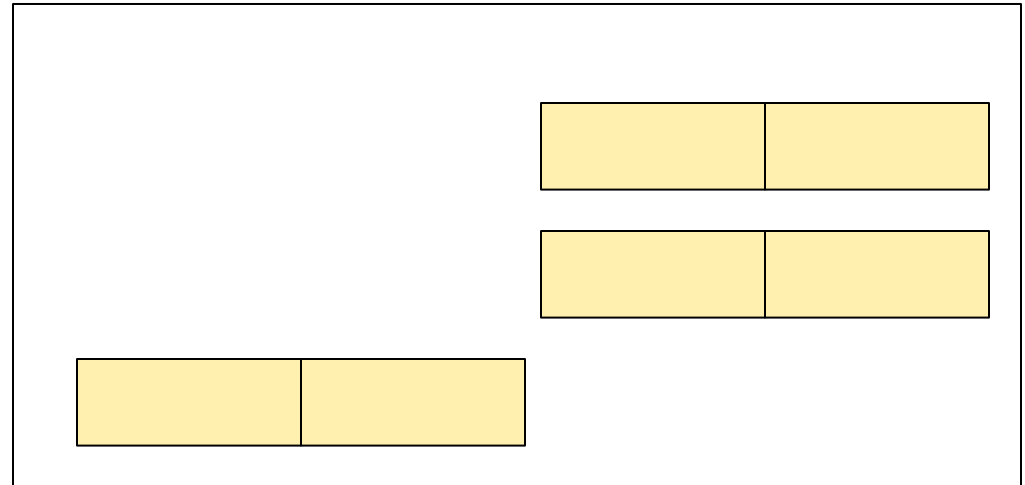
Two pass, hash-based grouping

Showing
tid, uid, tlen

$M = 3$

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



Two pass, hash-based grouping

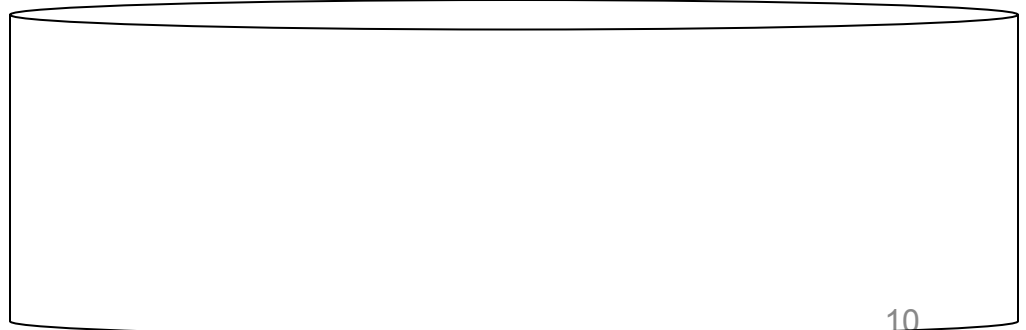
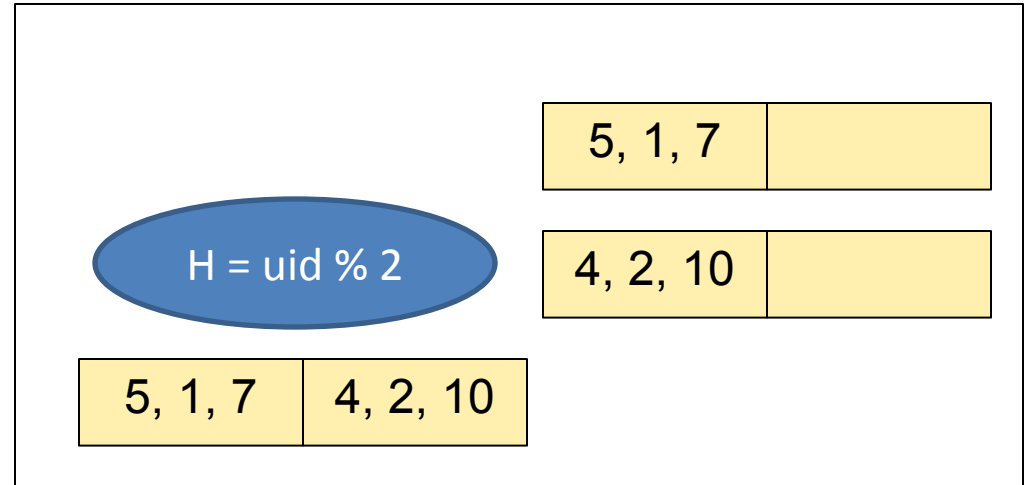
Showing
tid, uid, tlen

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



Two pass, hash-based grouping

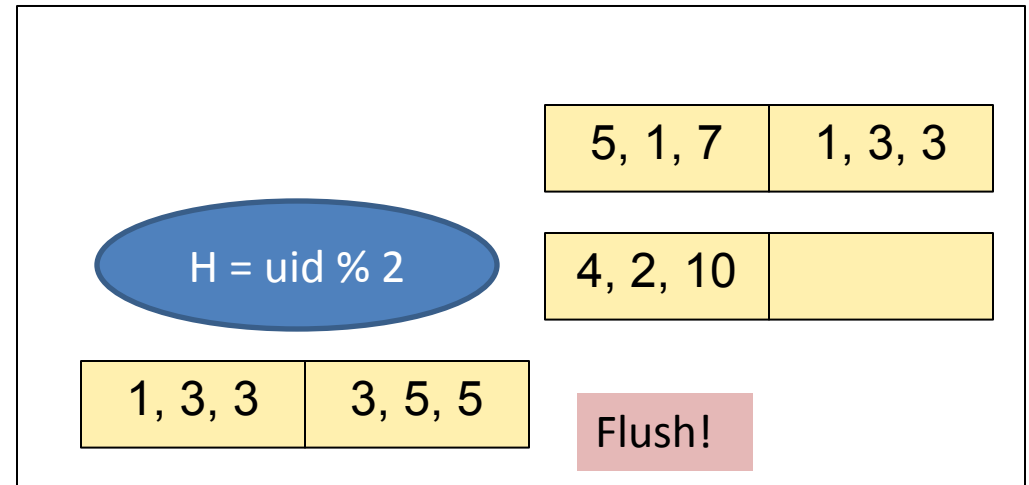
Showing
tid, uid, tlen

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



Two pass, hash-based grouping

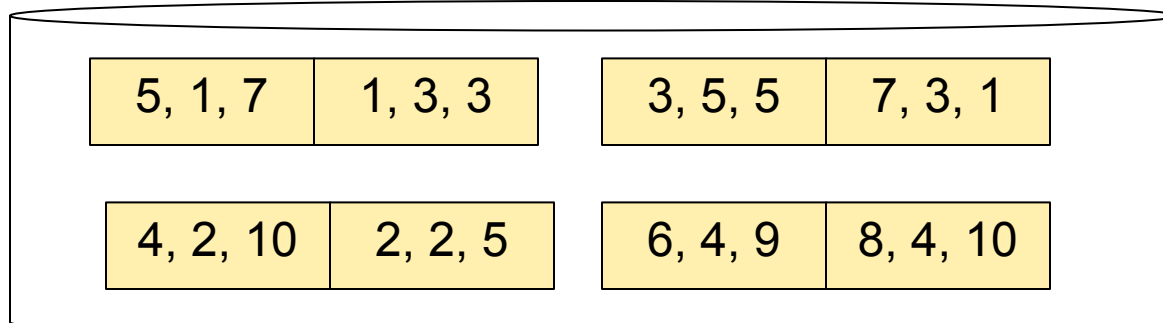
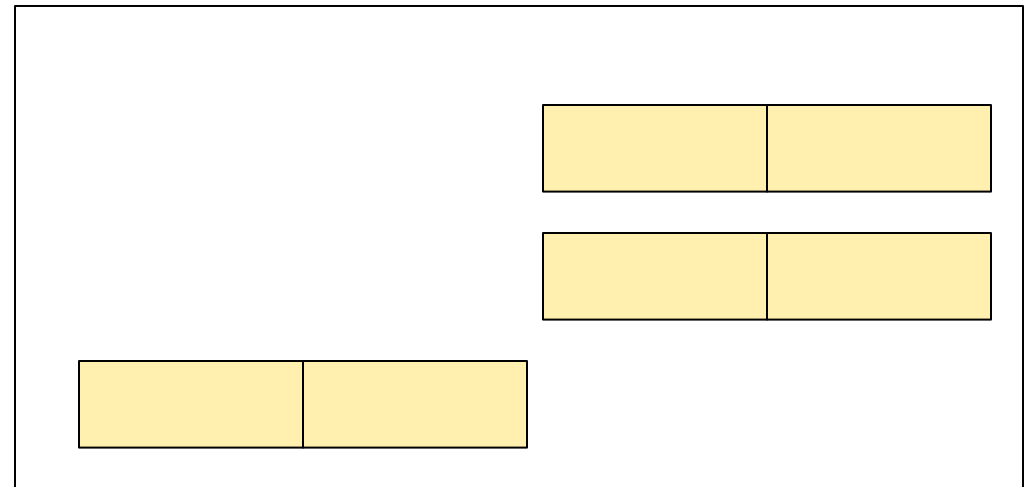
Showing
tid, uid, tlen

Final buffer and disk after pass 1

$M = 3$

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



Two pass, hash-based grouping

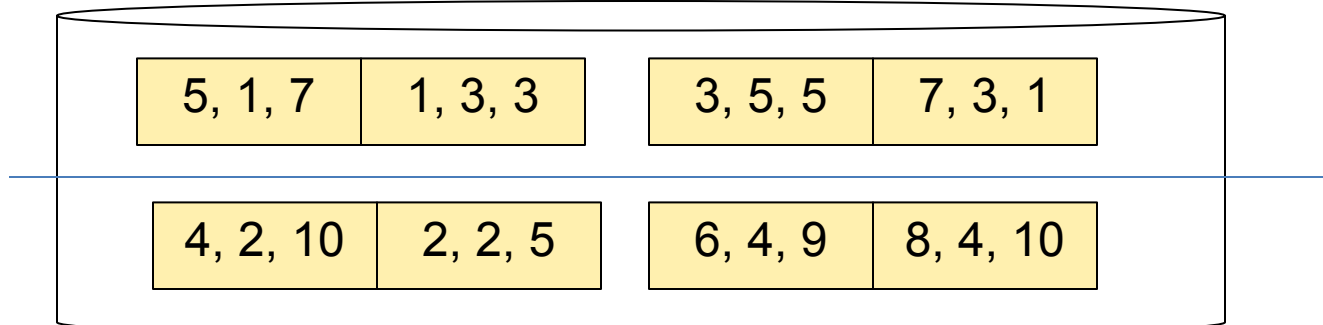
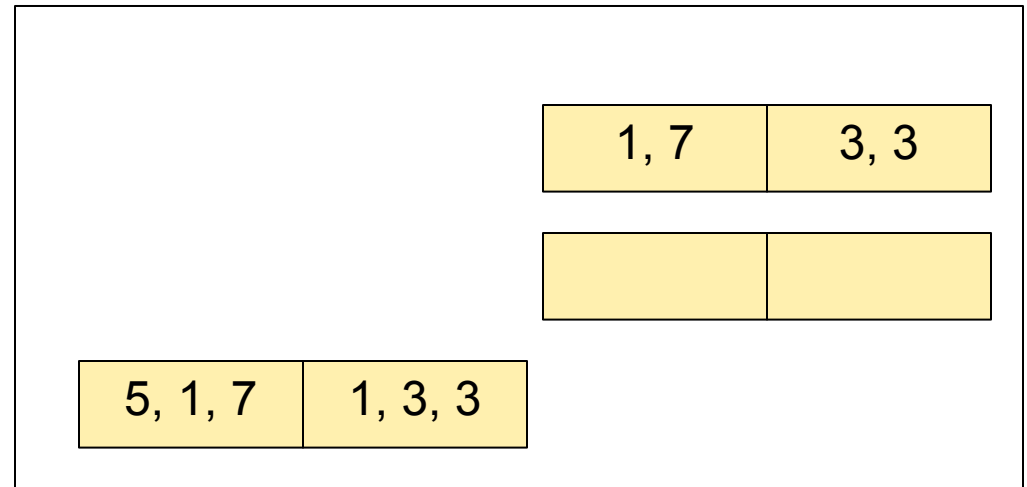
Showing
tid, uid, tlen

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

$M = 3$

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



Two pass, hash-based grouping

Showing
tid, uid, tlen

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

$M = 3$

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

Update min

1, 7	3, 3
5, 5	

3, 5, 5	7, 3, 1
---------	---------

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

Discussion

Cost?

- $3B(R)$

Assumptions?

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

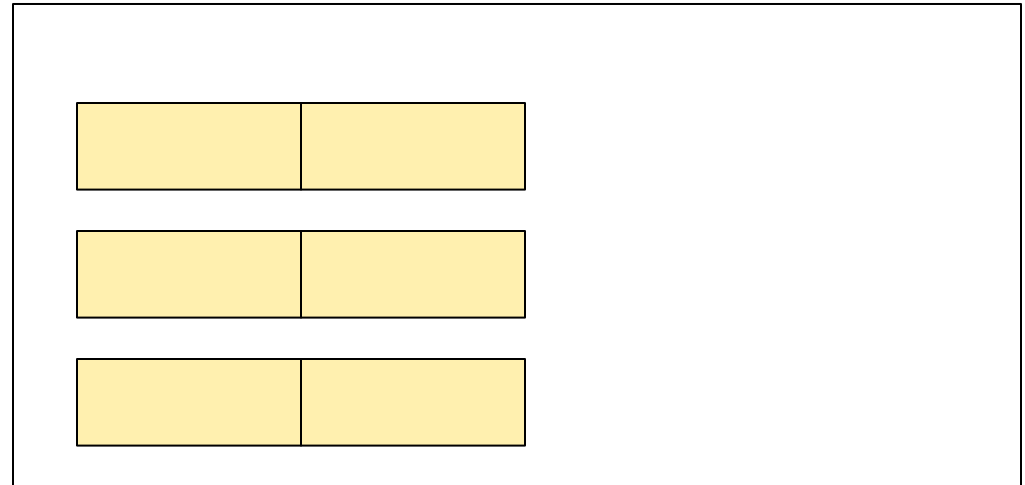
Two pass, sort-merge-based grouping

M = 3

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



Two pass, sort-merge-based grouping

Showing
tid, uid, tlen

Step 1: Divide R into M partitions
sort each partition in memory
(on group by attr = uid)
Write to disk

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

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

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

Two pass, sort-merge-based grouping

Showing
tid, uid, tlen

Step 1: Divide R into M partitions
sort each partition in memory
(on group by attr = uid)
Write to disk

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

6, 4, 9	8, 4, 10

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

Two pass, sort-merge-based grouping

Showing tid, uid, tlen

Step 2:

- Load first blocks from all runs
- Find minimum of each key by “Combine” approach in merge-sort
- Repeatedly find the least value of the sort key: next group

M = 3

Twee

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

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

(uid, min(tlen))
(1, 7)

Not showing the outputs in output buffer

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

Two pass, sort-merge-based grouping

Showing
tid, uid, tlen

Step 3: Find minimum of each key by "Combine" approach in merge-sort

M = 3

Repeatedly find the least value of the sort key:
next group

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

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

(uid, min(tlen))
(1, 7)
(2, 10)

Not showing the outputs in output buffer

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

Two pass, sort-merge-based grouping

Showing tid, uid, tlen

Step 2: Find minimum of each key by "Combine" approach in merge-sort

M = 3

Repeatedly find the least value of the sort key:
next group

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

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

(uid, min(tlen))
(1, 7)
(2, 10)

Not showing the outputs in output buffer

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

Two pass, sort-merge-based grouping

Showing tid, uid, tlen

Step 2: Find minimum of each key by "Combine" approach in merge-sort

M = 3

Repeatedly find the least value of the sort key: next group

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

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

(uid, min(tlen))
(1, 7)
(2, 10)

Not showing the outputs in output buffer

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

Two pass, sort-merge-based grouping

Showing
tid, uid, tlen

Step 2: Find minimum of each key by "Combine" approach in merge-sort

$M = 3$

Repeatedly find the least value of the sort key:
next group

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

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

(uid, min(tlen))
(1, 7)
(2, 5)
(3, 3)

Not showing the outputs in output buffer

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

Two pass, sort-merge-based grouping

Showing tid, uid, tlen

Step 2: Find minimum of each key by "Combine" approach in merge-sort

M = 3

Repeatedly find the least value of the sort key: next group

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

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

(uid, min(tlen))
 (1, 7)
 (2, 5)
 (3, 3)

Not showing the outputs in output buffer

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

Two pass, sort-merge-based grouping

Showing tid, uid, tlen

Step 2: Find minimum of each key by "Combine" approach in merge-sort

M = 3

Repeatedly find the least value of the sort key:
next group

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

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

(uid, min(tlen))
 (1, 7)
 (2, 5)
 (3, 1)
 (4, 9)
 (5, 5)

Not showing the outputs in output buffer

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

Discussion

Cost?

- $3B(R)$

Assumptions?

- Need to hold one block from each run in M pages
- $B(R) \leq M^2$

One pass vs. Two pass

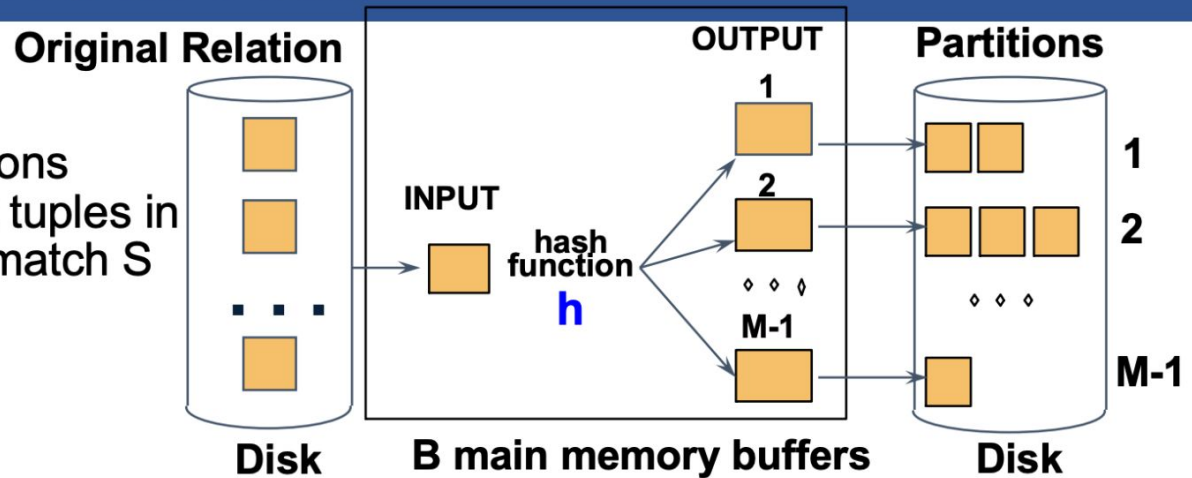
- One pass:
 - smaller disk I/O cost
 - e.g. $B(R)$ for one-pass hash-based aggregation
 - Handles smaller relations
 - e.g. $B(R) \leq M$
- Two/Multi pass:
 - Larger disk I/O cost
 - e.g. $3B(R)$ for two-pass hash-based aggregation
 - Can handle larger relations
 - e.g. $B(R) \leq M^2$

Review for Joins

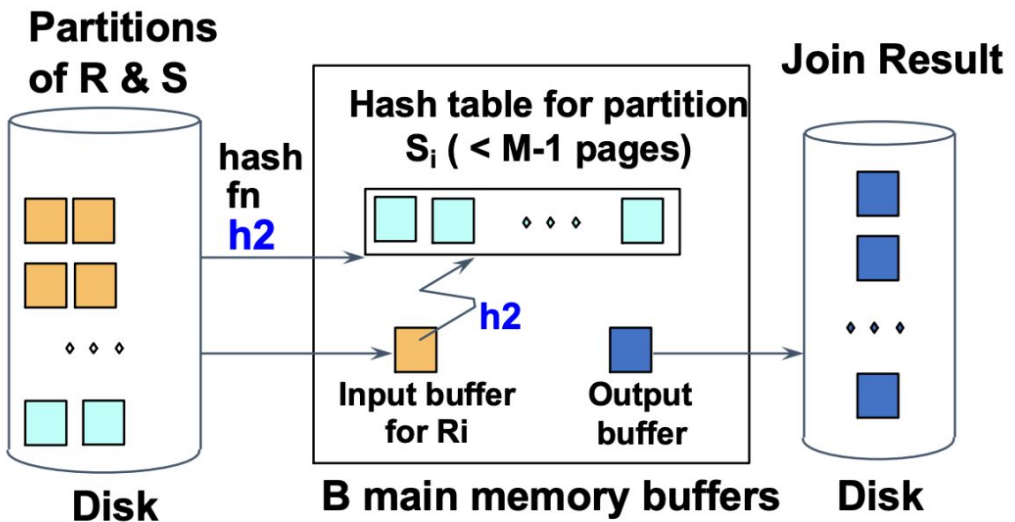
- Two-pass **Hash-based Join**
 - Cost: $3B(R) + 3B(S)$
 - Assumption: $\text{Min}(B(R), B(S)) \leq M^2$
- Two-pass **Sort-merge-based Join**
 - Implementation:
 - Cost: $3B(R) + 3B(S)$
 - For R, S: sort runs/sublists (2 I/O, read + write)
 - Join by combining all runs of R and S (only read, write not counted - 1 I/O)

Partitioned Hash-Join

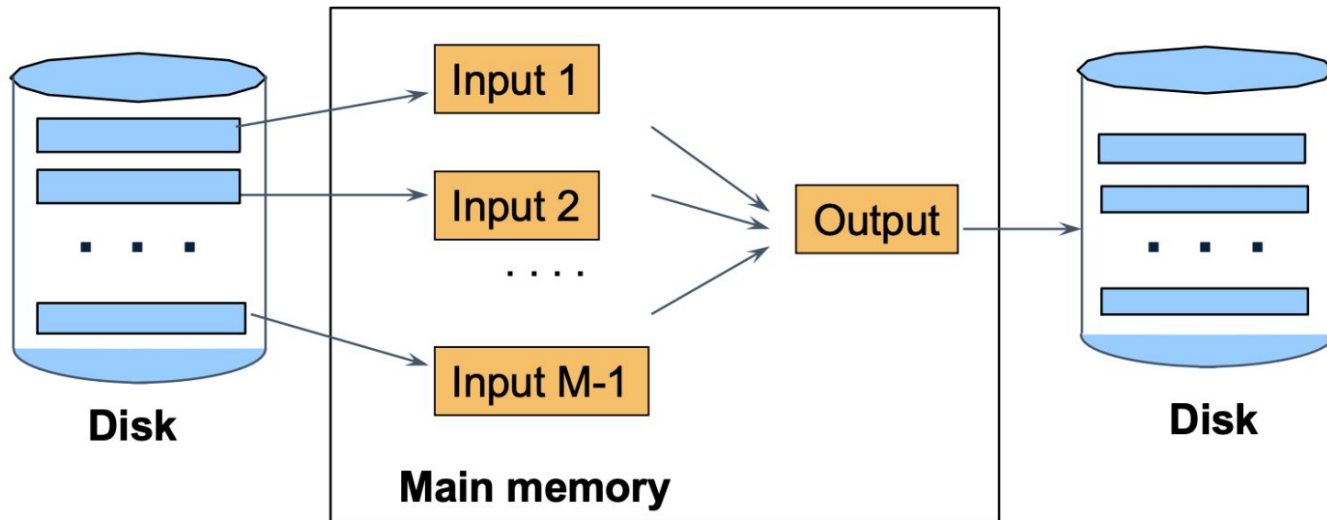
- Partition both relations using hash fn h : R tuples in partition i will only match S tuples in partition i .



- Read in a partition of R , hash it using h_2 ($\leftrightarrow h!$). Scan matching partition of S , search for matches.



Merge-Join



$M_1 = B(R)/M$ runs for R

$M_2 = B(S)/M$ runs for S

Merge-join $M_1 + M_2$ runs;

need $M_1 + M_2 \leq M$ to process all runs

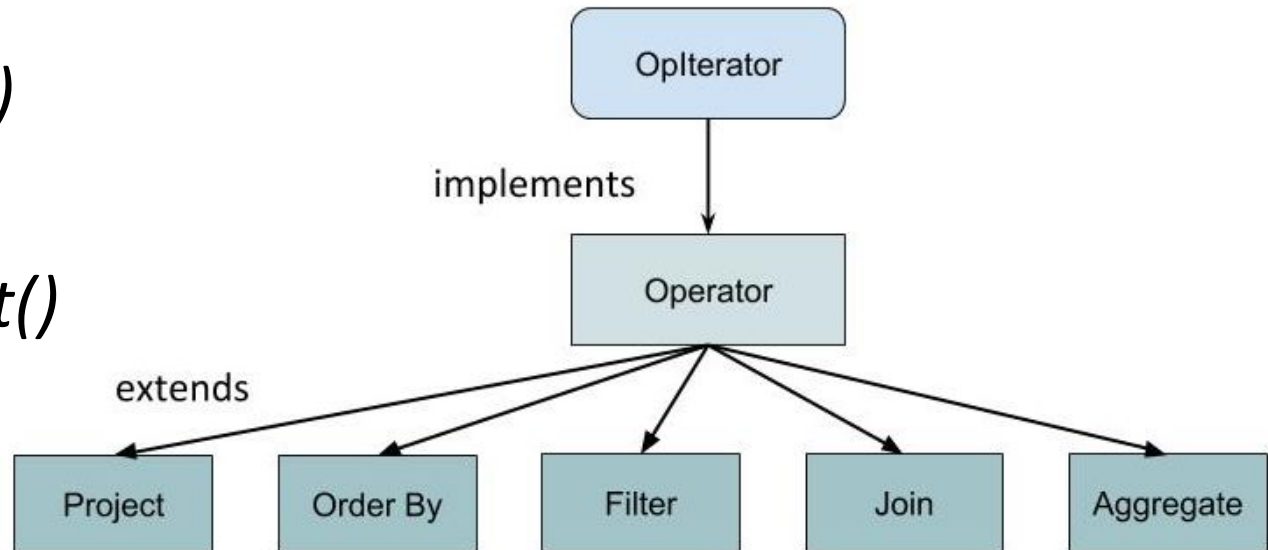
i.e. $B(R) + B(S) \leq M^2$

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 Oplterator over group aggregate results*

