

CS150A Database

Lu Sun

School of Information Science and Technology

ShanghaiTech University

Oct. 18, 2022

Today:

- Iterators and Joins:
 - Iterator Interface & Examples
 - Join operators

Readings:

- Database Management Systems (DBMS), Chapters 12 and 14

Iterators

DBMS 12

Recall from Last Lecture

query optimizer

SQL Query

```
SELECT S.name
FROM Reserves R, Sailors S
WHERE R.sid = S.sid
AND R.bid = 100
AND S.rating > 5
```

Query Parser
& Optimizer

Relational Algebra

$$\pi_{S.name}(\sigma_{R.bid=100 \wedge S.rating>5}(Reserves \bowtie_{R.sid=S.sid} Sailors))$$

relation of opt associated with implementation (iterator)

Optimized (Physical) Query Plan:

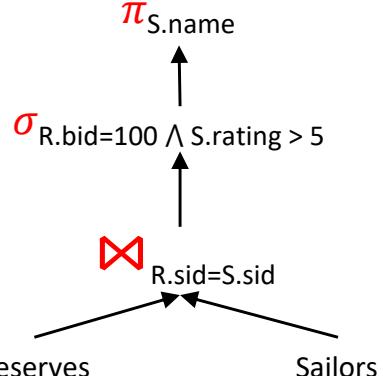
On-the-fly Project Iterator

On-the-fly Select Iterator

Indexed Nested Loop Join Iterator

Heap Scan Iterator

(Logical) Query Plan:

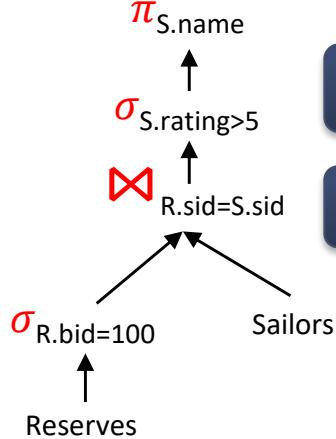


Equivalent to...

But actually will produce...

Operator Code

B+-Tree
Indexed Scan
Iterator



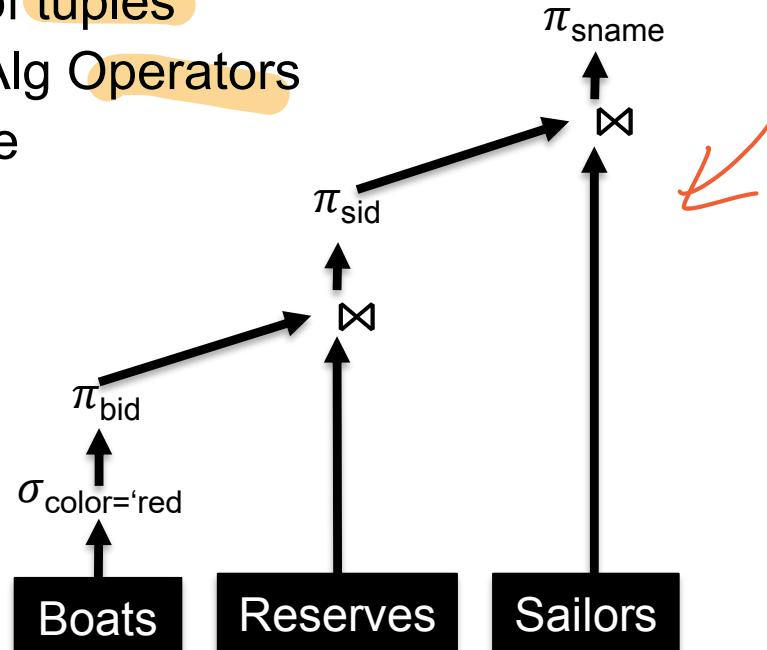
Relational Operators and Query Plans

$$\pi_{\text{sname}}(\pi_{\text{sid}}(\pi_{\text{bid}}(\sigma_{\text{color}=\text{'red'}}(\text{Boats})) \bowtie \text{Res}) \bowtie \text{Sailors})$$

- Query plan

- Edges encode “flow” of tuples
- Vertices = Relational Alg Operators
- Source vertices = table access operators ...

< data flow graph >

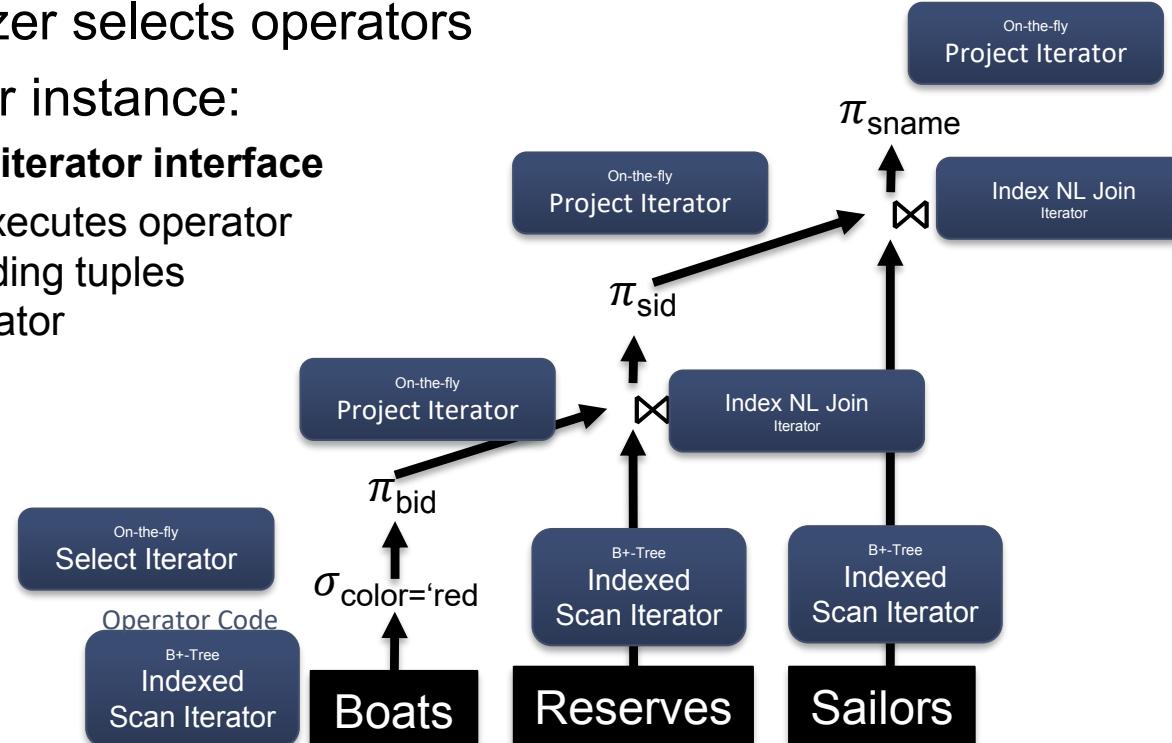


- Also called dataflow graph

Query Executor Instantiates Operators

$$\pi_{\text{sname}}(\pi_{\text{sid}}(\pi_{\text{bid}}(\sigma_{\text{color}=\text{'red'}}(\text{Boats})) \bowtie \text{Res}) \bowtie \text{Sailors})$$

- Query optimizer selects operators
- Each operator instance:
 - Implements **iterator interface**
 - Efficiently executes operator logic forwarding tuples to next operator



Iterator Interface

The relational operators implemented as subclasses of the class Iterator:

```
abstract class iterator {  
    void setup(List<Iterator> inputs); who's children  
    void init(args); set up the state  
    tuple next(); return a single tuple from iter  
    void close(); (in iter term)  
}
```

Notes:

- **Pull-based computation model**
 - e.g., Console calls **init** and **next** which propagates down graph
 - *init/next* can result in either *streaming* ("on-the-fly") or *blocking* ("batch") algorithm:
 - streaming: small, constant amount of work per call
 - blocking: does not produce output until it consumes its entire input!
- **Encapsulation**: any iterator can be input to any other! → not affected by the type of input or output
- **State**: iterators may maintain substantial "internal" state
 - e.g., hash tables, running counts, large sorted files ...
not share with others.

Example: Select (on-the-fly)

- init(predicate):

```
    child.init()  
    pred = predicate;  
    current = NULL;
```

- next(): *end of file* · *end of func*

```
    while (current != EOF && !pred(current))
```

```
        current = child.next();
```

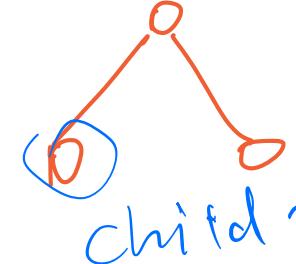
```
}
```

```
    return current;
```

- close():

```
    child.close()
```

*child
(local state)*



asking child for a new tuple

Example: Heap Scan

- `init(relation):`

```
heap = open heap file for this relation;  
cur_page = heap.first_page(); // first page  
cur_slot = cur_page.first_slot(); // first slot on that page
```

Cursor

(parent
call)

- `next():`

```
if (cur_page == NULL)  
    return EOF; // End Of Fun  
current = [cur_page, cur_slot]; // we will return this recordId  
// advance the slot  
cur_slot = cur_slot.next();  
if (cur_slot == NULL) {  
    // advance to next page, first slot  
    cur_page = cur_page.next();  
    if (cur_page != NULL)  
        cur_slot = cur_page.first_slot();  
}  
return current;
```

init.

- `close():`

```
heap.close()
```

Blocking

Example: Sort (2-pass)

col of sort key

- `init(keys):` // all of pass 0 in init, a blocking call
`child.init()` \hookrightarrow args.

repeatedly call `child.next()` and generate the sorted runs on disk, until child gives EOF
// set up for pass 1, assumes enough buffers to merge
open each sorted run file and load into input buffer for pass 1

- `next():` // pass 1 (assumes enough buffers to merge)

output = min tuple across all buffers
if min tuple was last one in its buffer, fetch next page from that run into buffer
return output (or EOF -- "End of File" -- if no tuples remain)

- `close():`

deallocate the runs files
`child.close()`

read 1 ^{page} from the sorted file
return minimum tuple

Example: Group By on Sorted input

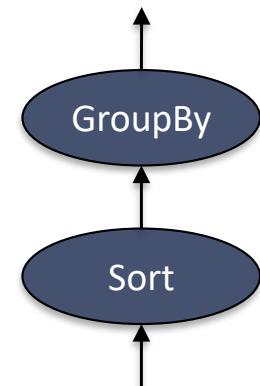
should ensure input is sorted.

(assumed here)

- `init(group_keys, aggs):`
`child.init()`
`cur_group = NULL;`
- `next():`
`result = NULL`
`do {`
 `tup = child.next();`
 `if (group(tup) != cur_group) { // New group!`
 - `if (cur_group != NULL) // Form a result for current group`
 `result = [cur_group, final() of all aggs]`
 - `cur_group = group(tup); → Make cur to the next group to ensure GB by func`
 `call init() on all the aggs`
- `}`
`else` call merge(tup) on all the aggs multiple aggs
`} while (!result);`
`return result; ↴ formed result`
- `close():`
`child.close()`
of a group

aggregational

agg_type	state	init	merge(x)	final
COUNT	count	0	count ++	count
SUM	sum	0	sum += x	sum
AVG	[count, sum]	[0, 0]	[count++, sum+=x]	sum / count
MIN	min	+infinity	min > x ? x : min	min



tuple from

fetch enough children

make cur to the next group to ensure GB by func

call merge(tup) on all the aggs multiple aggs

} while (!result);

return result; ↴ formed result

close():

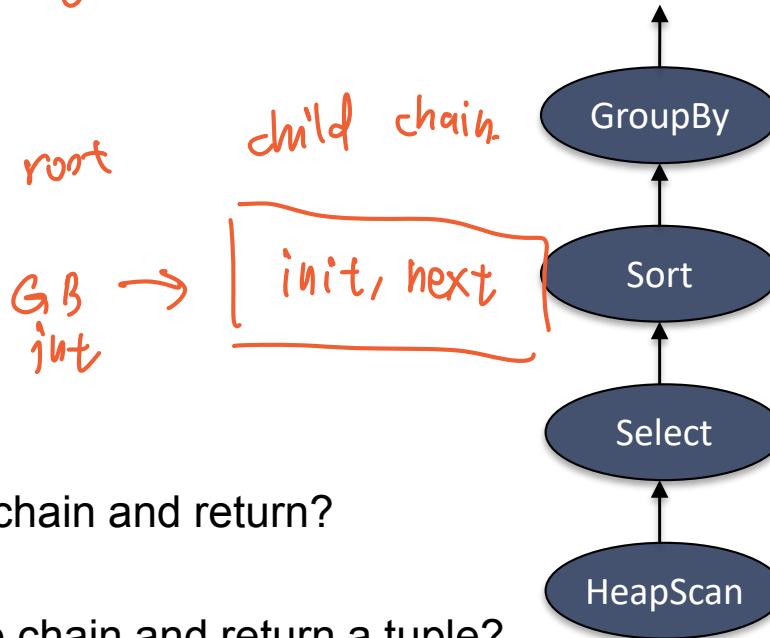
child.close()

of a group

only keep 1 tuple, memory - efficient

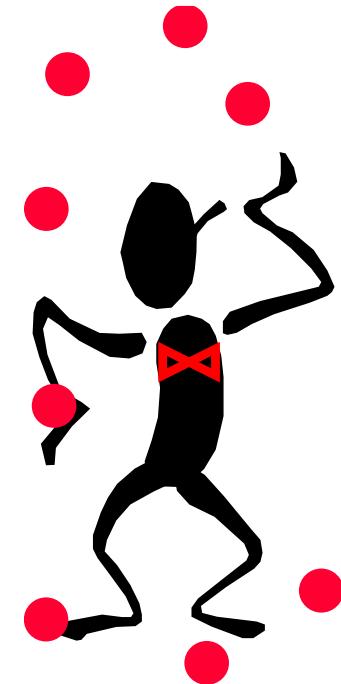
A Full Query Plan

- A Query Plan is Single-threaded!
- Trace the calls:
 - Call init() on the root GroupBy
 - How does init() recurse down the chain and return?
 - call next() on root
 - How does next() recurse down the chain and return a tuple?
- Note how the blocking operator (sort) interacts with the other, streaming operators
- Note how we **don't store** operator output on disk; tuples stream through the plan's call stack (`next()`)
 - Some operators like Sort use disk internally **run files**.



Join Operators

DBMS 14.4



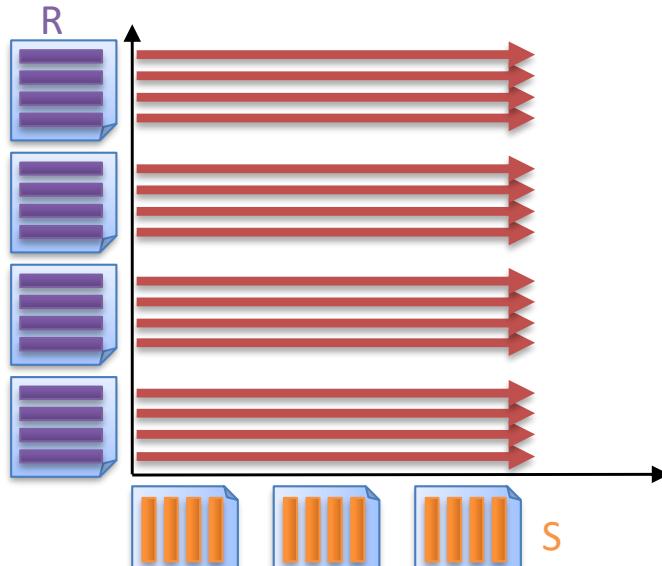
Schema for Examples

- Cost Notation
 - $[R]$: the number of pages to store R
 - p_R : number of records per page of R
 - $|R|$: the cardinality (number of records) of R
 - $|R| = p_R * [R]$
- Reserves (sid: int, bid: int, day: date, rname: string)
 - $[R]=1000$, $p_R=100$, $|R| = 100,000$
- Sailors (sid: int, sname: string, rating: int, age: real)
 - $[S]=500$, $p_S=80$, $|S| = 40,000$

Simple Nested Loops Join

```
foreach record r in R do  
    foreach record s in S do  
        if θ(ri, sj) then add <ri, sj> to result buffer
```

Note: for simplicity we do not present iterator implementations for the join algorithms.



$$[R] = 1000, p_R = 100, |R| = 100,000$$
$$[S] = 500, p_S = 80, |S| = 40,000$$

Cost:

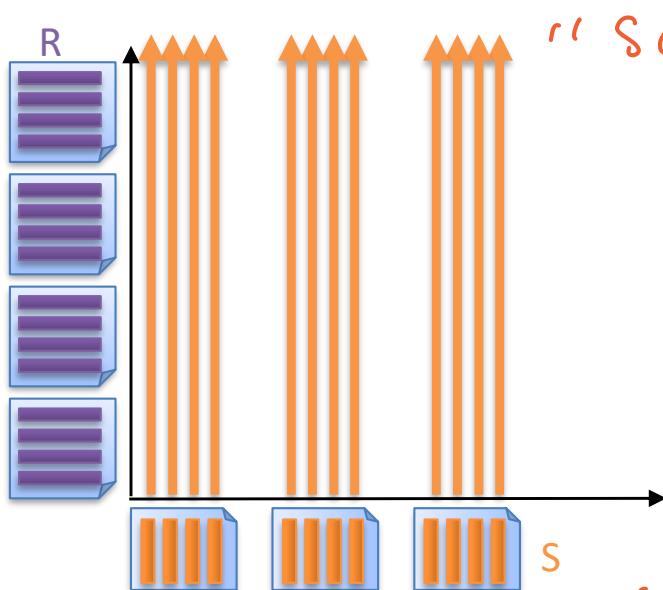
$$[R] + |R|[S]$$
$$= 50,001,000$$

of R records
of R page + card(R) × #S pages¹⁴

Changing the Join Order

ORDER matters.

```
foreach record s in S do  
    foreach record r in R do  
        if θ(ri, sj) then add <ri, sj> to result buffer
```



“Scan once”

$$[R] = 1000, p_R = 100, |R| = 100,000$$

$$[S] = 500, p_S = 80, |S| = 40,000$$

Cost:

$$\begin{aligned}[S] + |S|[R] \\ = 40,000,500 \\ \text{vs. } 50,001,000\end{aligned}$$

For each tuple of S

Scan R

Page Nested Loop Join (Another Method)

for each rpage in R:

 for each spage in S:

 for each rtuple in rpage:

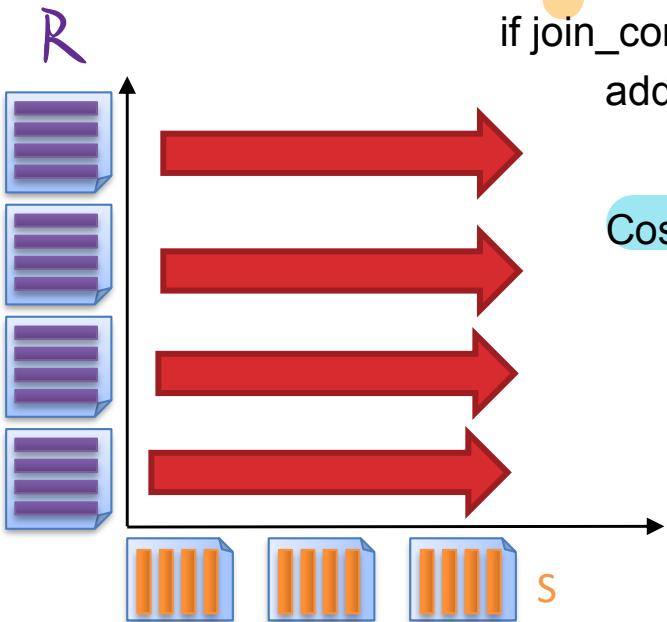
 for each stuple in spage.

 if join_condition(rtuple, stuple):

 add <rtuple, stuple> to result buffer

R page R page S page.

$$\begin{aligned} \text{Cost} &= [R] + ([R] * [S]) \\ &= 501,000 \end{aligned}$$



“Chunk” (batch)

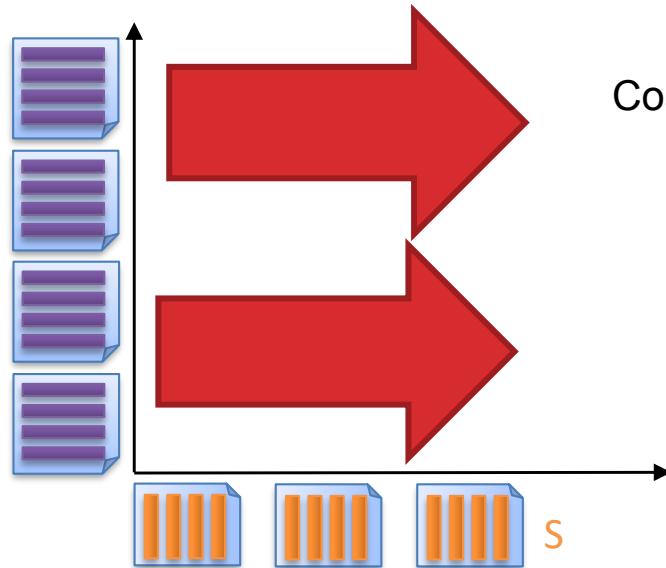
~~“Block”~~ Nested Loop Join

for each rchunk of $B-2$ pages of R:

 for each spage of S:

 for all matching tuples in spage and rchunk:

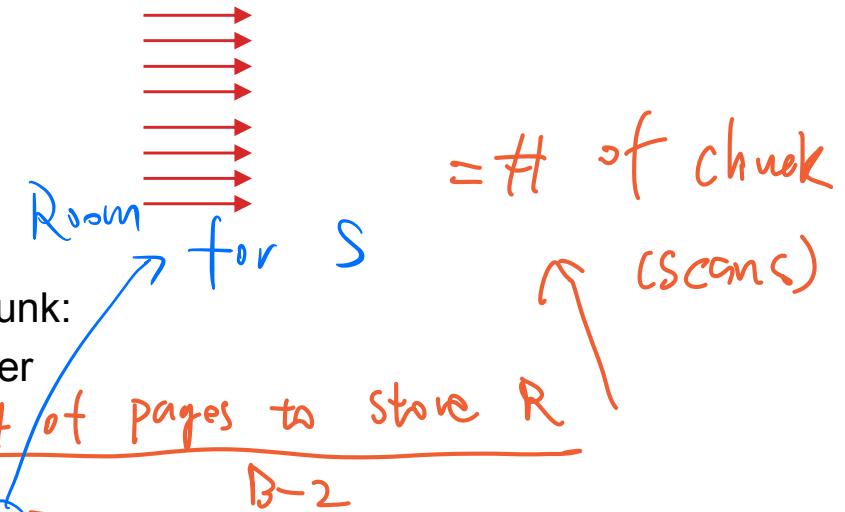
 add \langle rtuple, stuple \rangle to result buffer



$$\begin{aligned} \text{Cost} &= [R] + \lceil [R]/(B-2) \rceil * [S] \\ &= 1000 + \lceil 1000/(B-2) \rceil * 500 \\ &= 6,000 \text{ for } B=102 (\sim 100x \text{ better than Page NL!}) \end{aligned}$$

Size of B

Memory ↑ vs I/O times ↓
(R 0 O M)



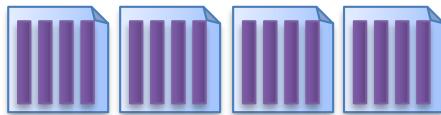
Index Nested Loops Join

foreach **tuple** r in R do

 foreach **tuple** s in S where $r_i == s_j$ do

 add $\langle r_i, s_j \rangle$ to result buffer

R

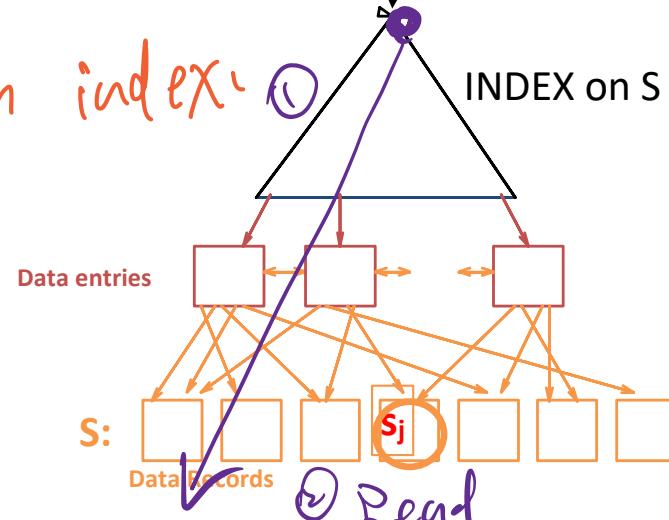


matches: col name matches

Can deal with
only equality

lookup(r_i)

by:
via lookup in index



Index Nested Loops Join Cost

foreach tuple r in R do

 foreach tuple s in S where $r_i == s_j$ do

 add $\langle r_i, s_j \rangle$ to result
 # R page

Cost = $|R| + |R| * \text{cost to find matching } S \text{ tuples}$
Card.

- If index uses Alt. 1 \rightarrow cost to traverse tree from root to leaf. (e.g., 2-4 IOs)
- For Alt. 2 or 3:
 - Cost to lookup RID(s); typically 2-4 IOs for B+Tree.
 - Cost to retrieve records from RID(s)
 - Clustered index: 1 I/O per page of matching S tuples.
 - Unclustered: up to 1 I/O per matching S tuple

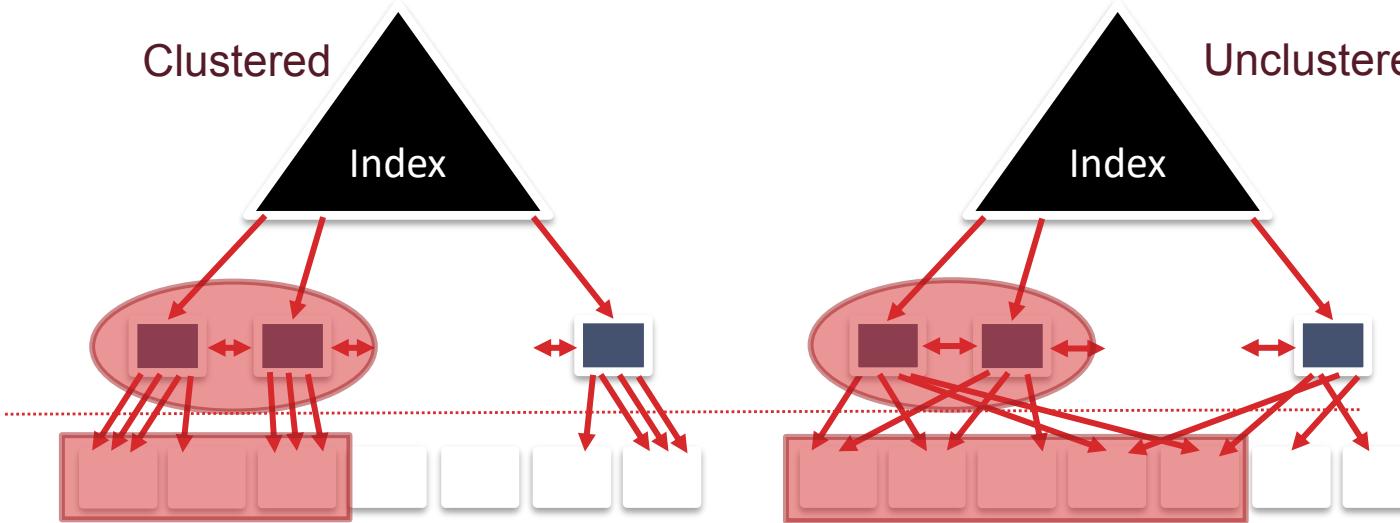
{ index alternatives ?
clustering ?

R Joins

better

Clustered vs. Unclustered Index Visualization 3

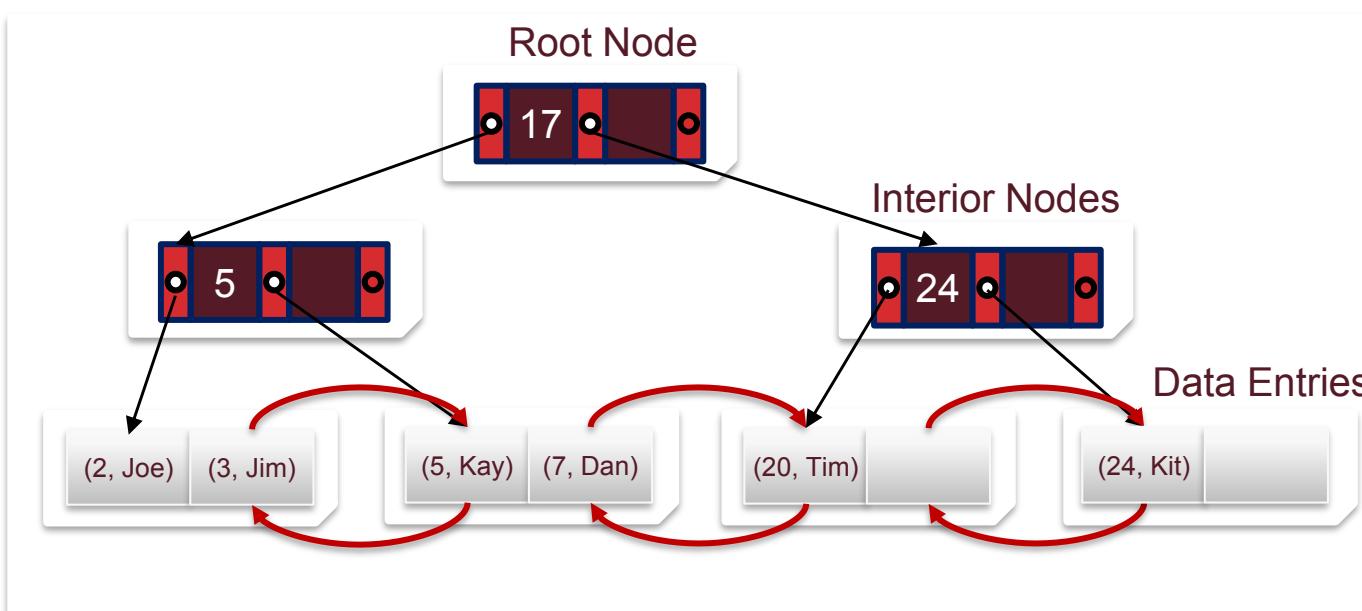
- To build a clustered index, first sort the heap file
 - Leave some free space on each block for future inserts
 - Index entries direct search for data entries



apdx.

Alternative 1 Index (B+ Tree)

- Record contents are stored in the index file
 - No need to follow pointers



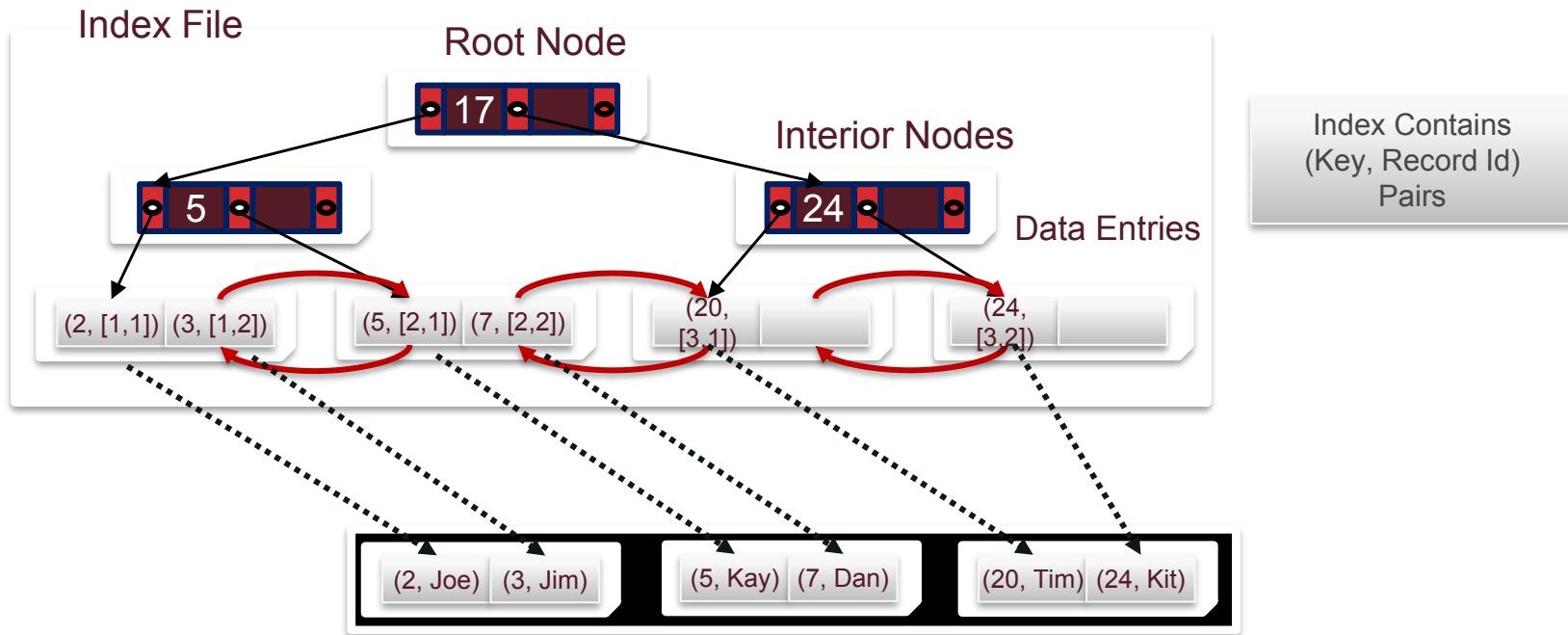
uid	name
2	Joe
3	Jim
5	Kay
7	Dan
20	Tim
24	Kit

Alternative 2 Index

reference Id.

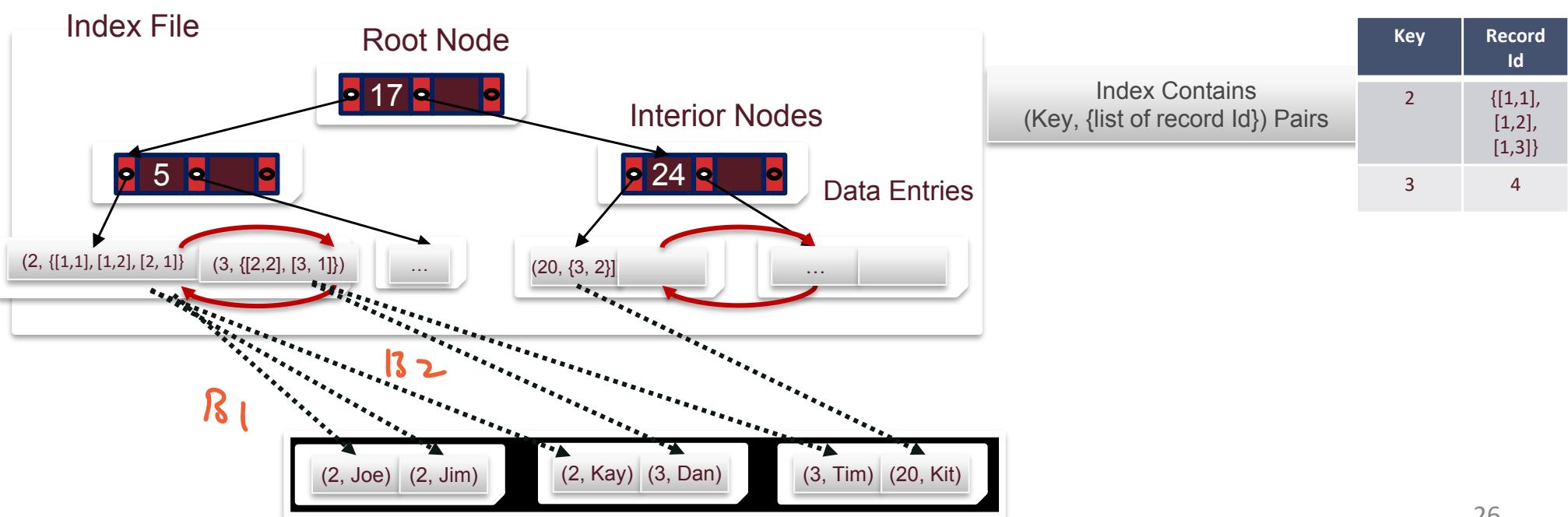
- Alternative 2: **By Reference**, $\langle k, \text{rid of matching data record} \rangle$
 - We used in slides above

uid	name
2	Joe
3	Jim
5	Kay
7	Dan
20	Tim
24	Kit



Alternative 3 Index

- Alternative 3: **By List of references, $\langle k, \text{list of rids of matching data records} \rangle$**
 - Alternative 3 more compact than alternative 2
 - For very large rid lists, **single data entry spans multiple blocks**



EXAMPLE

Index Nested Loops Join Cost, Part 2

- Reserves (*sid*: int, *bid*: int, *day*: date, *rname*: string)
 - $[R]=1000$, $p_R=100$, $|R| = 100,000$
主键
- Sailors (*sid*: int, *sname*: string, *rating*: int, *age*: real)
 - $[S]=500$, $p_S=80$, $|S| = 40,000$
 - Index on sid

Index Nested Loops Join Cost, Part 3

Cost to scan
↑ R

- Unclustered Cost(R, S) = $|R| + |R| * (\text{Search} + \# \text{ matching tuples})$
- Clustered Cost(R, S): $|R| + |R| * (\text{Search} + \# \text{ of matching pages})$
- Here, sid is the primary key for Sailors, so there is exactly one matching sailor for each tuple in R
→ root: additional 1
- Unclustered B+-Tree height 2 (3 I/Os from root to leaf):
 - $R \bowtie S$: $1000 + (100,000) * (3 + 1) = 401,000$
[R] [R] tuple 提取成本, since 主键 only here.
- Clustered B+-tree height 2 (3 I/Os from root to leaf)
 - $R \bowtie S$: $1000 + (100,000) * (3 + 1) = 401,000$
so 1

Index Nested Loops Join Cost, Part 3



- Unclustered Cost(R, S) = $[R] + |R| * (\text{Search} + \text{HeapAccess})$
- Clustered Cost(R, S): $[R] + |R| * (\text{Search}) + \text{distinct_vals}(R) * \text{HeapAccess}$
- Unclustered B+-Tree height 2 (3 I/Os from root to leaf):
 - $R \bowtie S$: $1000 + (100,000)*4 = 401,000$
- Clustered B+-tree height 2 (3 I/Os from root to leaf)
 - $R \bowtie S$: $1000 + (100,000)*3 + \text{distinct_vals}(R) = 301,000 + X < 100,000$

Sort-Merge Join

SMJ

- Requires equality predicate:
 - Equi-Joins & Natural Joins
- Two Stages:
 - Sort tuples in R and S by join key
 - All tuples with same key in consecutive order
 - Input might already be sorted ... why?
 - Join Pass: Merge-scan the sorted partitions and emit tuples that match

Sort-Merge Join, Part 1

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of "block" of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

Marked

R S

sid	sname	sid	bid
22	dustin	28	103
28	yuppy	28	104
31	lubber	31	101
31	lubber2	31	102
44	guppy	42	142
58	rusty	58	107

assume: already sorted < sid here)

Sort-Merge Join, Part 2

```
do {  
    if (!mark) {  
        while (r < s) { advance r } advance the smaller  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s → marker ≠ F  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

The diagram illustrates the sort-merge join process. It shows two tables: a left table with columns sid and sname, and a right table with columns sid and bid. A red arrow points from the left table to the right table, indicating the flow of data during the join.

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

Sort-Merge Join, Part 3

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname	sid	bid
22	dustin	28	103
28	yuppy	28	104
31	lubber	31	101
31	lubber2	31	102
44	guppy	42	142
58	rusty	58	107

Sort-Merge Join, Part 4

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

Sort-Merge Join, Part 5

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

Sort-Merge Join, Part 6

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
    }  
    return result  
}  
else {  
    reset s to mark  
    advance r  
    mark = NULL  
}  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

Sort-Merge Join, Part 7

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
    }  
    return result  
}  
else {  
    reset s to mark  
    advance r  
    mark = NULL  
}  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

Sort-Merge Join, Part 8

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103

Sort-Merge Join, Part 9

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103

Sort-Merge Join, Part 10

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103

Sort-Merge Join, Part 11

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103

Sort-Merge Join, Part 12

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	sname	bid
28	yuppy	103
28		104
31		101
31		102
42		142
58		107

Sort-Merge Join, Part 13

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 14

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 15

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of "block" of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 16

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 17

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 18

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 19

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 20

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of "block" of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 21

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 22

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 23

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 24

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 25

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
    }  
    return result  
}  
else {  
    reset s to mark  
    advance r  
    mark = NULL  
}  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104

Sort-Merge Join, Part 26

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101

Sort-Merge Join, Part 27

```
do {
    if (!mark) {
        while (r < s) { advance r }
        while (r > s) { advance s }
        // mark start of “block” of S
        mark = s
    }
    if (r == s) {
        result = <r, s>
        advance s
        return result
    }
    else {
        reset s to mark
        advance r
        mark = NULL
    }
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101

Sort-Merge Join, Part 28

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101

Sort-Merge Join, Part 29

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101

Sort-Merge Join, Part 30

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
    }  
    return result  
}  
else {  
    reset s to mark  
    advance r  
    mark = NULL  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101

Sort-Merge Join, Part 31

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102

Sort-Merge Join, Part 32

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102

Sort-Merge Join, Part 33

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102

Sort-Merge Join, Part 34

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102

Sort-Merge Join, Part 35

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102

Sort-Merge Join, Part 36

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname	sid	bid
22	dustin	28	103
28	yuppy	28	104
31	lubber	31	101
31	lubber2	31	102
44	guppy	42	142
58	rusty	58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102

Sort-Merge Join, Part 37

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

The diagram illustrates the sort-merge join process. It shows two sorted tables and the merging process:

- Table 1 (Left):** sid (blue header), sname (blue header). Rows: (22, dustin), (28, yuppy), (31, lubber), (31, lubber2), (44, guppy), (58, rusty).
- Table 2 (Right):** sid (orange header), bid (orange header). Rows: (28, 103), (28, 104), (31, 101), (31, 102), (42, 142), (58, 107).
- Merge Process:** A red arrow points from Table 1 to Table 2, indicating the merging of rows from Table 1 with matching sid values from Table 2.

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102

Sort-Merge Join, Part 38

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102

Sort-Merge Join, Part 39

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102

Sort-Merge Join, Part 40

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102

Sort-Merge Join, Part 41

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
    }  
    return result  
}  
else {  
    reset s to mark  
    advance r  
    mark = NULL  
}  
}
```

sid	sname	sid	bid
22	dustin	28	103
28	yuppy	28	104
31	lubber	31	101
31	lubber2	31	102
44	guppy	42	142
58	rusty	58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102

Sort-Merge Join, Part 42

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101

Sort-Merge Join, Part 43

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101

Sort-Merge Join, Part 44

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101

Sort-Merge Join, Part 45

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
    }  
    return result  
}  
else {  
    reset s to mark  
    advance r  
    mark = NULL  
}
```

sid	sname	sid	bid
22	dustin	28	103
28	yuppy	28	104
31	lubber	31	101
31	lubber2	31	102
44	guppy	42	142
58	rusty	58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101

Sort-Merge Join, Part 46

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
    }  
    return result  
}  
else {  
    reset s to mark  
    advance r  
    mark = NULL  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101

Sort-Merge Join, Part 47

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 48

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 49

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 50

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 51

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 52

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 53

```
do {
    if (!mark) {
        while (r < s) { advance r }
        while (r > s) { advance s }
        // mark start of “block” of S
        mark = s
    }
    if (r == s) {
        result = <r, s>
        advance s
        return result
    }
    else {
        reset s to mark
        advance r
        mark = NULL
    }
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 54

```
do {
    if (!mark) {
        while (r < s) { advance r }
        while (r > s) { advance s }
        // mark start of "block" of S
        mark = s
    }
    if (r == s) {
        result = <r, s>
        advance s
        return result
    }
    else {
        reset s to mark
        advance r
        mark = NULL
    }
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 55

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 56

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 57

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 57

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 58

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 59

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 60

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname	sid	bid
22	dustin	28	103
28	yuppy	28	104
31	lubber	31	101
31	lubber2	31	102
44	guppy	42	142
58	rusty	58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 61

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname	bid
22	dustin	
28	yuppy	
31	lubber	
31	lubber2	
44	guppy	
58	rusty	

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 62

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 63

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
    }  
    return result  
}  
else {  
    reset s to mark  
    advance r  
    mark = NULL  
}
```

sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 64

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
    }  
    return result  
}  
else {  
    reset s to mark  
    advance r  
    mark = NULL  
}
```

sid	sname	bid
22	dustin	
28	yuppy	
31	lubber	
31	lubber2	
44	guppy	
58	rusty	

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102

Sort-Merge Join, Part 65

```
do {  
    if (!mark) {  
        while (r < s) { advance r }  
        while (r > s) { advance s }  
        // mark start of “block” of S  
        mark = s  
    }  
    if (r == s) {  
        result = <r, s>  
        advance s  
        return result  
    }  
    else {  
        reset s to mark  
        advance r  
        mark = NULL  
    }  
}
```

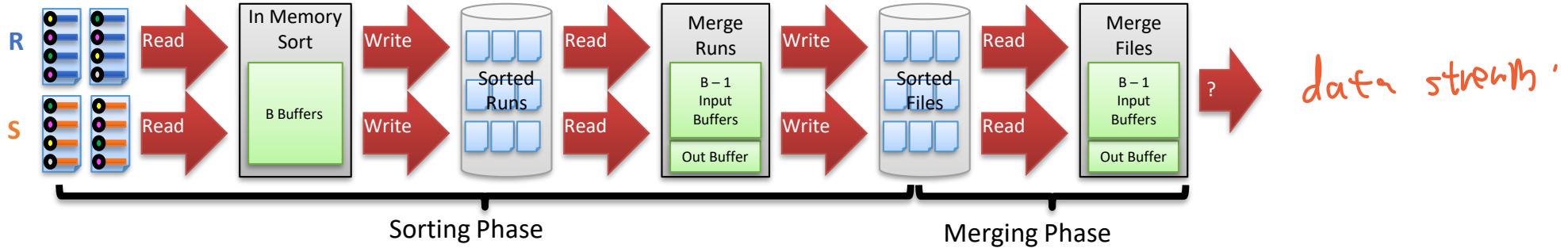
sid	sname
22	dustin
28	yuppy
31	lubber
31	lubber2
44	guppy
58	rusty

sid	bid
28	103
28	104
31	101
31	102
42	142
58	107

sid	sname	bid
28	yuppy	103
28	yuppy	104
31	lubber	101
31	lubber	102
31	lubber2	101
31	lubber2	102
58	rusty	107

only 2 pass

Cost of Sort-Merge Join



- Cost: Sort R + Sort S + $([R]+[S])$
 - But in worst case, last term could be $|R| * |S|$ (very unlikely!)
 - Q: what is worst case?
 - Question: How big does the buffer have to be to sort both R and S in two passes each?
 - Suppose buffer $B > \sqrt{(\max([R], [S]))}$
 - Both R and S can be sorted in 2 passes
 - $4*1000 + 4*500 + (1000 + 500) = 7500$
- ↓ ↓* *↓ ↓* *↓ ↓*
- ↓ ↓ ↓ ↓ ↓ ↓

Join First, Sort Later

SELECT sid, bid, sname, rname

FROM R, S

WHERE R.sid = S.sid

ORDER BY sid

Ask for sorted
in the out put

[R]=1000, p_R=100, |R| = 100,000

[S]=500, p_S=80, |S| = 40,000

B = 102

- Reserves (sid: int, bid: int, day: date, rname: string)
- Sailors (sid: int, sname: string, rating: int, age: real)
- Block NLJ
 - Join: [S] + ([S]/(B-2)) * [R]
 - Sort: ?

(A)

Join First, Sort Later Part 2

SELECT sid, bid, sname, rname

FROM R, S

WHERE R.sid = S.sid

ORDER BY sid

[R]=1000, p_R=100, |R| = 100,000

[S]=500, p_S=80, |S| = 40,000

B = 102

- Reserves (sid: int, bid: int, day: date, rname: string)
- Sailors (sid: int, sname: string, rating: int, age: real)

- Block NLJ

- Join: [S] + ([S]/(B-2))*[R] = 5,500

- Sort: $2 * [R] * (1 + \log_{B-1} [R]/B) = 4000$

if the joined result
same size as R



Sort First, Join Later

```
SELECT sid, bid, sname, rname  
FROM R, S  
WHERE R.sid = S.sid  
ORDER BY sid
```

$[R]=1000, p_R=100, |R| = 100,000$
 $[S]=500, p_S=80, |S| = 40,000$
 $B = 102$

- Reserves (sid : int, bid : int, day : date, $rname$: string)
- Sailors (sid : int, $sname$: string, $rating$: int, age : real)

Sort R: $2*[R]^*(2) = 4000$

Sort S: $2*[S]^*(2) = 2000$

$R + S = 1500$

Total = 7500

$\angle B(B-1)$

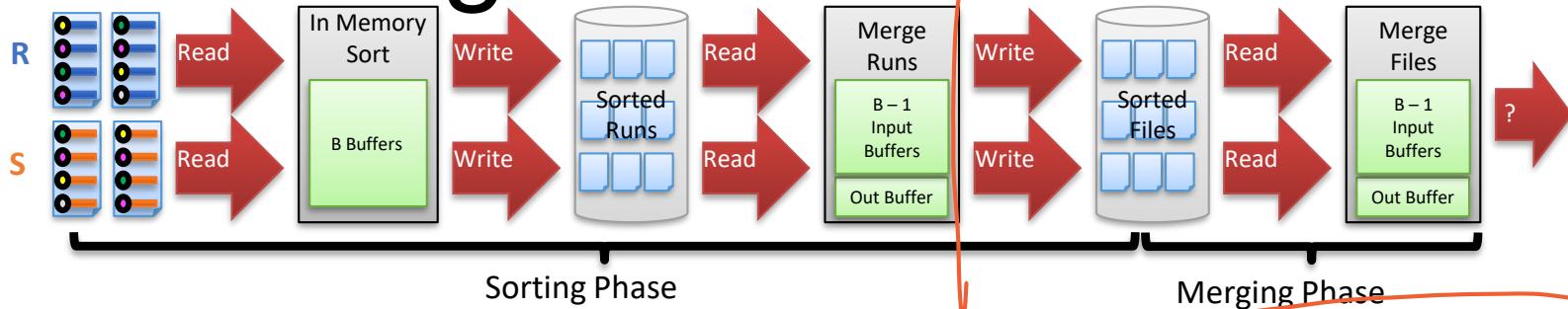


2 pass sorting

finish

I/o cost differs⁹²

Sort-Merge Refinement



- An important refinement combines last pass of merge-sort with join pass
 - Given enough buffers to accommodate all runs in R and S on the penultimate (second-to-last) pass of sorting
 - Example for 2-pass SMJ (join during the final merging pass of sort)
 - Read R and write out sorted runs (pass 0)
 - Read S and write out sorted runs (pass 0)
 - Merge R-runs and S-runs, while finding $R \bowtie S$ matches
 - 2-pass Cost = $3*[R] + 3*[S] = 3000+1500 = 4500$

Merge
and
Join

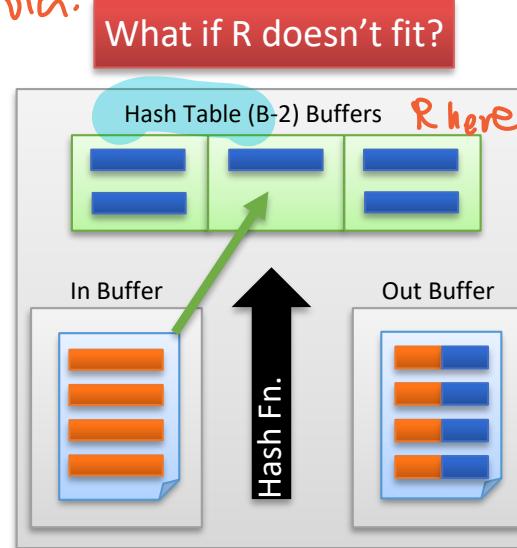
$$B(B-1) > \sqrt{R+S}$$

Naïve in Memory Hash Join

only for equal Join.

- Requires equality predicate:
 - Works for **Equi-Joins & Natural Joins**
- Assume **R** is smaller relation
 - Require **R** to fit in memory
- Simple algorithm:
 - Load all **R** into hash table
 - Scan **S** and probe **R**
- Memory requirements?
 - $R < (B-2) * \text{hash_fill}$

hash factor
(cost of hash table)



Properties that help

sub-queries

- $\sigma_{\text{sid}=4 \vee \text{sid}=6}(\text{R} \bowtie_{\text{sid}} \text{S}) = \sigma_{\text{sid}=4}(\text{R} \bowtie_{\text{sid}} \text{S}) \cup \sigma_{\text{sid}=6}(\text{R} \bowtie_{\text{sid}} \text{S})$
- Can Decompose Into Smaller “Partial Joins”
- $\text{R} \bowtie_{\text{sid}} \text{S} = \bigcup (\sigma_{\text{hash(sid)}}(\text{R}) \bowtie_{\text{sid}} \sigma_{\text{hash(sid)}}(\text{S}))$ for each sid.
- Pick a hash function so that $\sigma_{\text{hash(sid)}}(\text{R})$ fits in memory!

each hash bucket batches of sids.

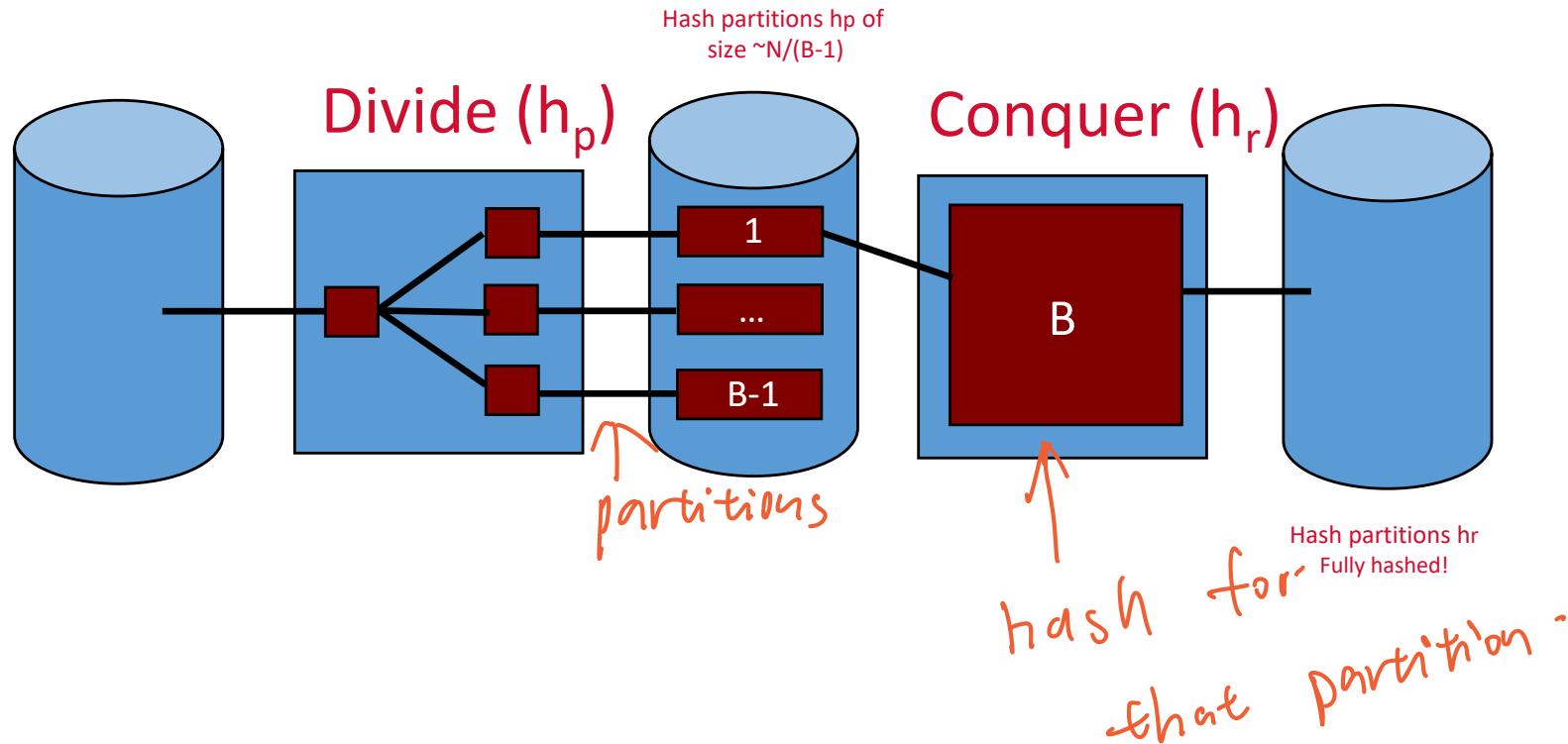
(batch of values)

fit in memory

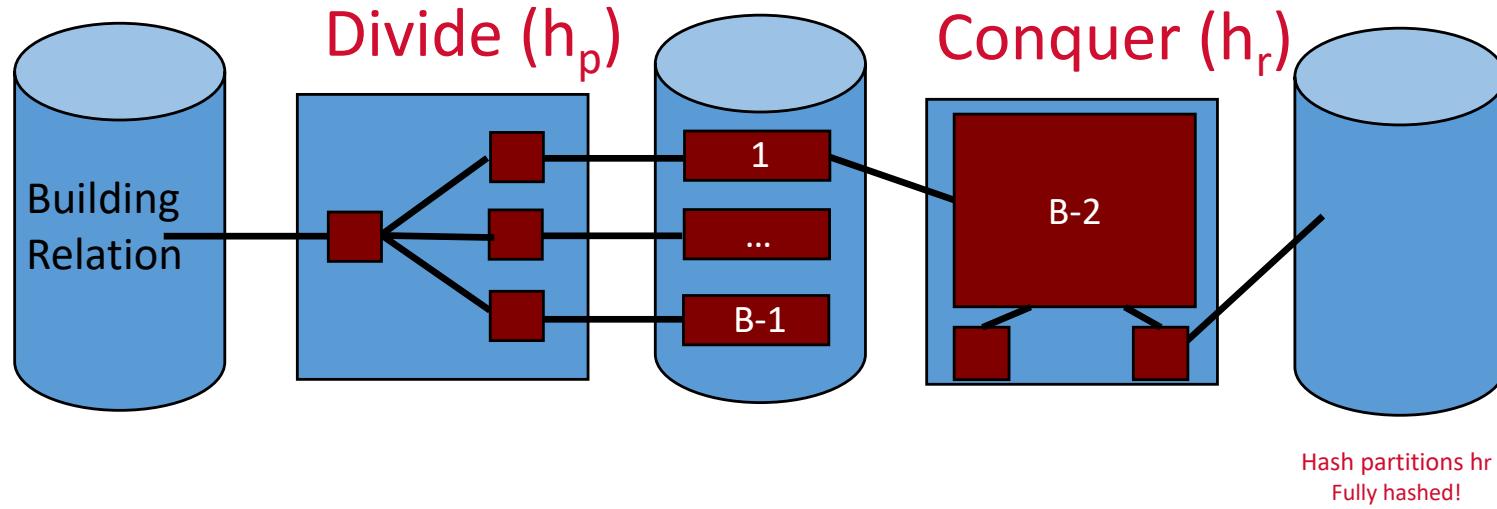
Grace Hash Join

- Requires equality predicate:
 - **Equi-Joins & Natural Joins**
- Two Stages:
 - **Partition** tuples from **R** and **S** by join key and store on scratch disk
 - all tuples for a given key **in same partition**
 - **Build & Probe** a separate hash table **for each partition** (like in Naïve Hash)
 - Assume **partition** of smaller rel. fits in memory
 - Recurse if necessary...

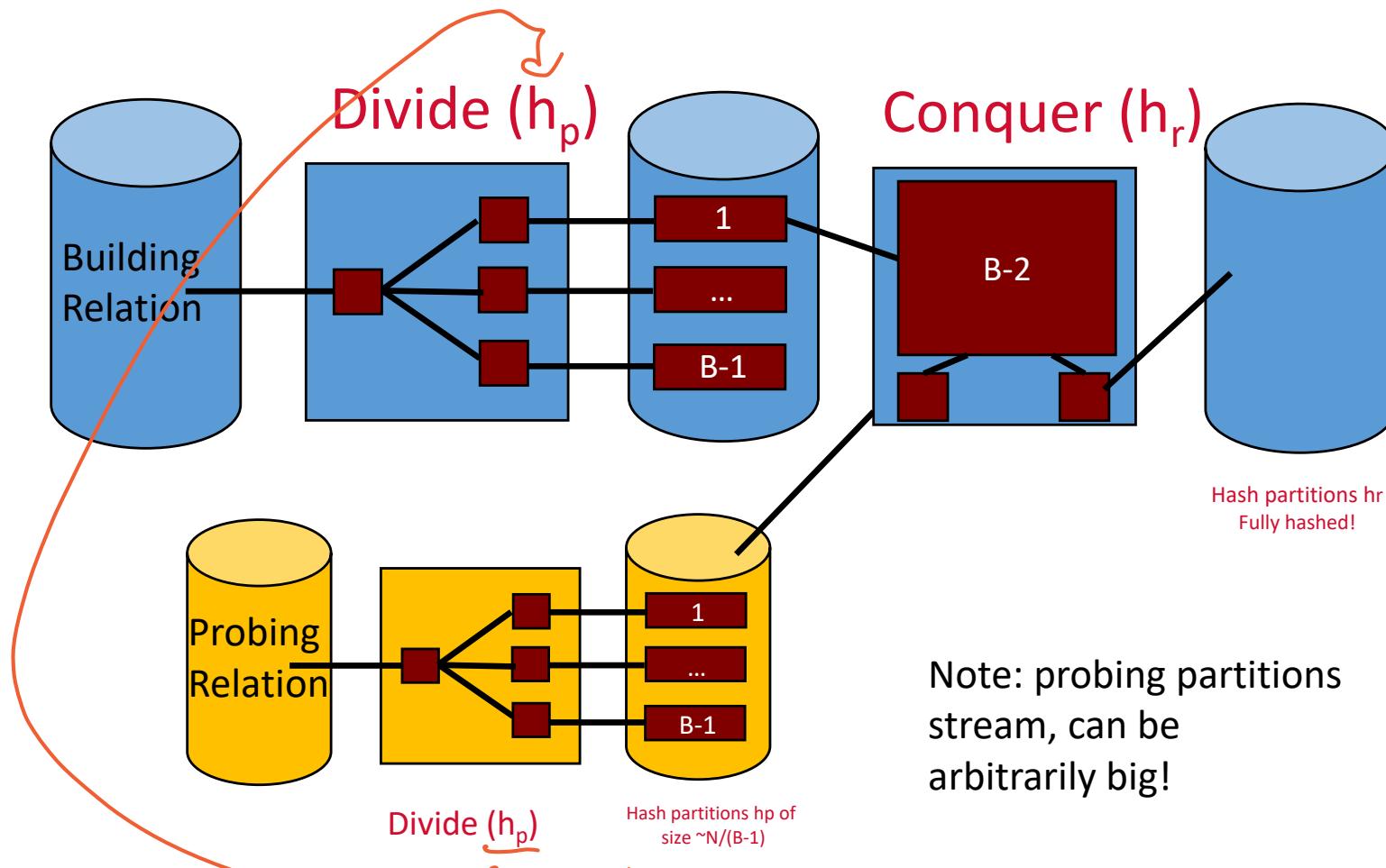
Remember External Hashing?



Sketch of Grace Hash Join



Sketch of Grace Hash Join, cont.



Same hash

PsuedoCode, Grace Hash

For Cur in {R, S}

 For page in Cur

 Read page into input buffer

 For tup~~k~~ on page

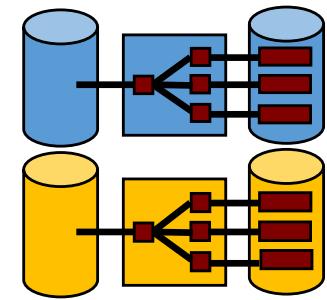
 Place tup in output buf $\text{hash}_p(\text{tup.joinkey})$

 If output buf full then flush to disk partition

Flush output bufs to disk partitions

partition

(hashing for R, S)



PsuedoCode, Grace Hash, cont.

For Cur in {R, S}

 For page in Cur

 Read page into input buffer

 For tup on page

 Place tup in output buf $\text{hash}_p(\text{tup.joinkey})$

 If output buf full then flush to disk partition

 Flush output bufs to disk partitions

For i in $[0..(B-1)]$ *partition range*

 For page in R_i

 For tup on page

 Build tup into memory $\text{hash}_r(\text{tup.joinkey})$

 For page in S_i

 Read page into input buffer

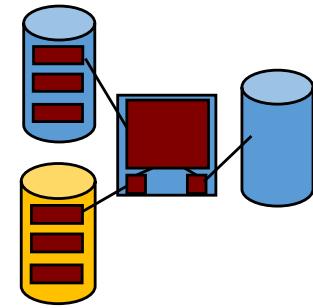
 For tup on page

 Probe memory $\text{hash}_r(\text{tup.joinkey})$ for matches

 Send all matches to output buffer

 Flush output buffer if full

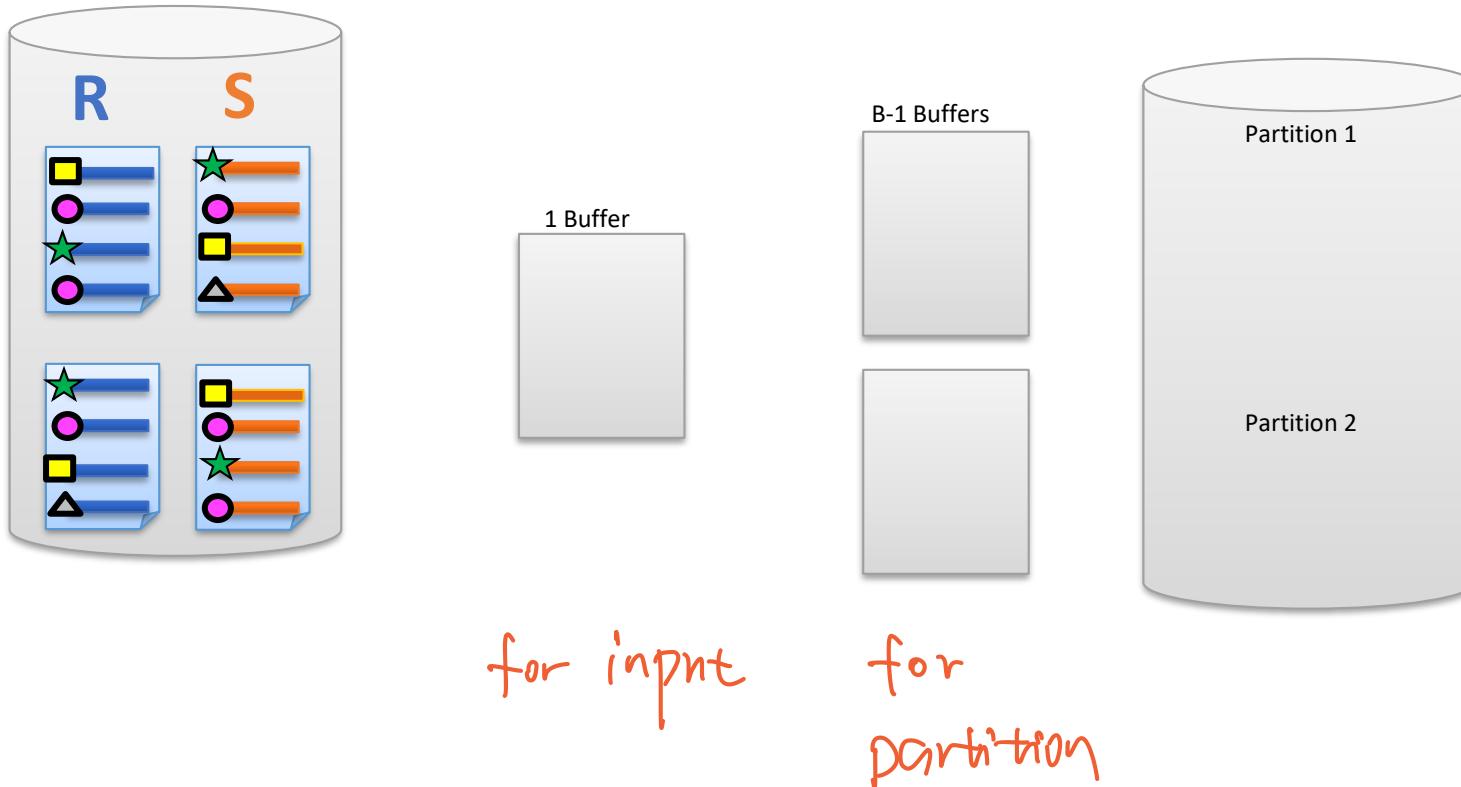
hash table.



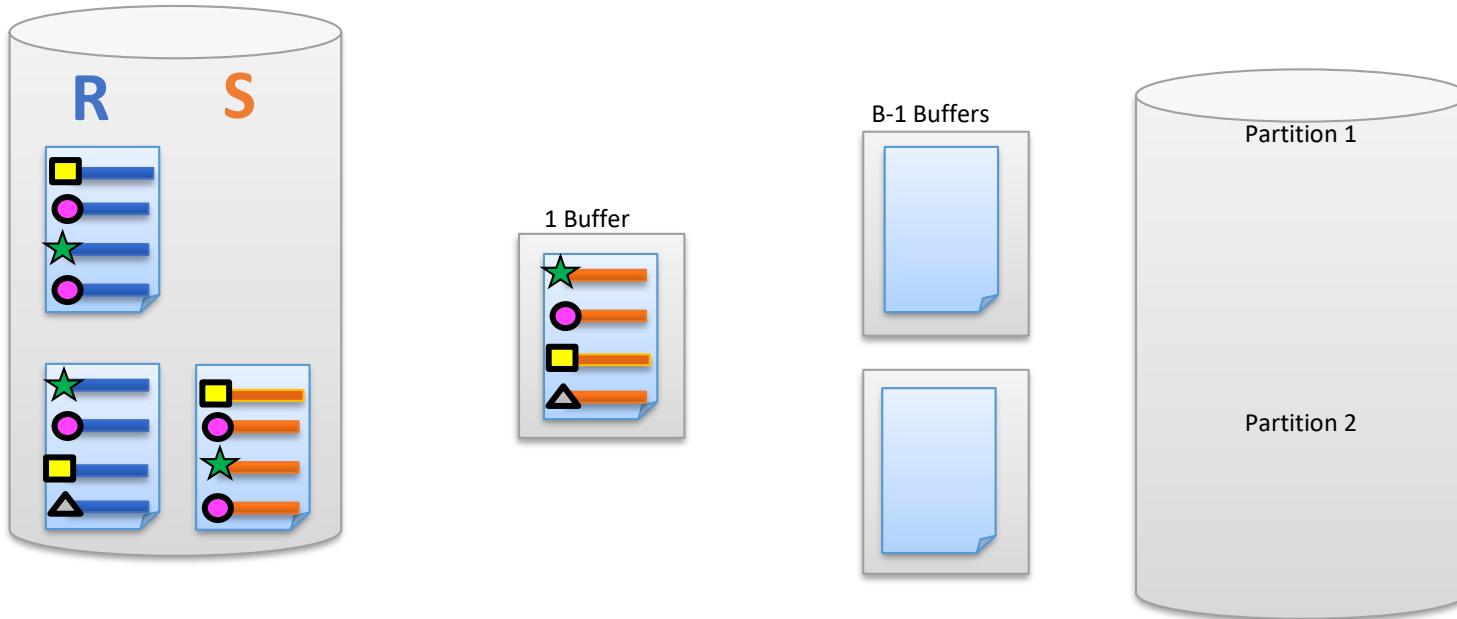
Grace Hash Join

- An animation
- Two phases:
 - Partition (divide)
 - Build & Probe hash tables (conquer)

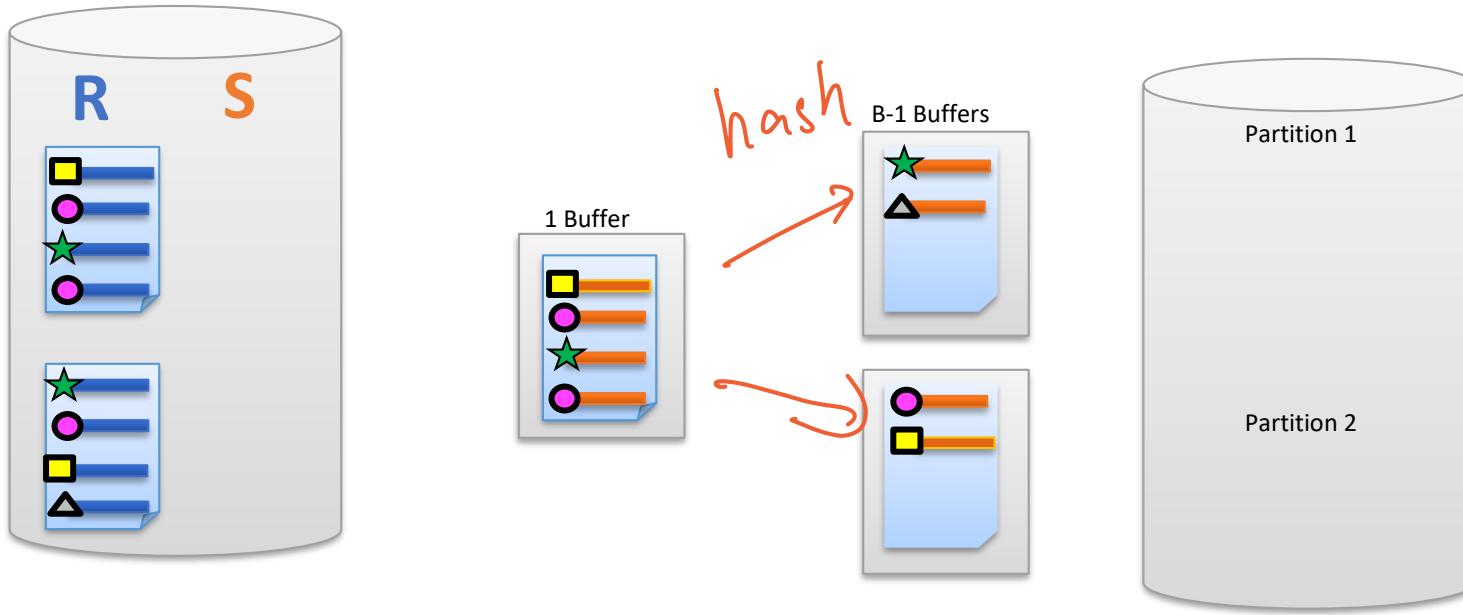
Grace Hash Join: *Partition*



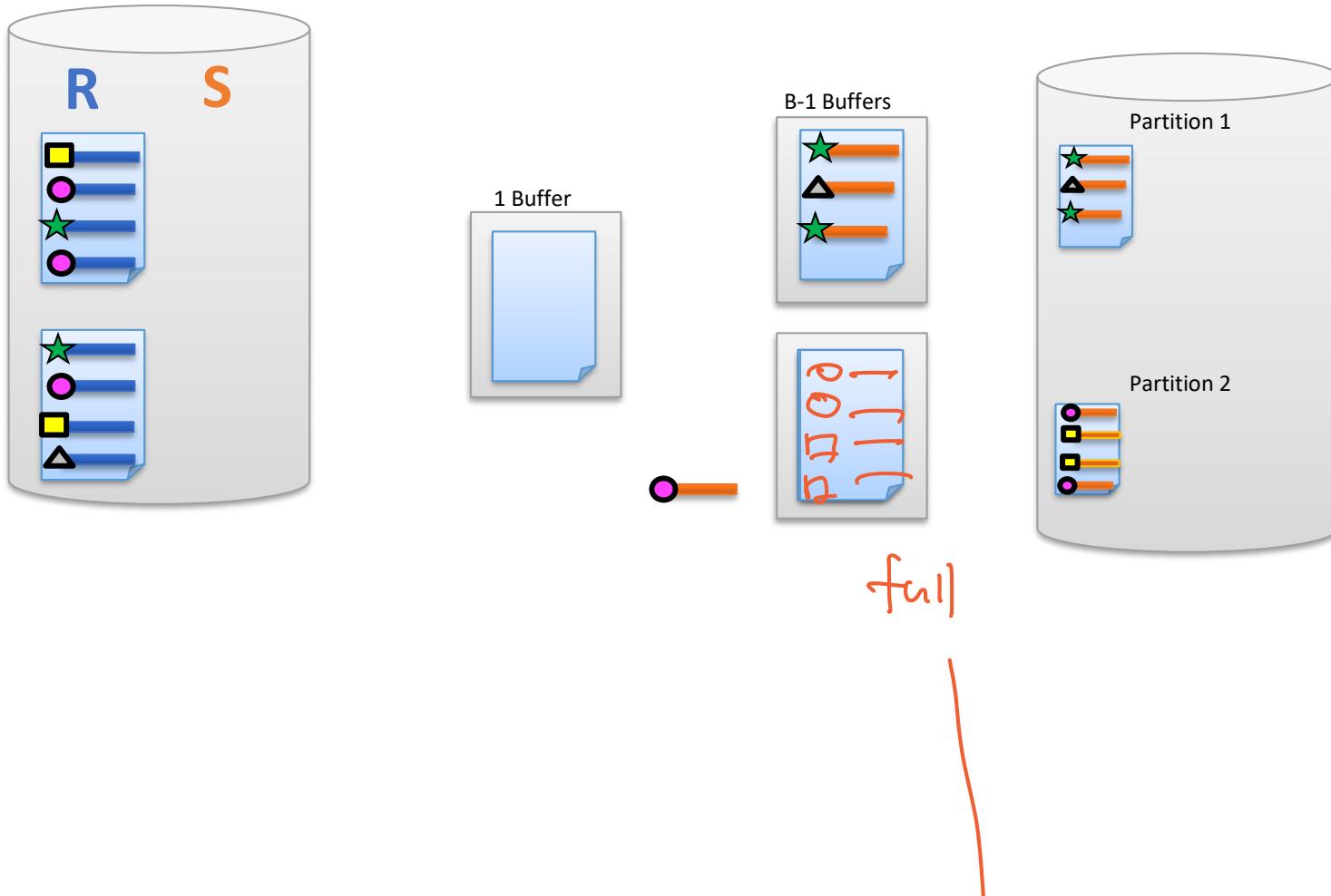
Grace Hash Join: *Partition, Part 2*



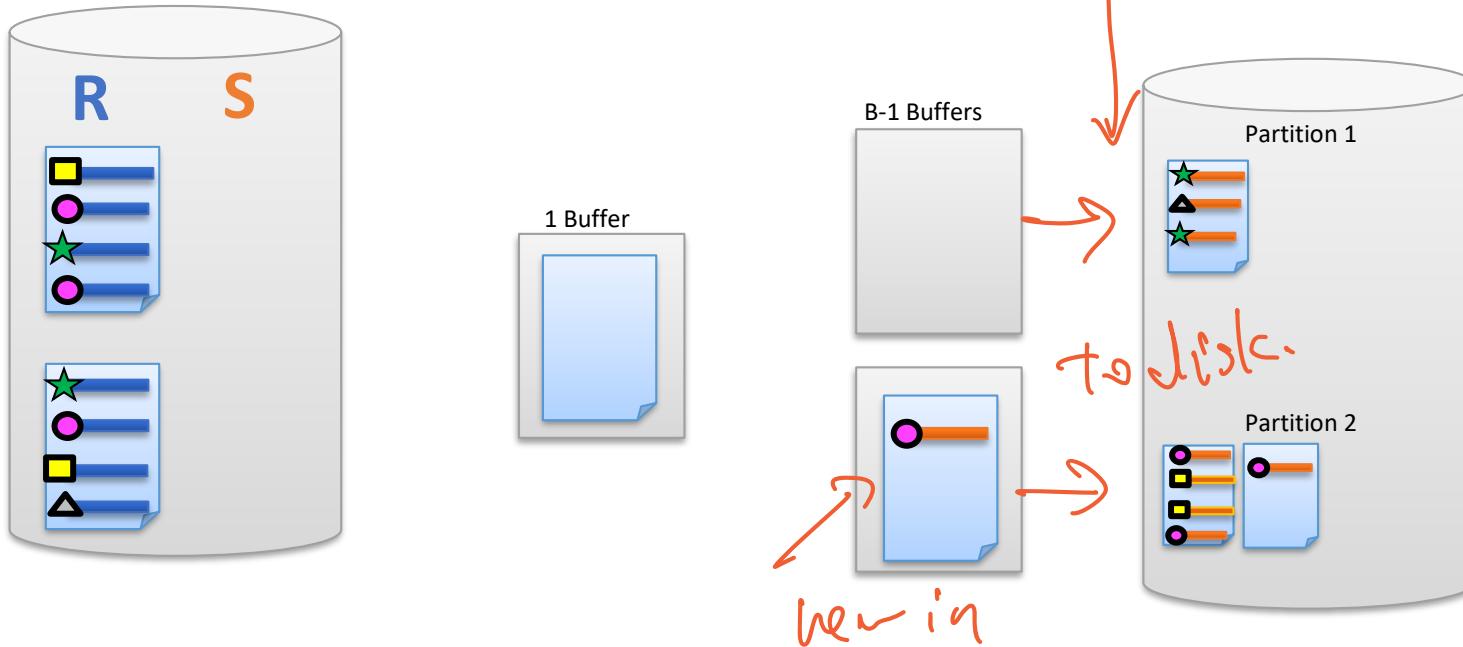
Grace Hash Join: *Partition, Part 3*



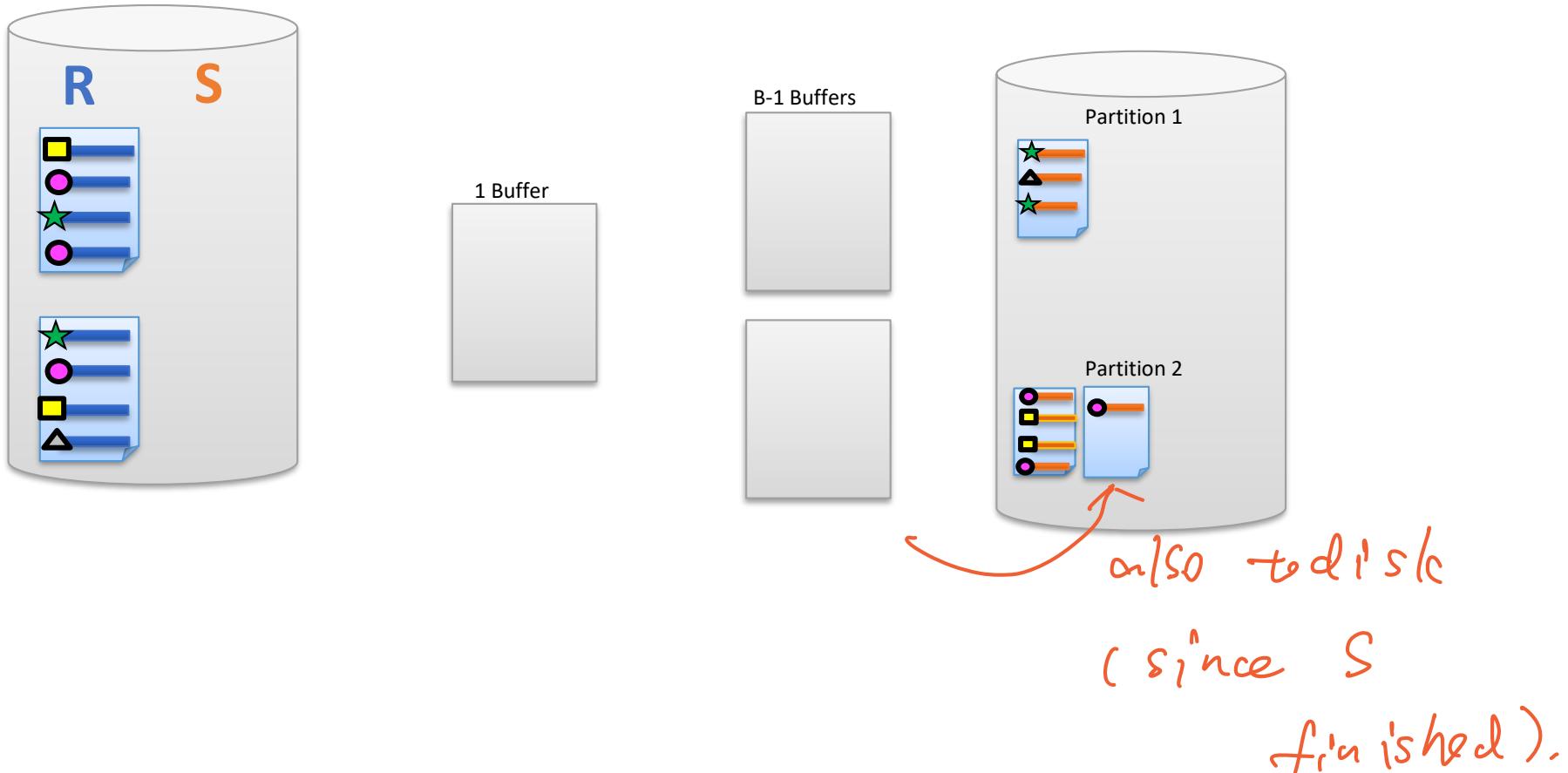
Grace Hash Join: *Partition Part 4*



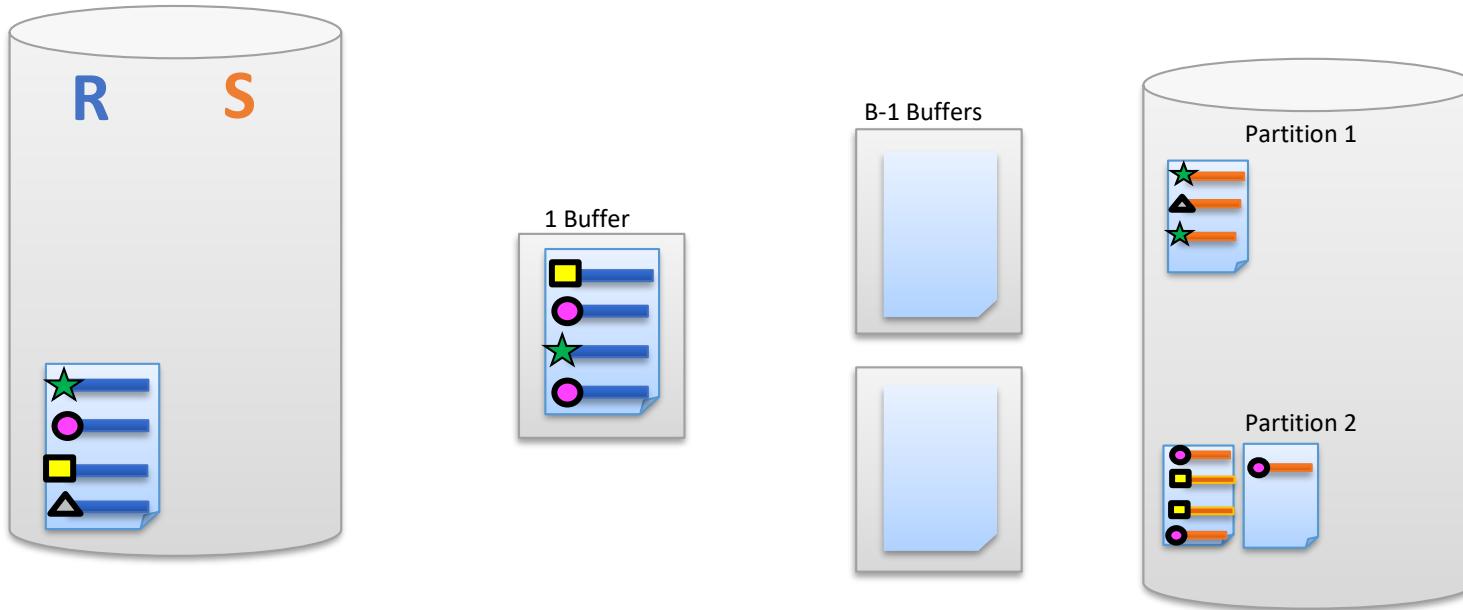
Grace Hash Join: *Partition Part 5*



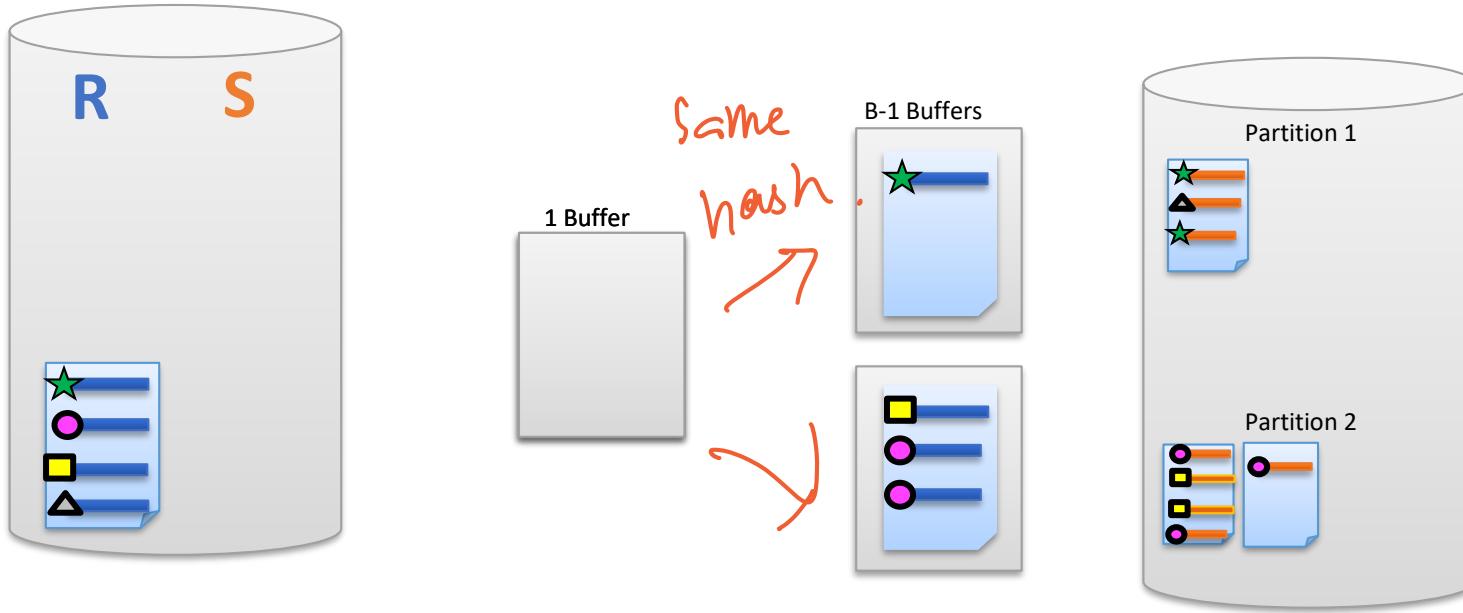
Grace Hash Join: *Partition Part 6*



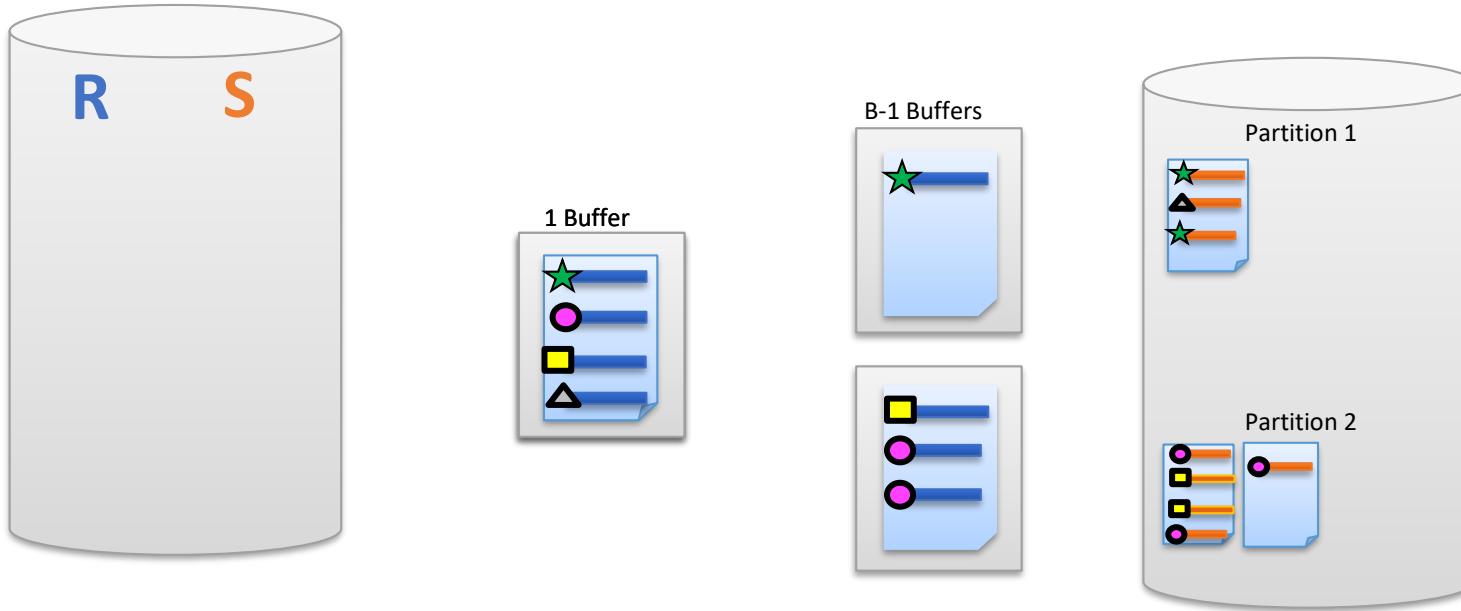
Grace Hash Join: *Partition Part 7*



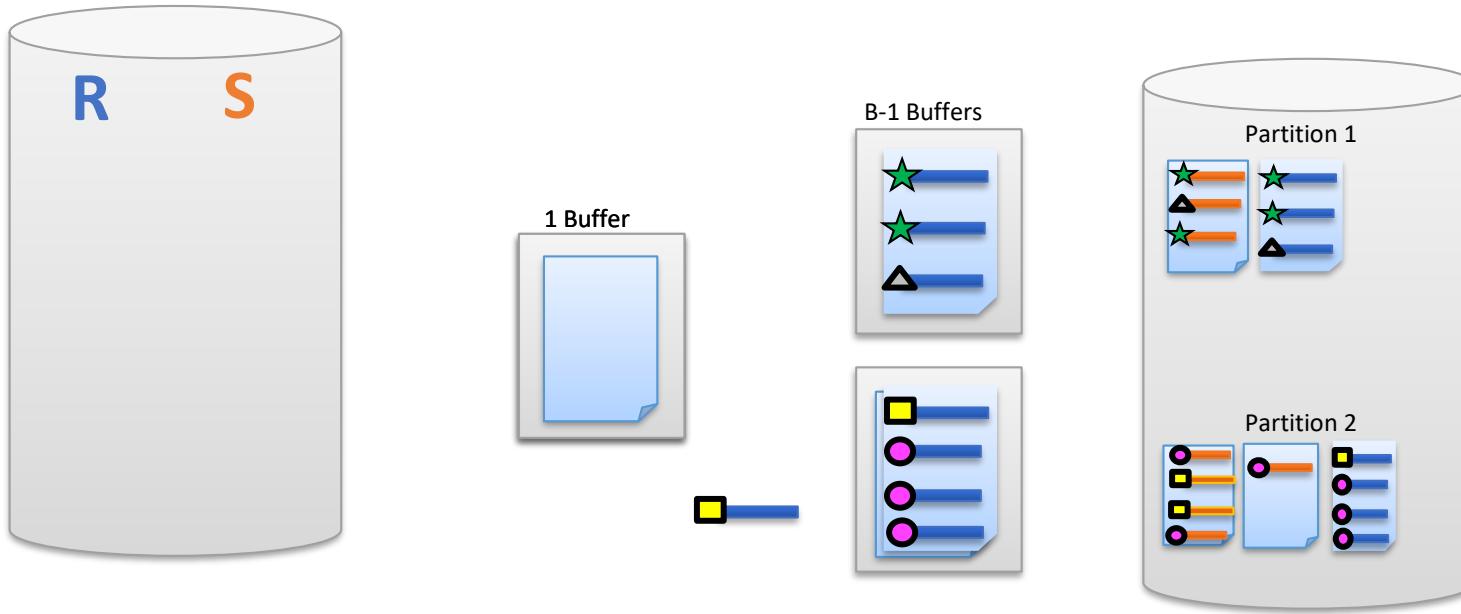
Grace Hash Join: *Partition Part 8*



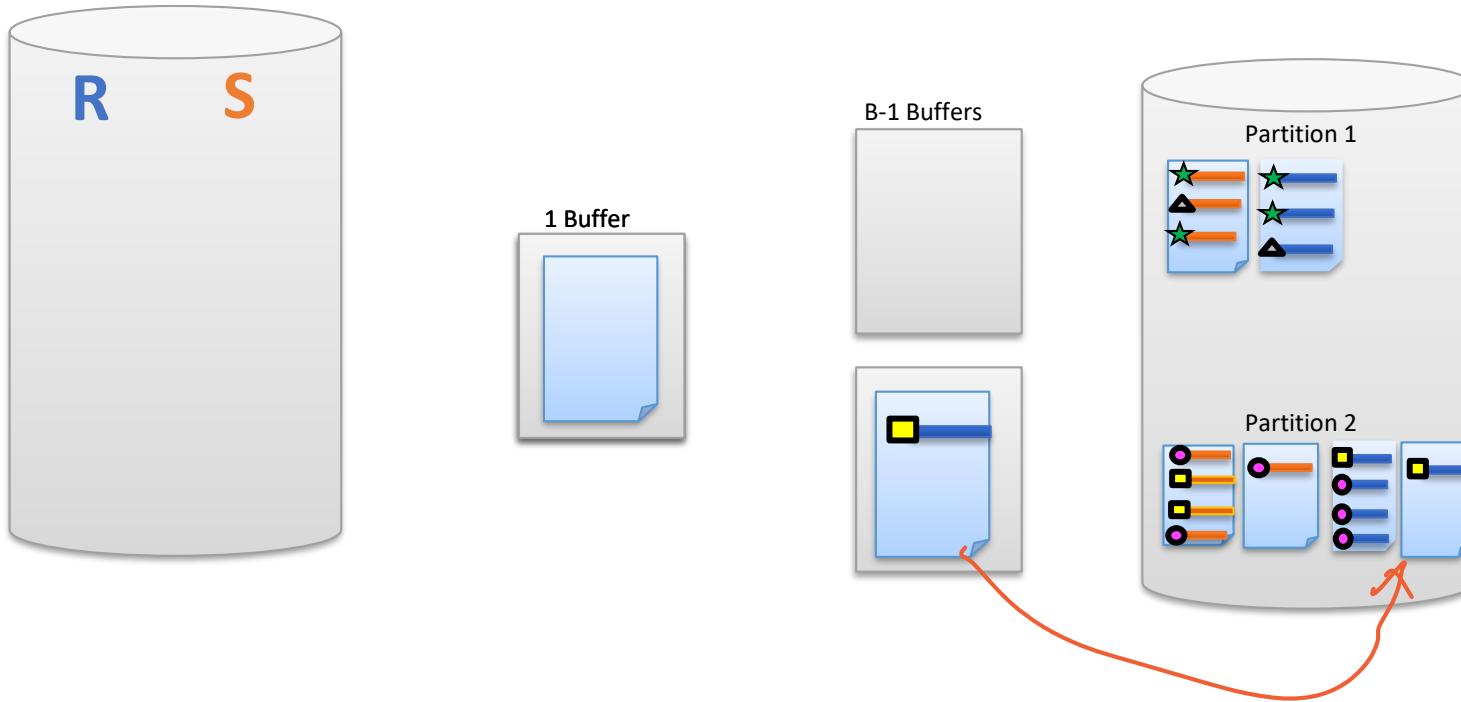
Grace Hash Join: *Partition 9*



Grace Hash Join: *Partition Part 10*



Grace Hash Join: *Partition Part 11*

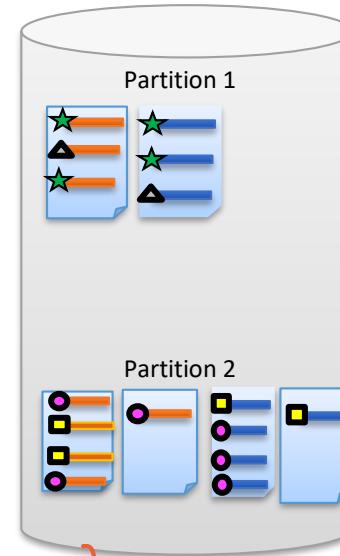


partition finished.

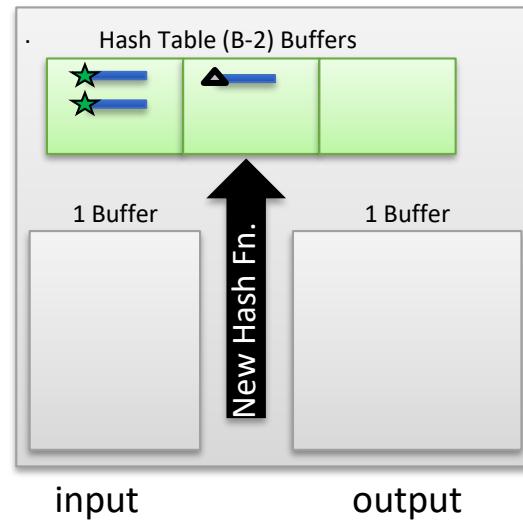
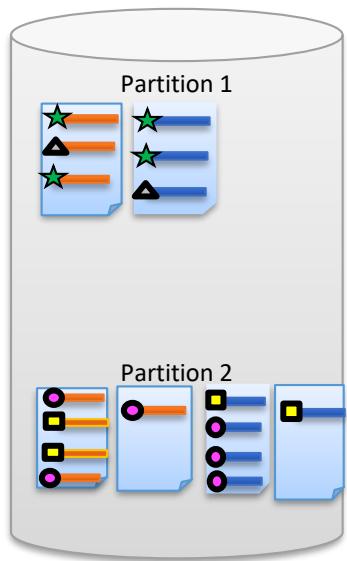
Grace Hash Join: *Partition Part 12*

- Each key is assigned to one partition
 - e.g., green star keys only in Partition 1 →
- Sensitive to key Skew
 - Fuchsia circle Key
- Each partition could be on a different disk or even different machine

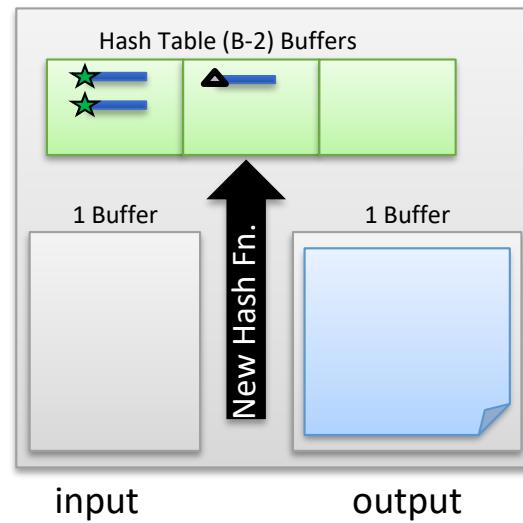
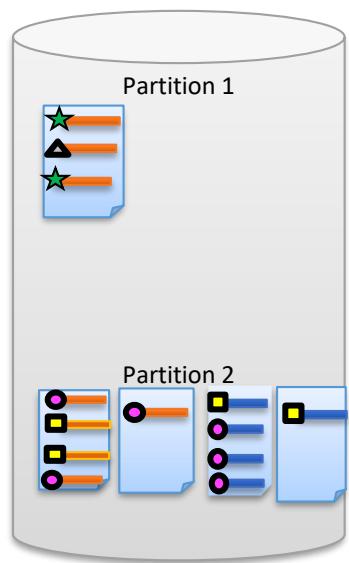
(also can be done on a diff machine)



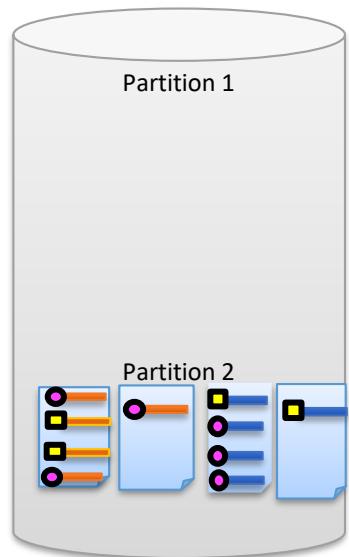
Grace Hash Join: *Build & Probe*



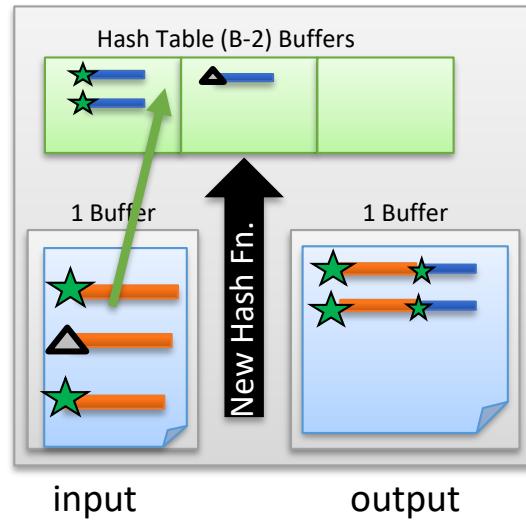
Grace Hash Join: *Build & Probe Part 2*



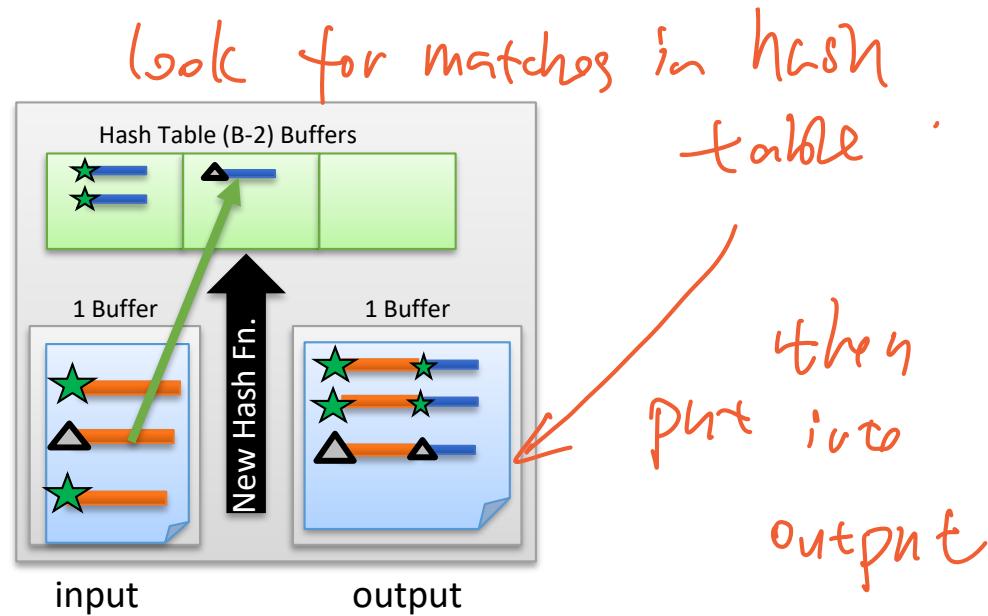
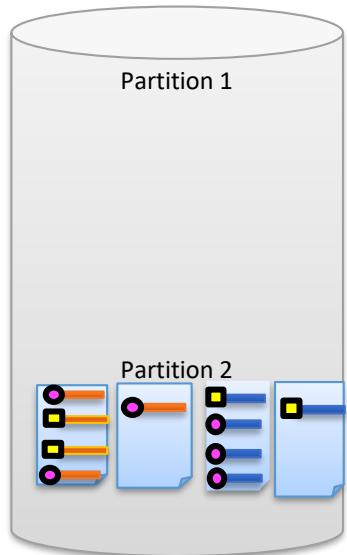
Grace Hash Join: *Build & Probe Part 3*



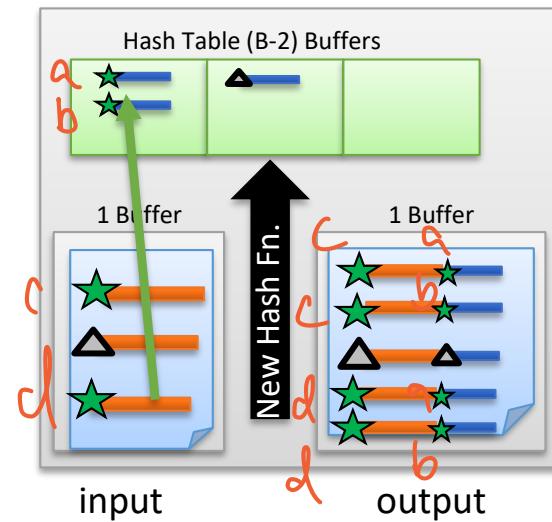
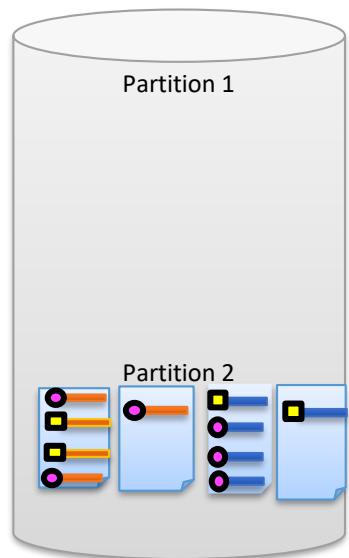
building hash table'



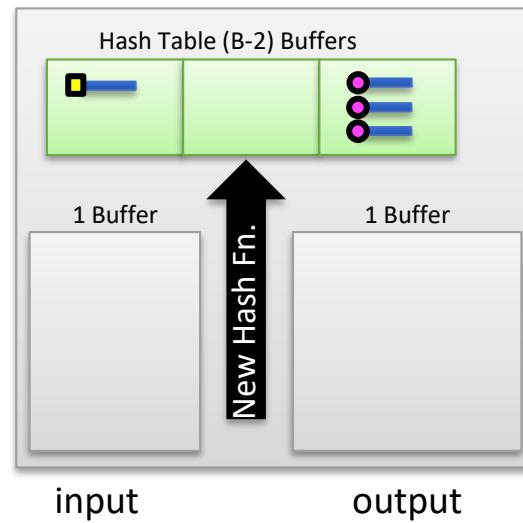
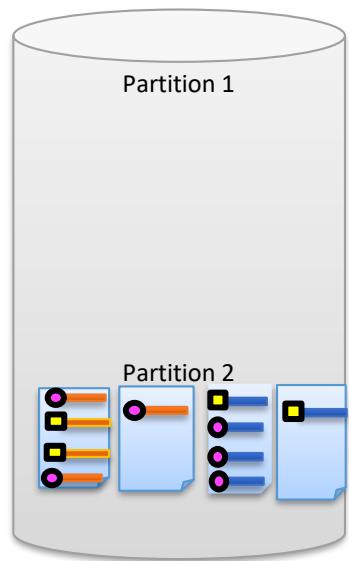
Grace Hash Join: *Build & Probe Part 4*



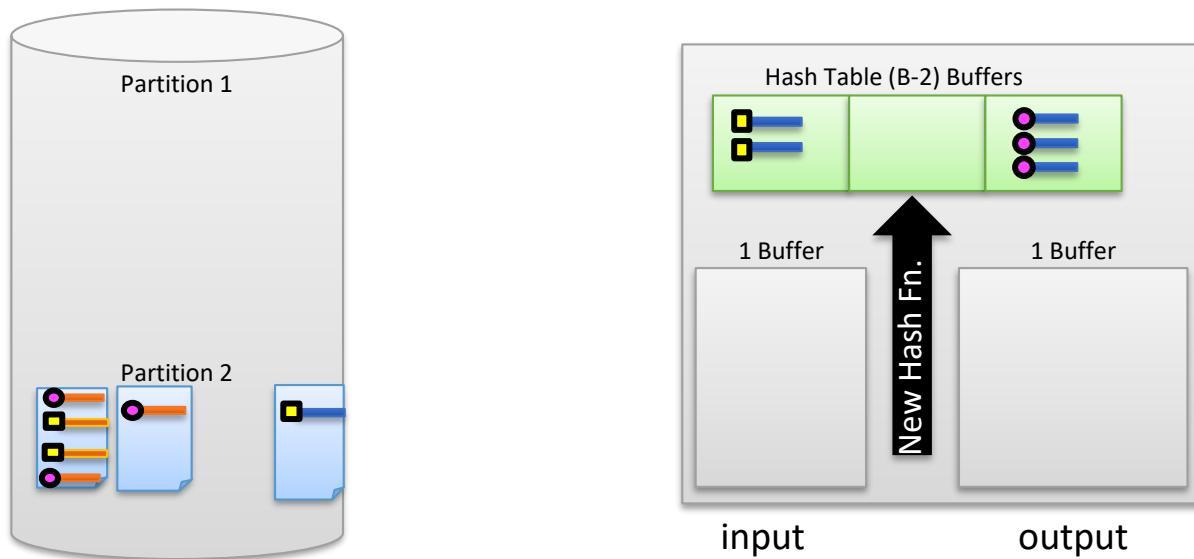
Grace Hash Join: *Build & Probe Part 5*



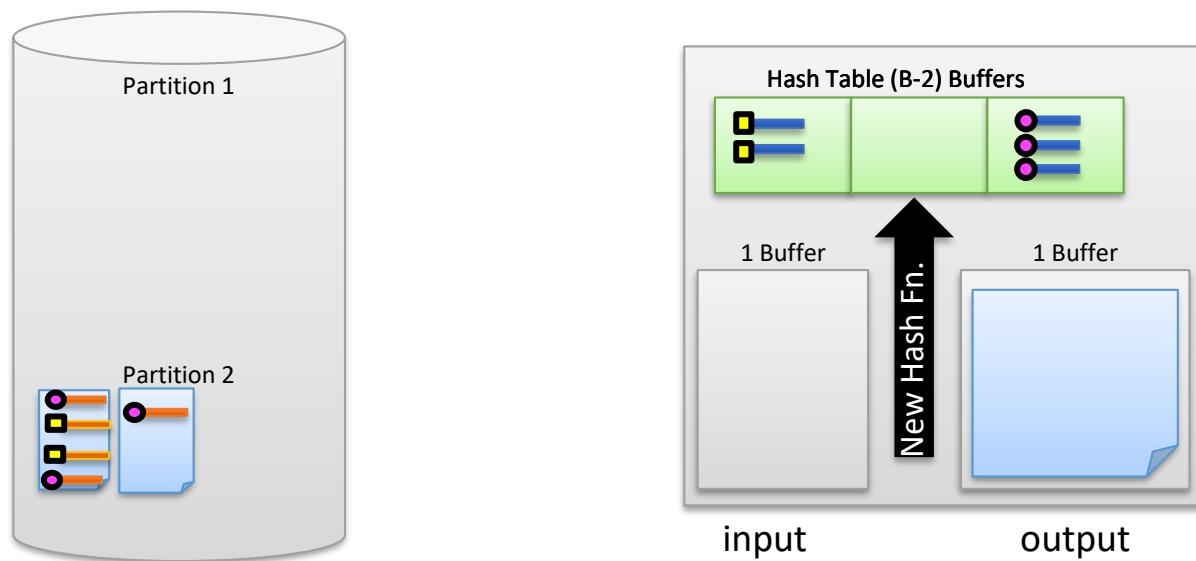
Grace Hash Join: *Build* & *Probe* Part 6



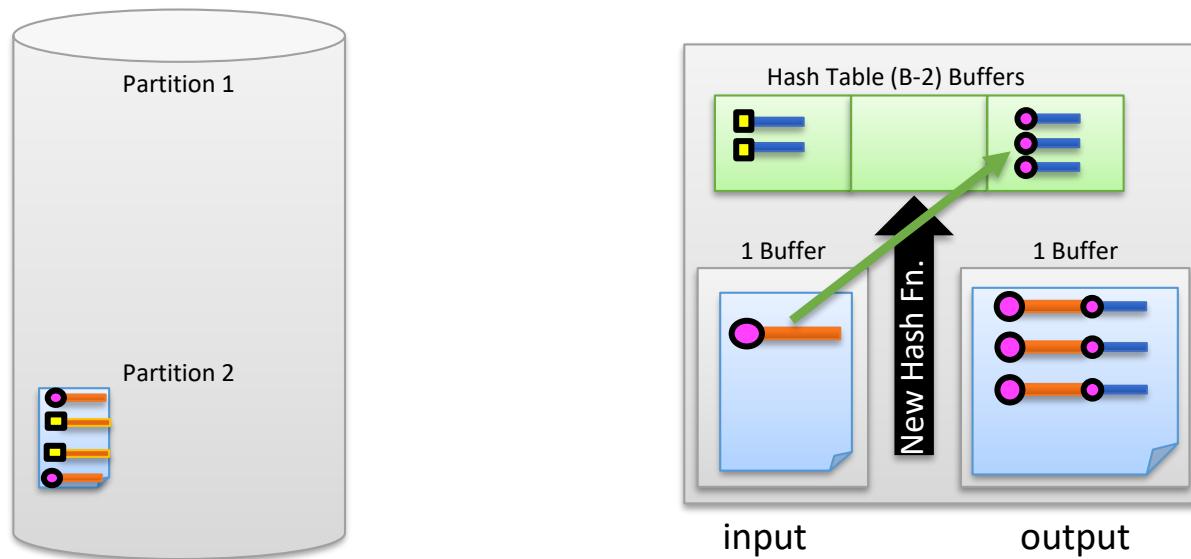
Grace Hash Join: *Build* & *Probe* Part 7



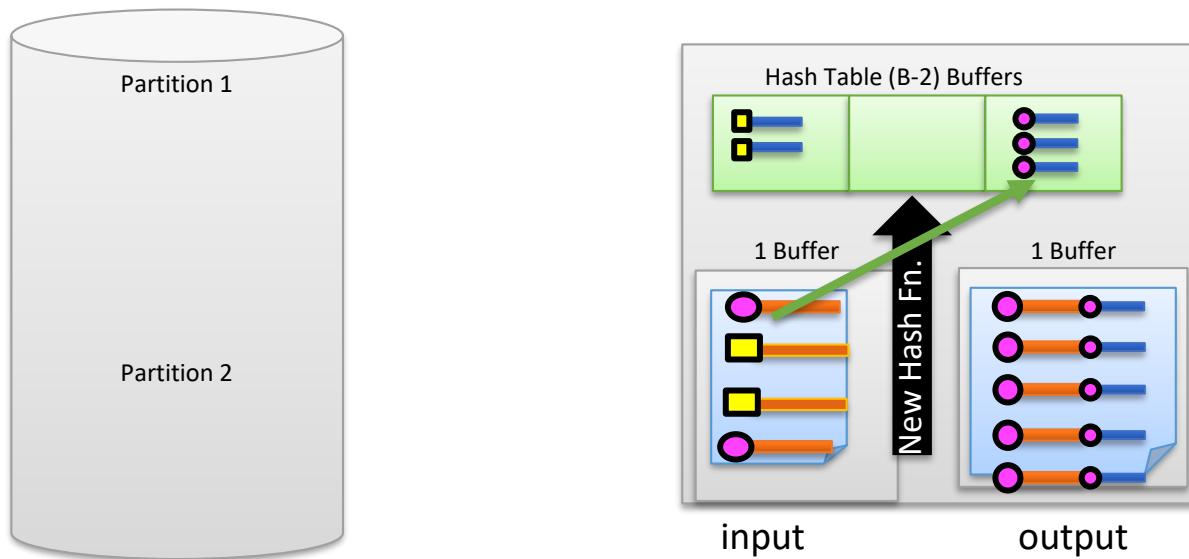
Grace Hash Join: *Build & Probe Part 8*



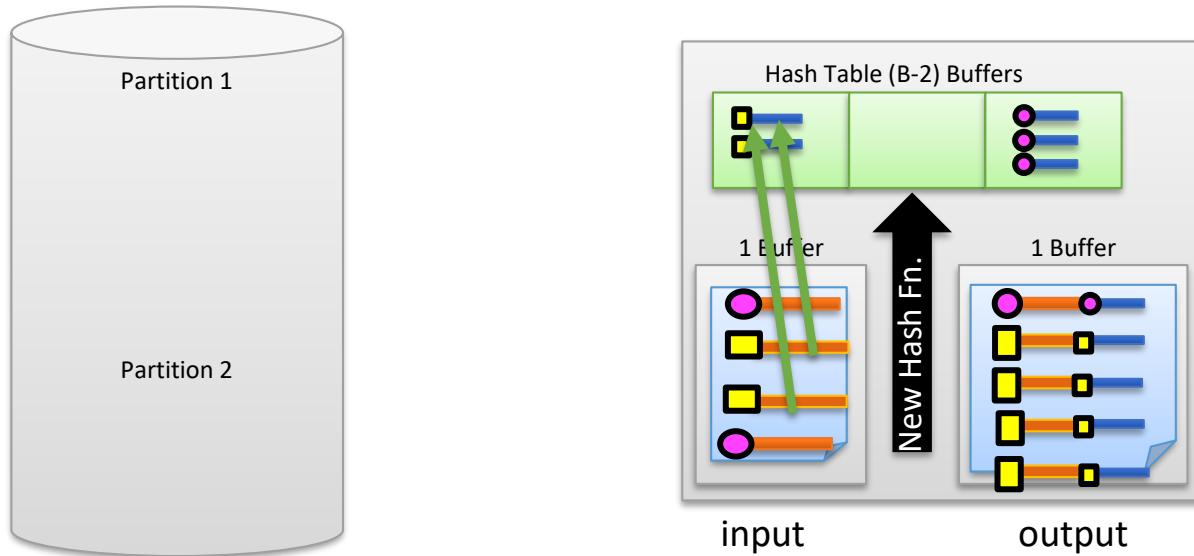
Grace Hash Join: *Build & Probe Part 9*



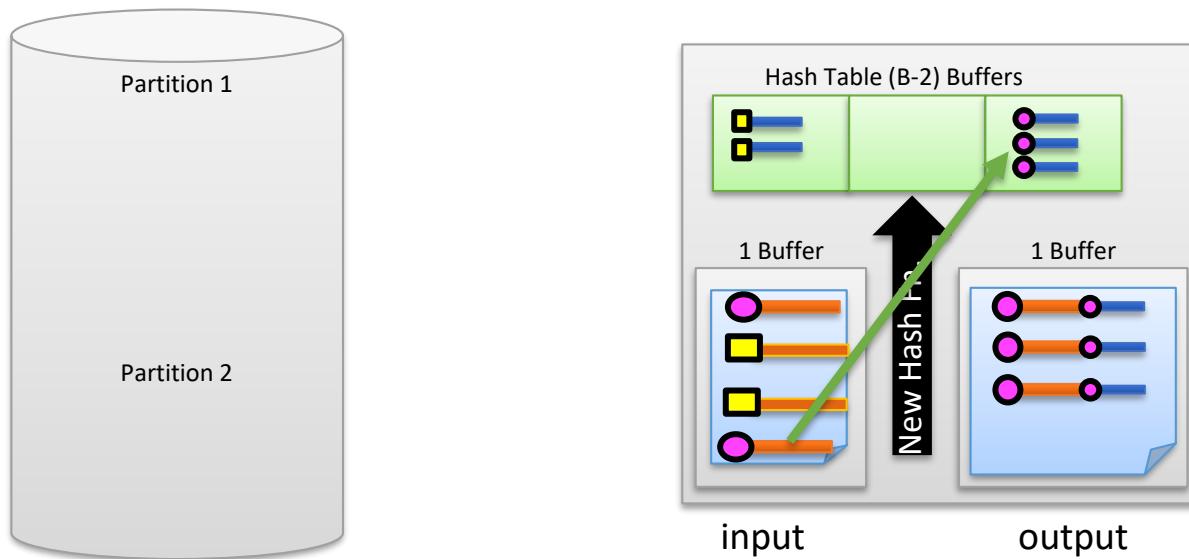
Grace Hash Join: *Build & Probe Part 10*



Grace Hash Join: *Build & Probe Part 11*



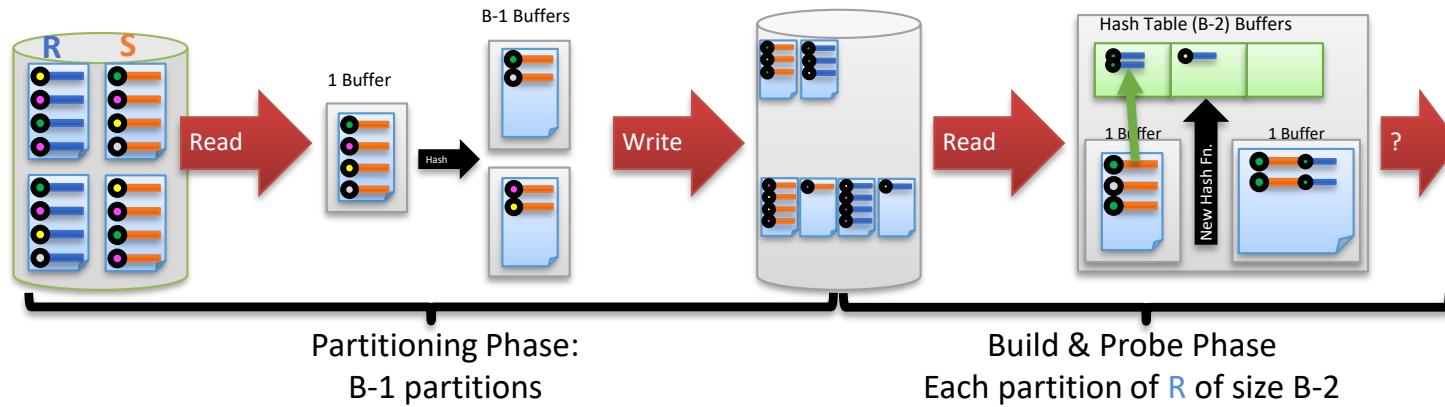
Grace Hash Join: *Build & Probe Part 12*



flushing output whenever

Summary of Grace Hash Join

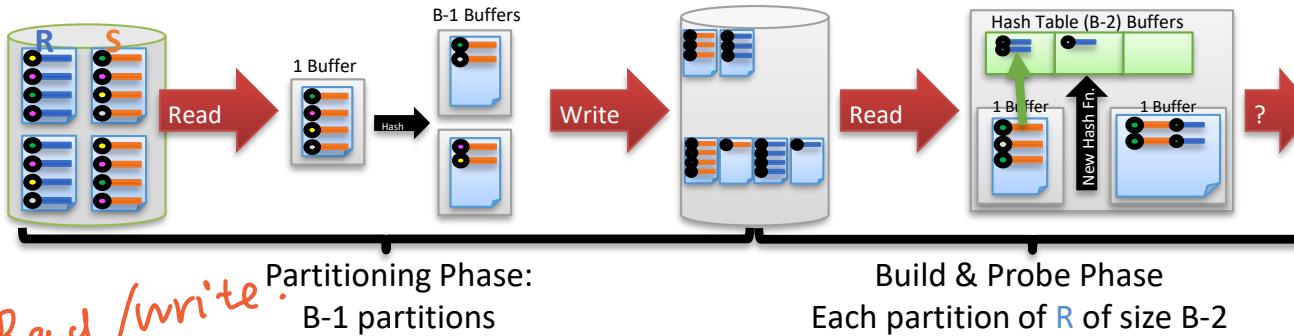
full ,



What is the Cost?

Cost of Hash Join

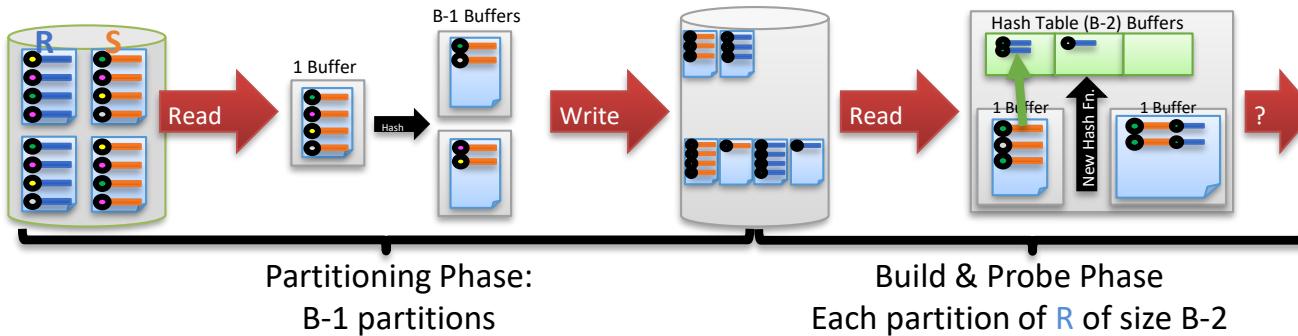
$[R]=1000, p_R=100, |R| = 100,000$
 $[S]=500, p_S=80, |S| = 40,000$



- Partitioning phase: read+write both relations
 $\textcircled{R} \ 2([R]+[S]) \text{ I/Os}$ card.
- Matching phase: read both relations, forward output
 $\textcircled{R} \ [R]+[S]$
- Total cost of 2-pass hash join = $3([R]+[S])$
 - $3 * (1000 + 500) = 4500$

Cost of Hash Join Part 2

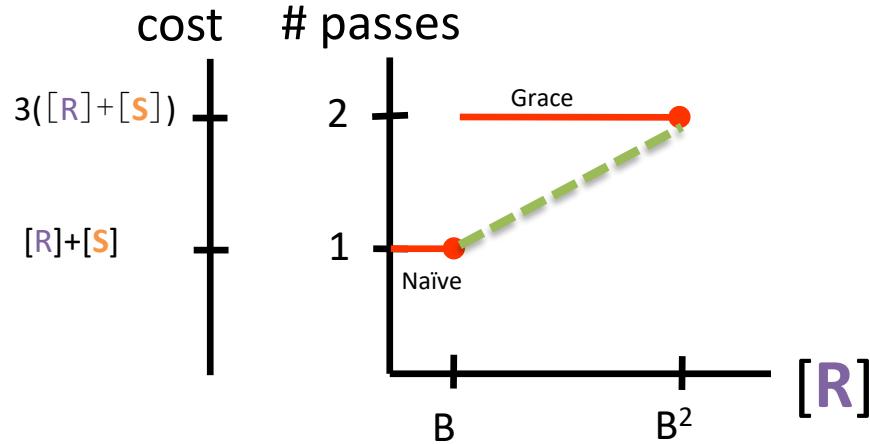
$[R]=1000, p_R=100, |R| = 100,000$
 $[S]=500, p_S=80, |S| = 40,000$



- **Memory Requirements?**
- Build hash table on R with uniform partitioning
 - **Partitioning Phase** divides R into $(B-1)$ runs of size $[R] / (B-1)$
 - **Matching Phase** requires each $[R] / (B-1) < (B-2)$
 - $R < (B-1)(B-2) \approx B^2$
- Note: no constraint on size of S (probing relation)!

Cost of Hash Join Part 3

$[R]=1000, p_R=100, |R| = 100,000$
 $[S]=500, p_S=80, |S| = 40,000$



- **Naïve Hash Join:** *requires $[R] < B$*
 - Put all of R in hash table
 - 1/3 the I/O cost of Grace!
- **Grace Hash Join:** 2-passes for $[R] < B^2$
- **Hybrid Hash Join:** an algorithm that adapts between the two
 - Tricky to tune

Hash Join vs. Sort-Merge Join

- Sorting pros:
 - Good if input already sorted, or need output sorted
 - Not sensitive to data skew or bad hash functions
- Hashing pros:
 - For join: # passes depends on size of smaller relation
 - E.g. if smaller relation is <B, naïve/hybrid hashing is great
 - Good if input already hashed, or need output hashed

Recap

- Nested Loops Join
 - Works for arbitrary Θ
 - Make sure to utilize memory in blocks
- Index Nested Loops
 - For equi-joins
 - When you already have an index on one side
- Sort/Hash
 - For equi-joins
 - No index required
 - Hash better if one relation is much smaller than other
- No clear winners – may want to implement them all
- Be sure you know the cost model for each
 - You will need it for query optimization!