CS 4604: Introduction to Database Management Systems

Query Processing

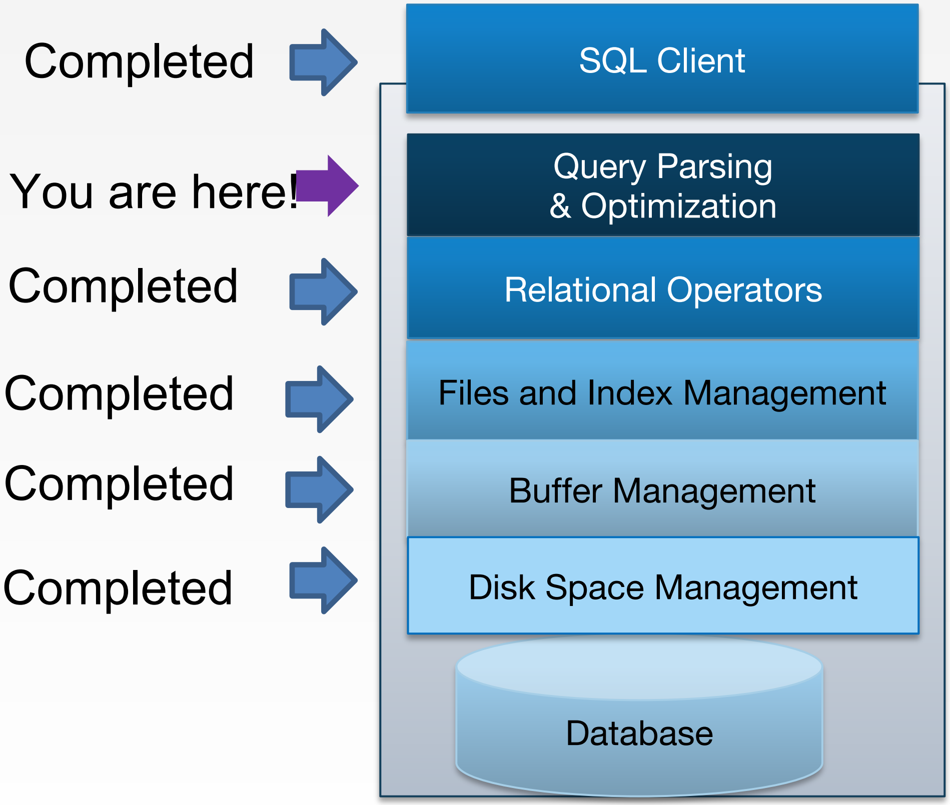
Virginia Tech CS 4604 Sprint 2021

Instructor: Yinlin Chen

Today's Topics

- Implementation of the relational operators
  - Selection
  - Projection
  - Join
  - Set & aggregate operations

Today


Completed
You are here!
Completed
Completed
Completed
Completed
SQL Client
Query Parsing
& Optimization
Relational Operators
Files and Index Management
Buffer Management
Disk Space Management
Database

An Overview of the Layer Above

SQL Query

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

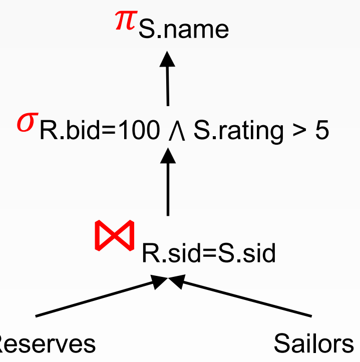
Query Parser & Optimizer

Relational Algebra

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

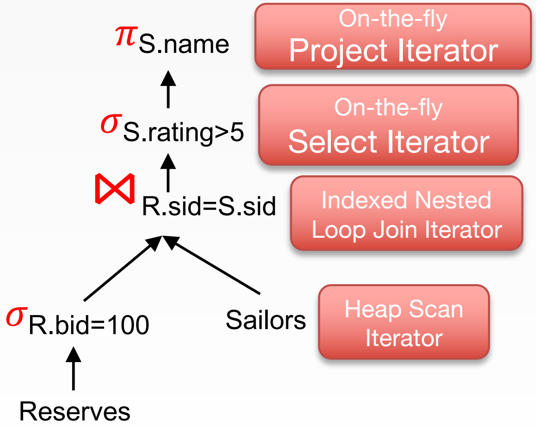
Equivalent to…

(Logical) Query Plan:



But actually will produce…

Optimized (Physical) Query Plan:



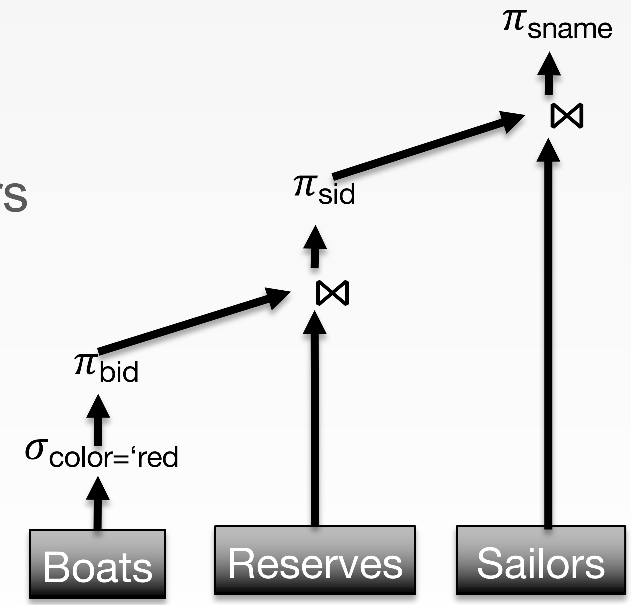
Operator Code

B+-Tree Indexed Scan Iterator

Relational Operators and Query Plans

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

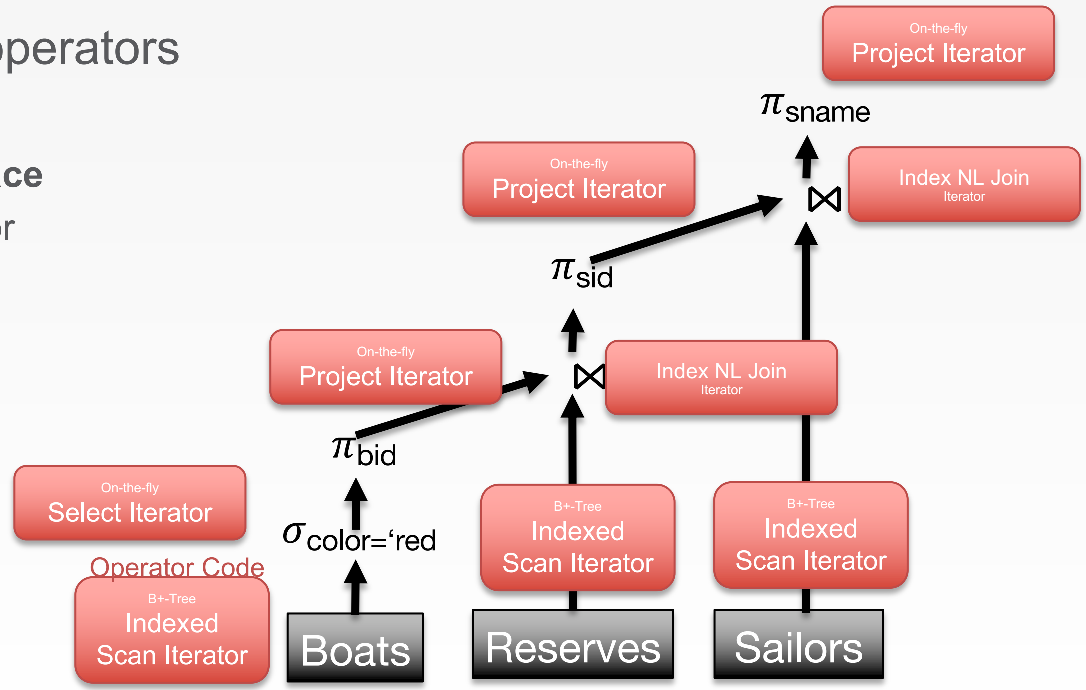
- Query plan
  - Edges encode "flow" of tuples
  - Vertices = Relational Algebra Operators
  - Source vertices = table access operators …
- Also called dataflow graph



Query Executor Instantiates Operators

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

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



Relational Operations

- Some database operations (Joins) are EXPENSIVE
- Performance can be improved by:
  - clever implementation techniques for operators
  - exploiting “equivalencies” of relational operators
  - using statistics and cost models to choose among these.

Relational Operations Implement

- *Selection* (σ): Selects a subset of rows from relation
- *Projection* (π): Deletes unwanted columns from relation
- *Join* ($\bowtie$): Allows us to combine two relations
- *Set-difference* (-): Tuples in relation 1, but not in relation 2
- *Union* ($\cup$): Tuples in relation 1 and in relation 2
- *Aggregation* (SUM, MIN, etc.) and GROUP BY
- Operations can be composed

Schema for Examples

Sailors (<u>*sid*: integer</u>, *sname*: string, *rating*: integer, *age*: real)
Reserves (<u>*sid*: integer, *bid*: integer, *day*: dates</u>, *rname*: string)

Sailors:

- Each tuple is 50 bytes long, 80 tuples per page, 500 pages.
- N=500, p_S=80.

Reserves:

- Each tuple is 40 bytes long, 100 tuples per page, 1000 pages.
- M=1000, p_R=100.

Simple Selections

Of the form $\sigma_{R.attr\ op\ value}(R)$

Question: how best to perform?

```
SELECT  *
FROM    Reserves R
WHERE   R.rname = 'Joe'
```

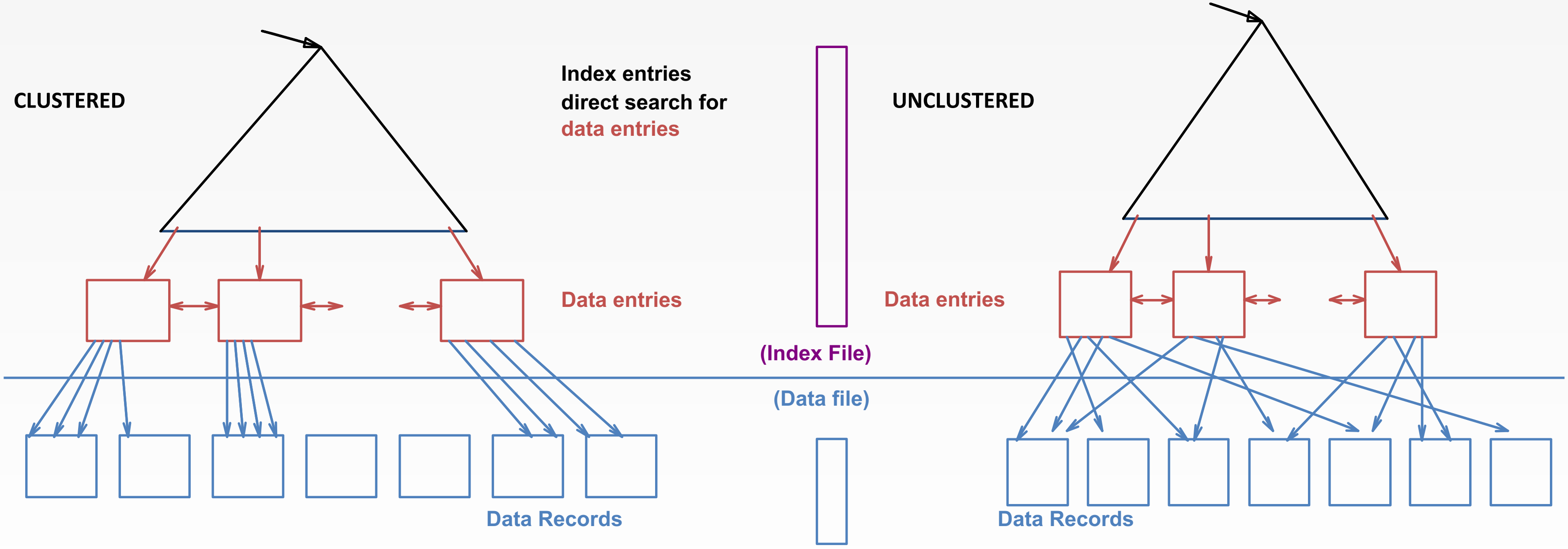
Simple Selections

- No index, unsorted data
  - Must essentially scan the whole relation
  - Cost is O(M)
    - For “reserves” = 1000 I/Os.
- No index, sorted data
  - cost of binary search O ($\log_2 M$) + number of pages containing results
  - For reserves = 10 I/Os + $\lceil$selectivity * #pages$\rceil$

Using an Index for Selections

- Use index to find qualifying data entries (tuples), then retrieve corresponding data records
- Hash index useful only for equality selections
- Cost depends on #qualifying tuples, and clustering
  - Finding qualifying data entries (typically small)
  - Plus, cost of retrieving records (could be large **w/o** clustering)

Selections using Index (cnt'd)



Selections using Index

- Example “reserves” relation: 100 tuples per page, 1000 pages
- If 10% of tuple qualify (100 pages, 10,000 tuples)
  – With a clustered index, cost is little more than 100 I/Os;
  – if unclustered, could be up to 10,000 I/Os!

Selections using Index (cnt'd)

- Important refinement for unclustered indexes:
  - 1. Find qualifying data entries.
  - 2. Sort the rid's of the data records to be retrieved.
  - 3. Fetch rids in order. This ensures that each data page is looked at just once (though # of such pages likely to be higher than with clustering).

The Projection Operation

```
SELECT DISTINCT R.sid, R.bid
FROM   Reserves R
```

- Issue is removing duplicates.
- Basic algorithm: sorting
  1. Scan R, extract only the needed attributes
  2. Sort the resulting set
  3. Scan the sorted result and remove adjacent duplicates
- Cost of Reserves:
  1. Scan cost: 1000 I/Os, assume with size ratio 0.25 = 250 pages writing to a temporary relation: 250 I/Os
  2. Assume we have 20 buffer pages can sort in 2 passes: 2 * 2 * 250 = 1000 I/Os
  3. 250 I/Os

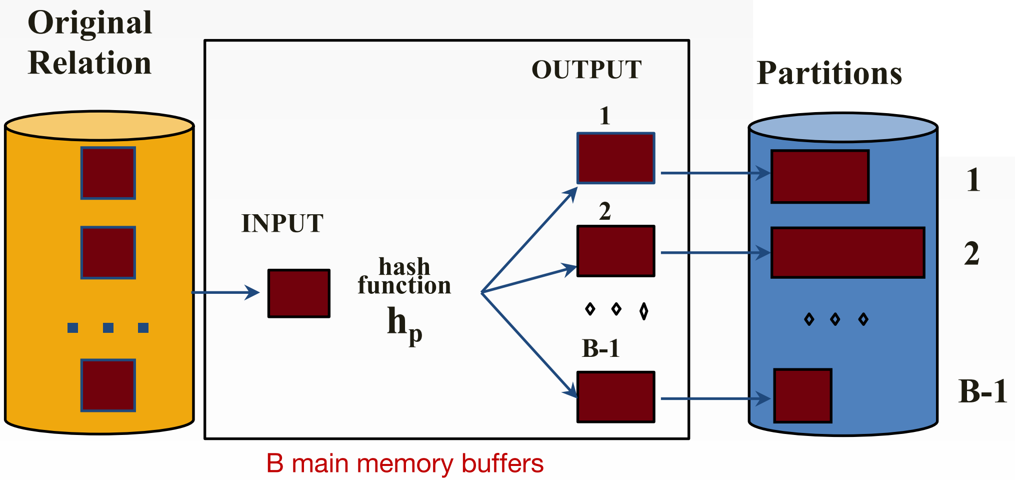
  Total cost: 1000 +250 + 2 * 2 * 250 + 250 = 2500 I/Os

The Projection Operation

- Issue is removing duplicates.
- Basic algorithm: hashing
  1. Partitioning phase (Divide)
  2. Duplication elimination phase

```
SELECT DISTINCT R.sid, R.bid
FROM  Reserves R
```

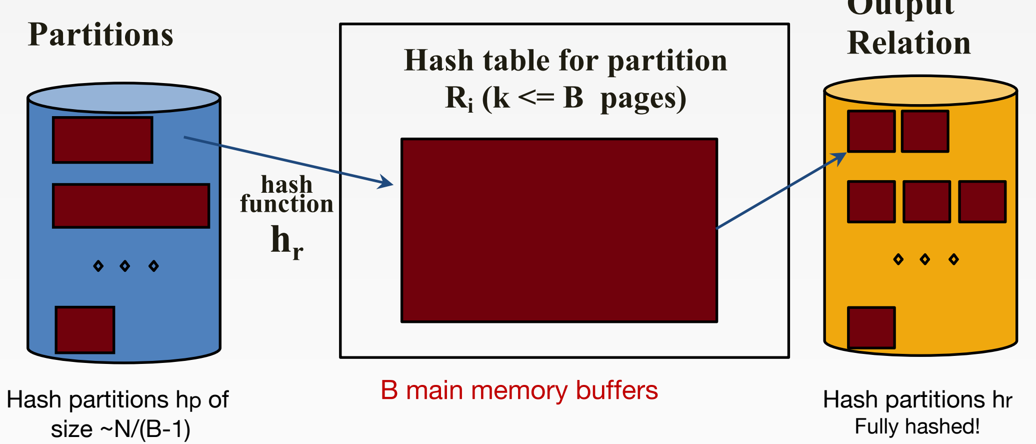
Partition:
(Divide)



The Projection Operation

Rehash:
(Conquer)

Hash partitions h_p of size ~N/(B-1)



Cost of Reserves:

1. Divide: 1000 I/Os + 250 I/Os
2. Conquer: 250 I/Os

Total cost: 1500 I/Os

The Join Operation

- Joins are very common
- Joins can be very expensive (cross product in worst case)
- Many approaches to reduce join cost
- Join techniques
  - Nested-loops join
  - Index-nested loops join
  - Sort-merge join
  - Hash join

The Join Operation

```
SELECT  *
FROM    Reserves R1, Sailors S1
WHERE   R1.sid=S1.sid
```

- Cost Notation
  - [R] : the number of pages to store R
  - p_R : number of records (tuples) 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)
  - $[S]=500$, $p_S=80$, $|S| = 40{,}000$

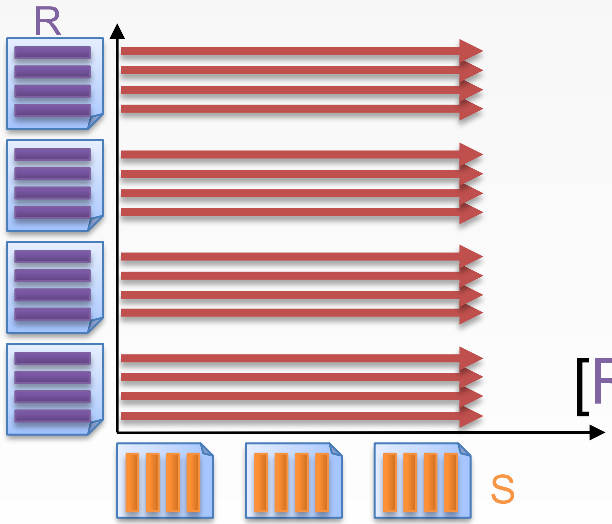
Simple Nested Loops Join

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:

$[R] + |R|[S]$

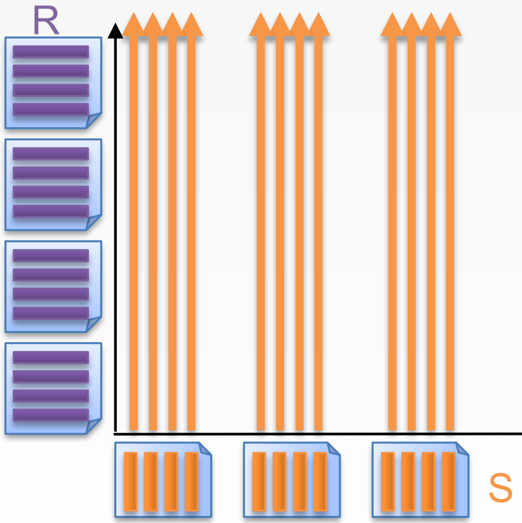
$= 50{,}001{,}000$

Changing the Join Order

foreach **record** s in S do

foreach **record** r in R do

if **θ(ri, sj)** then add <ri, sj> to result buffer



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

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

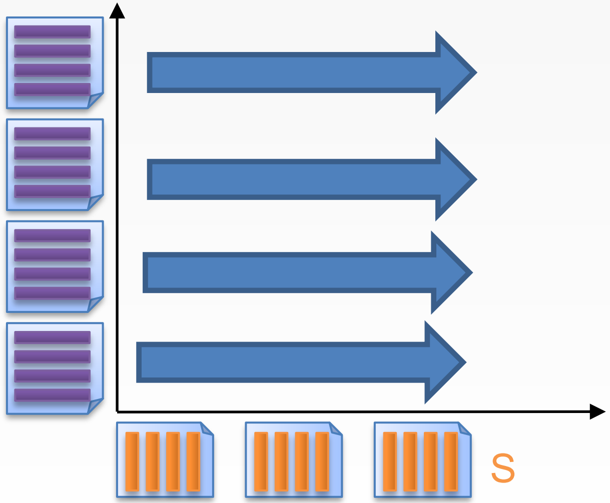
Cost:

$[S] + |S|[R]$

$= \quad 40{,}000{,}500$

vs. 50,001,000

Page Nested Loop Join




```
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
```

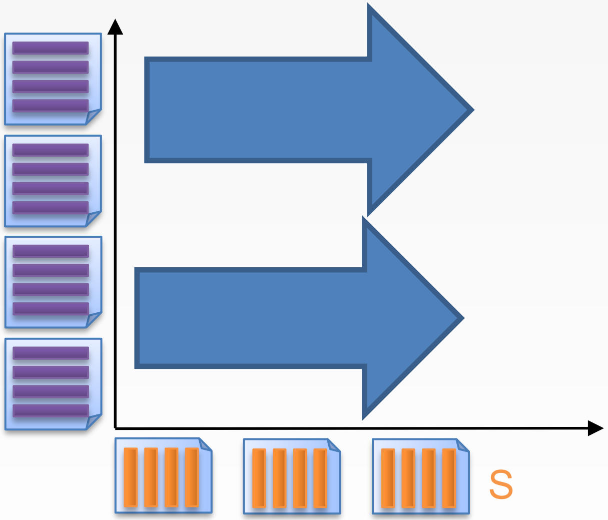
Cost = [R] + ([R] * [S])

= 1000 + (1000 * 500)

= 501,000

Block Nested Loops Join

for each rchunk of B-2 pages of R:
for each spage of S:
for all matching tuples in spage and rchunk:
add <rtuple, stuple> to result buffer



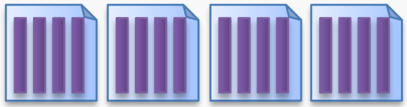
Cost = [R] + ⌈ [R]/(B-2) ⌉ * [S]
= 1000 + ⌈ 1000/(B-2) ⌉ * 500
= 6,000 for B=102 (~100x better than Page NL!)

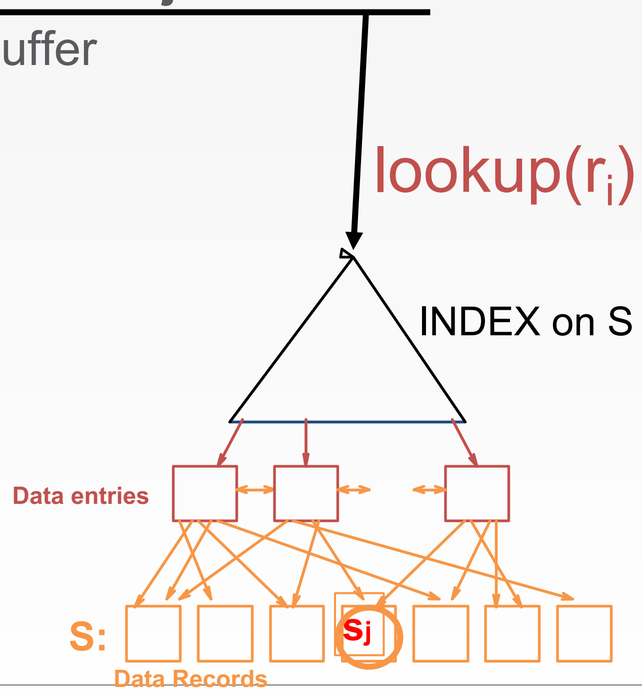
Use Buffer Pages

Index Nested Loops Join

```
foreach tuple r in R do
    foreach tuple s in S where ri == sj  do
        add <ri, sj> to result buffer
```

R





sid is the primary key for Sailors, so there is exactly one matching sailor for each tuple in R

Index Nested Loops Join Cost

Cost = [R] + |R| * cost to find matching S tuples

- If index uses Alt. 1 → 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**
- Clustered Cost(R,S): [R] + |R| * (Search + # of matching **pages**)
  - B+-tree height 2 (3 I/Os from root to leaf)
  - R⋈S: 1000 + (100,000)*(3 +1) = 401,000
- Unclustered Cost(R,S) = [R] + |R| * (Search + # matching **tuples**)
  - B+-Tree height 2 (3 I/Os from root to leaf)
  - R⋈S: 1000 + (100,000)*(3 +1) = 401,000

Sort-Merge Join

- 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**
  - 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
  }
}
```

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 2

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

sid	sname	bid
28	yuppy	103

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

VIRGINIA TECH

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
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 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
  }
}
```

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 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
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 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
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 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

VIRGINIA TECH™

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

VIRGINIA TECH™

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
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 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
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 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

VIRGINIA TECH™

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
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

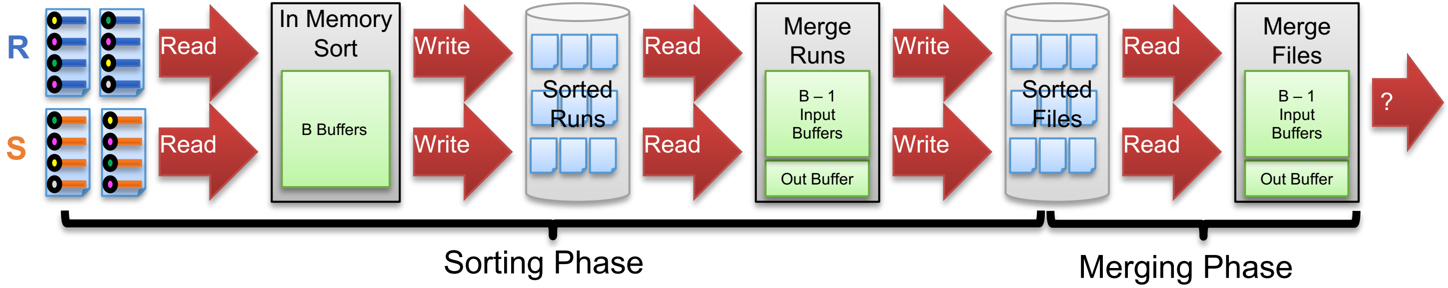
```
do {
  if (!mark) {
    while (r < s) { advance r }
    while (r > s) { advance s }
    // mark start of "block" of S
    mark = s
  }
  if (r == s) {
    result = <r, s>
    advance s
    return result
  }
  else {
    reset s to mark
    advance r
    mark = NULL
  }
}
```

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

Cost of Sort-Merge Join

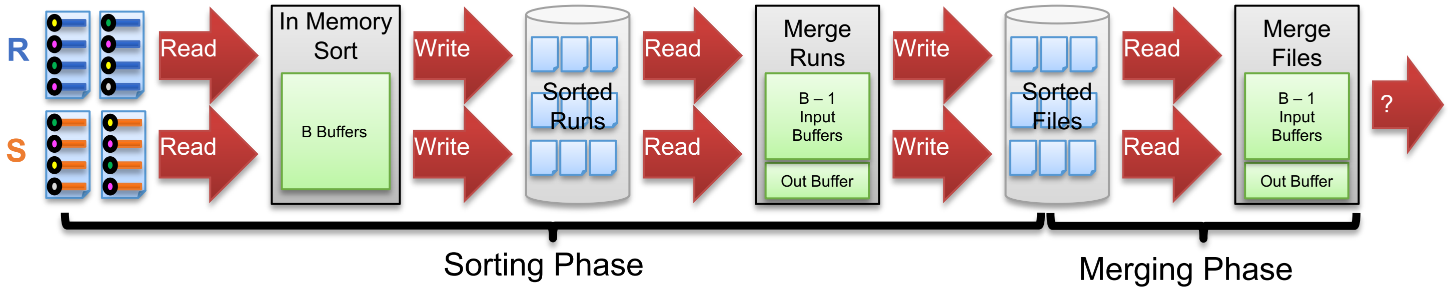


- Cost: Sort R + Sort S + ([R]+[S])
  - But in worst case, last term could be [R] * [S]
  - Q: what is worst case? All tuples in both relations contain the same value in the join attribute (very unlikely!)

- Assume we have 100 buffer pages to sort both R and S in two passes

- Require buffer B > $\sqrt{(max([R], [S]))}$
  – Both R and S can be sorted in 2 passes, and one merge pass
  – 2 * 2 *1000 + 2 * 2* 500 + (1000 + 500) = 7500

Sort-Merge Refinement



- An important refinement combines last pass of merge-sort with join pass
  - Given **enough buffers** to sort both relations simultaneously…
  - Do the 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 ⋈ S matches
  - 2-pass Cost = 3*[R] + 3*[S] = 3000+1500 = 4500
    - **Requires** $\mathbf{B} \geq \sqrt{R} + \sqrt{S}$ = 32 + 23 = 55

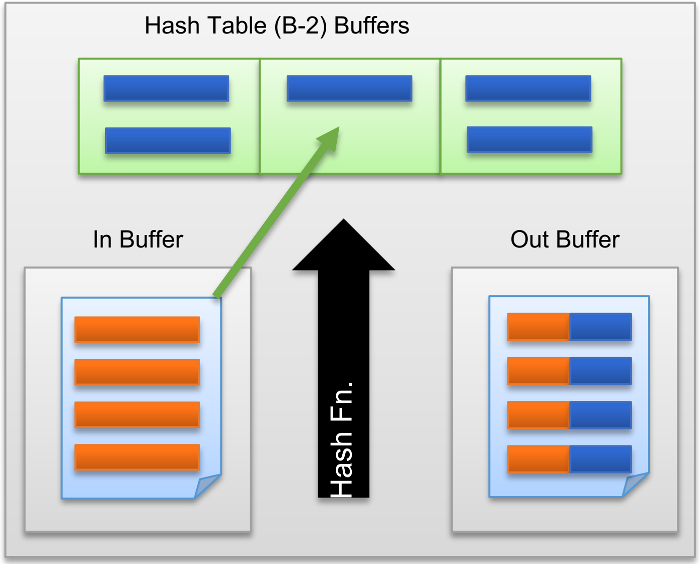
Sort-Merge Join

- Useful if
  - one or both inputs are already sorted on join attribute(s)
  - output is required to be sorted on join attributes(s)
- “Merge” phase can require some back tracking if duplicate values appear in join column

Naïve in Memory Hash 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) * hash_fill

What if R doesn't fit?



Properties that help

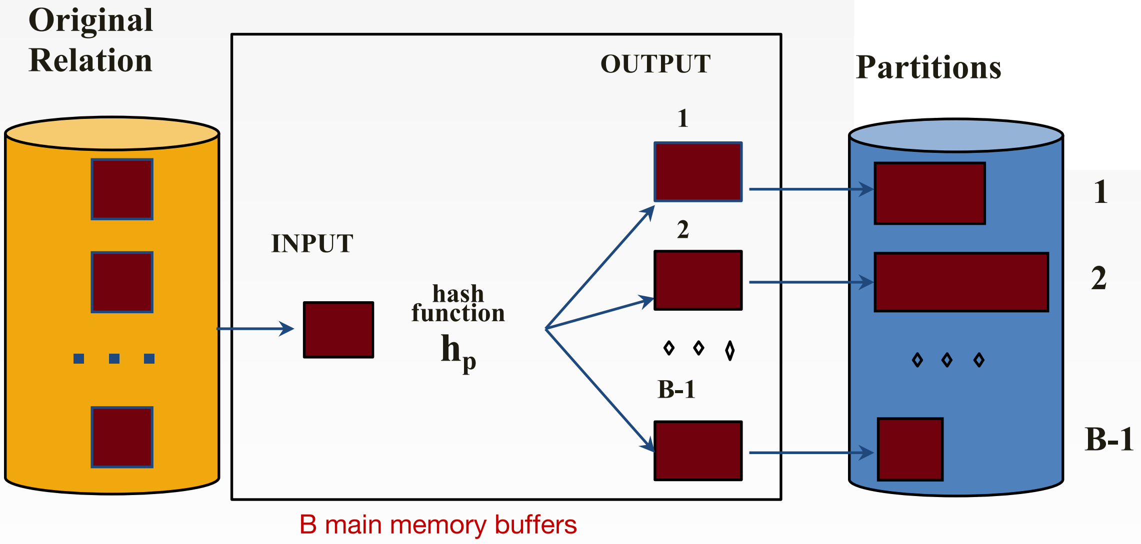
- $\sigma_{sid=4 \vee sid=6}(R \bowtie_{sid} S) = \sigma_{sid=4}(R \bowtie_{sid} S) \cup \sigma_{sid=6}(R \bowtie_{sid} S)$
- Can Decompose Into Smaller "Partial Joins"
- $R \bowtie_{sid} S = \cup (\sigma_{hash(sid)}(R) \bowtie_{sid} \sigma_{hash(sid)}(S))$
- Pick a hash function so that $\sigma_{hash(sid)}(R)$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

Grace Hash Join

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

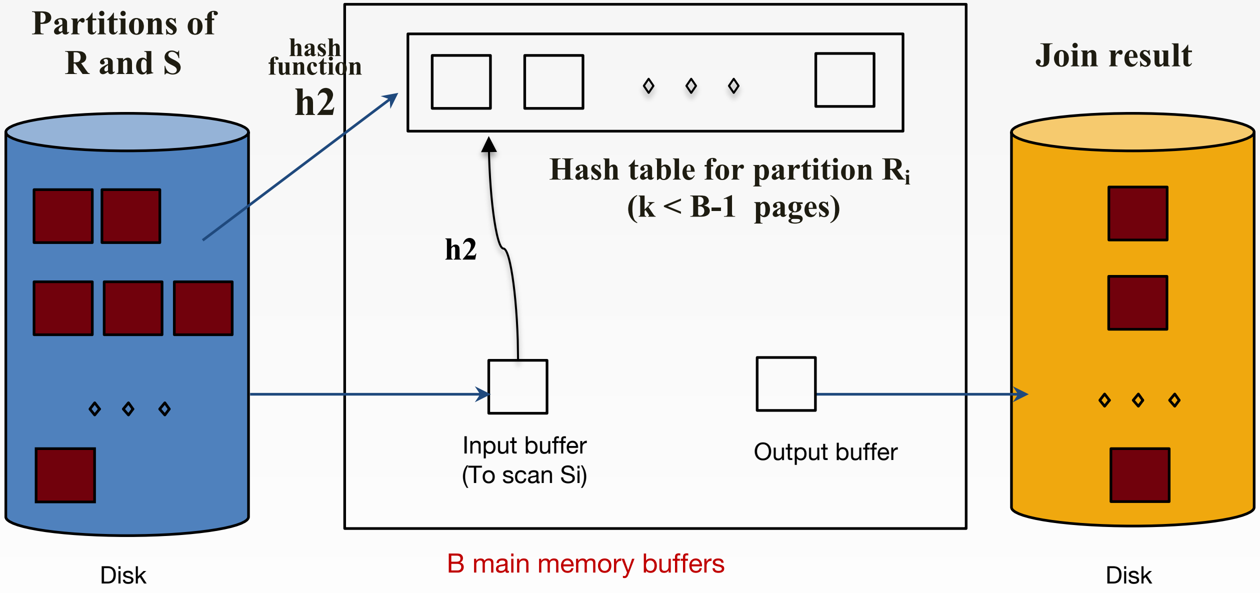
Grace Hash Join

Partitioning phase of Hash Join



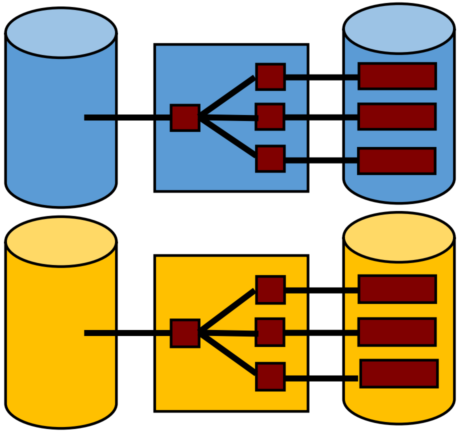
Grace Hash Join

Probing phase of Hash Join



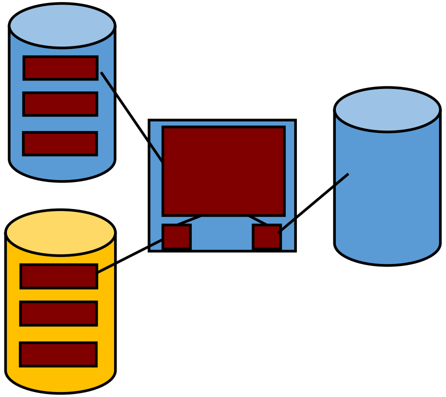
Grace Hash: Partition Pseudocode

```
For Cur in {R, S}
    For page in Cur
        Read page into input buffer
        For tup on page
            Place tup in output buf hash_p(tup.joinkey)
            If output buf full then flush to disk partition
    Flush output bufs to disk partitions
```

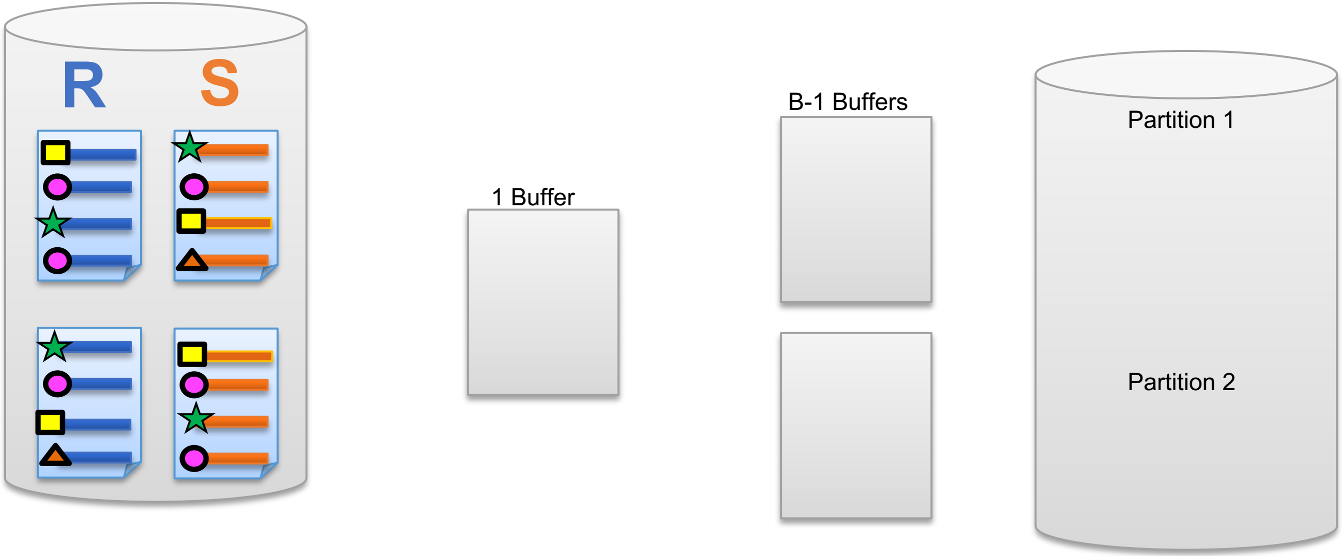


Grace Hash: Partition Pseudocode

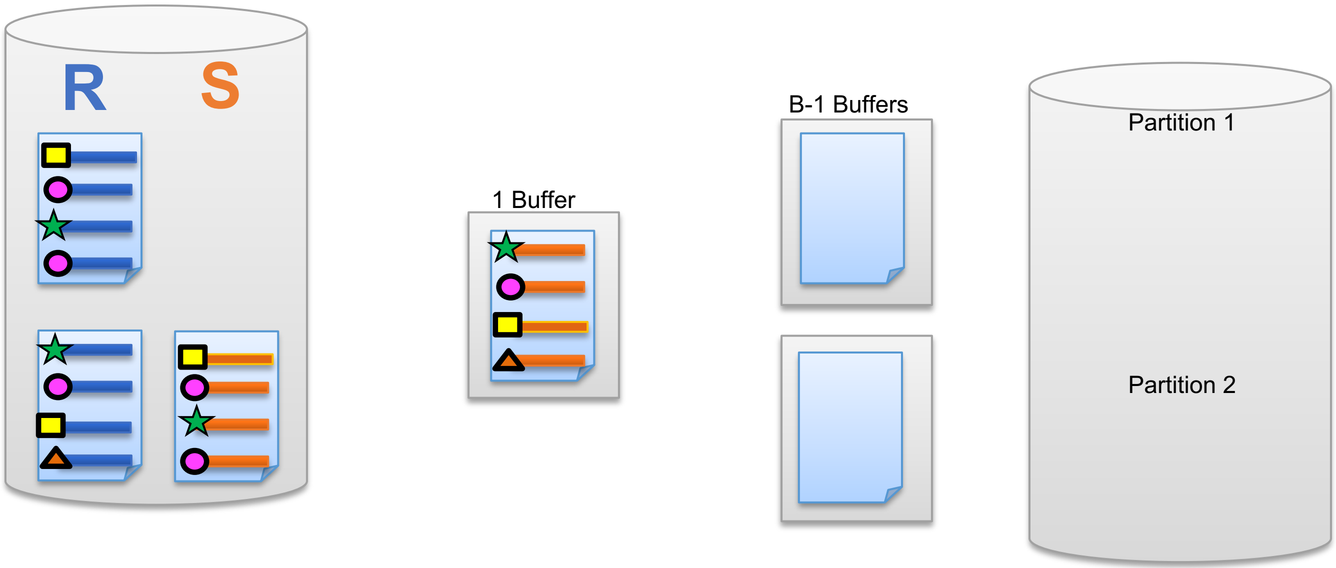
```
For i in [0..(B-1))
   For page in R_i
      For tup on page
         Build tup into memory hash_r(tup.joinkey)
   For page in S_i
      Read page into input buffer
      For tup on page
         Probe memory hash_r(tup.joinkey) for matches
         Send all matches to output buffer
         Flush output buffer if full
```



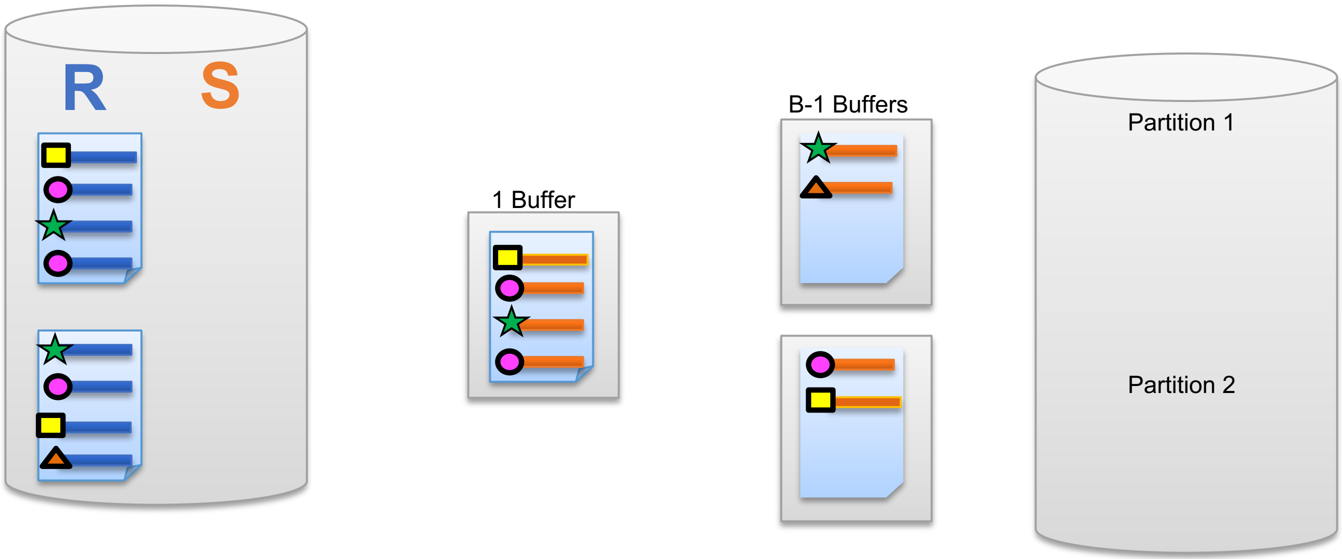
Grace Hash Join: *Partition*



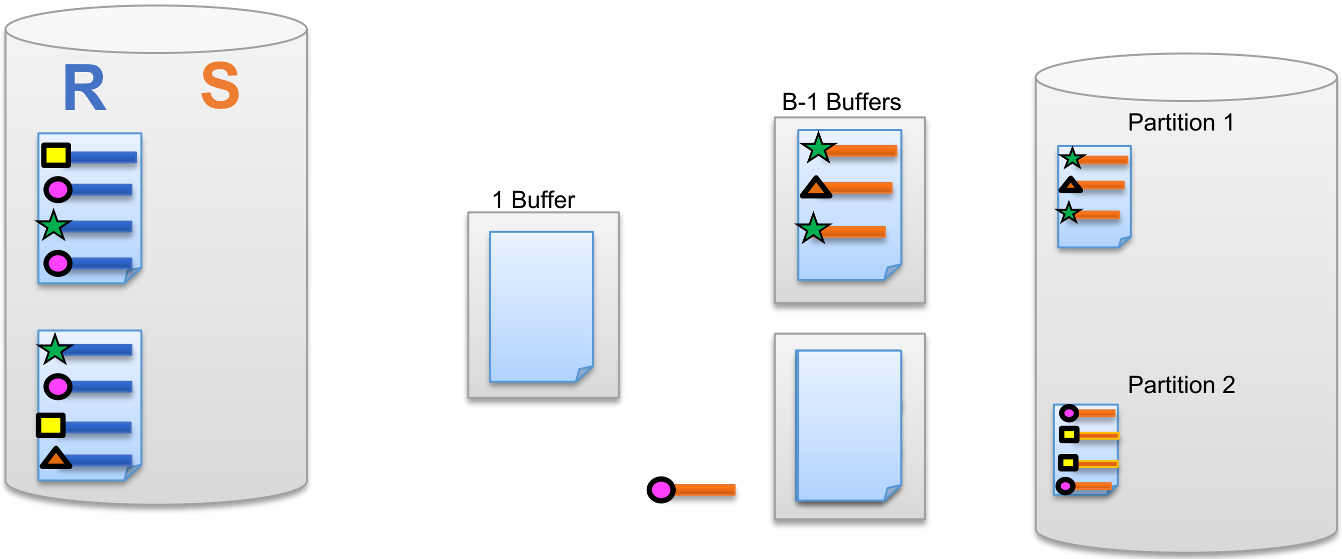
Grace Hash Join: *Partition, Part 2*



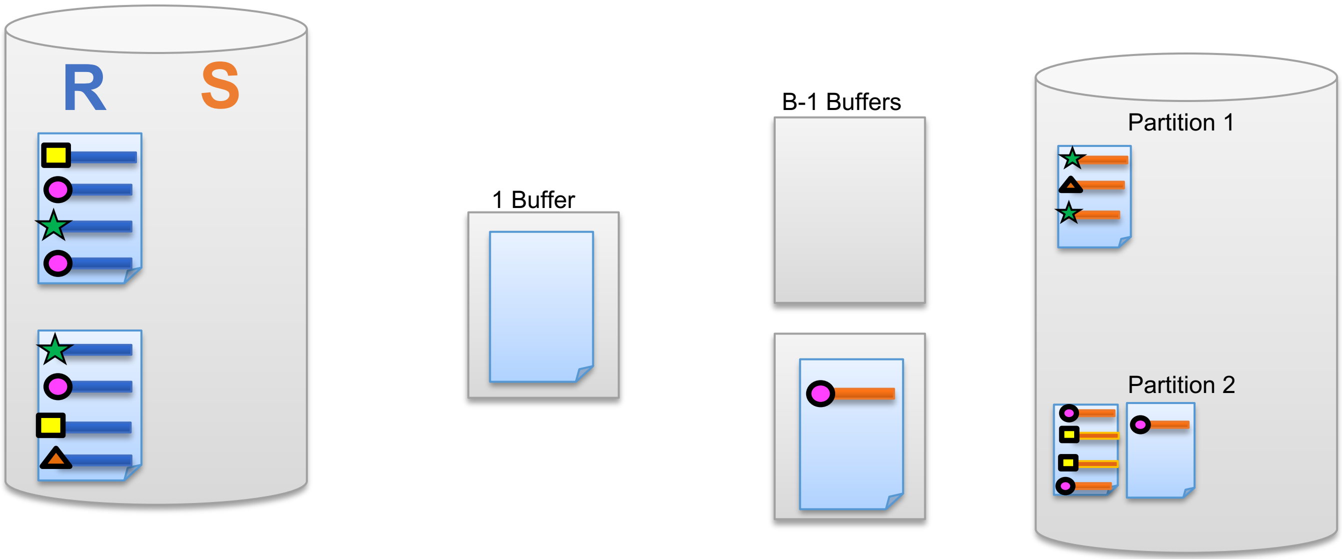
Grace Hash Join: *Partition, Part 3*



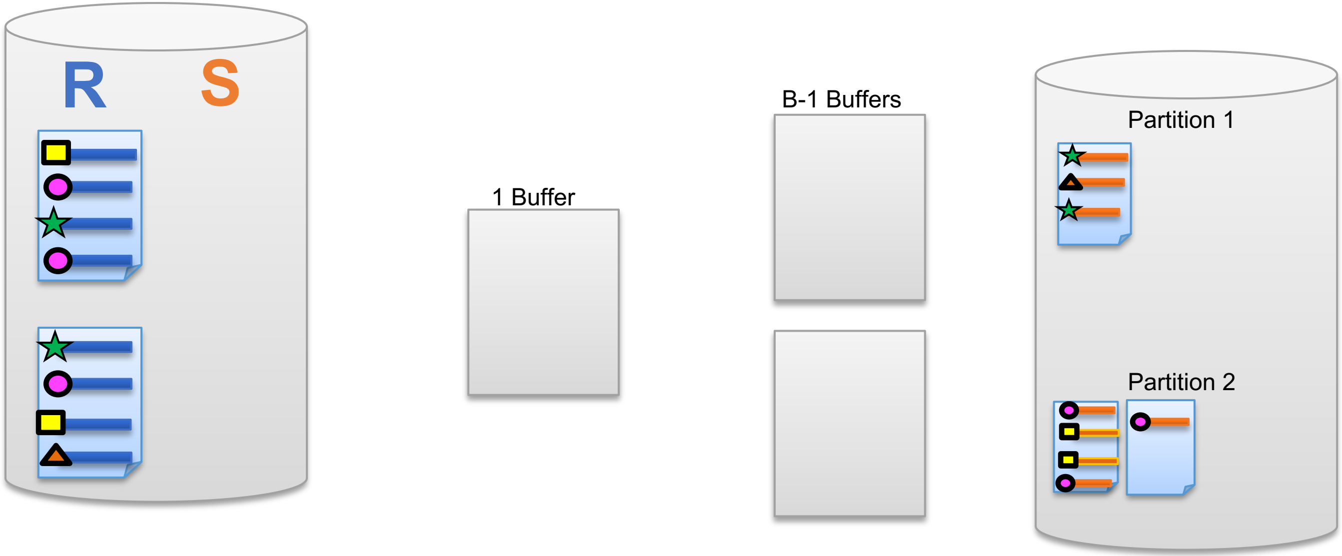
Grace Hash Join: *Partition Part 4*



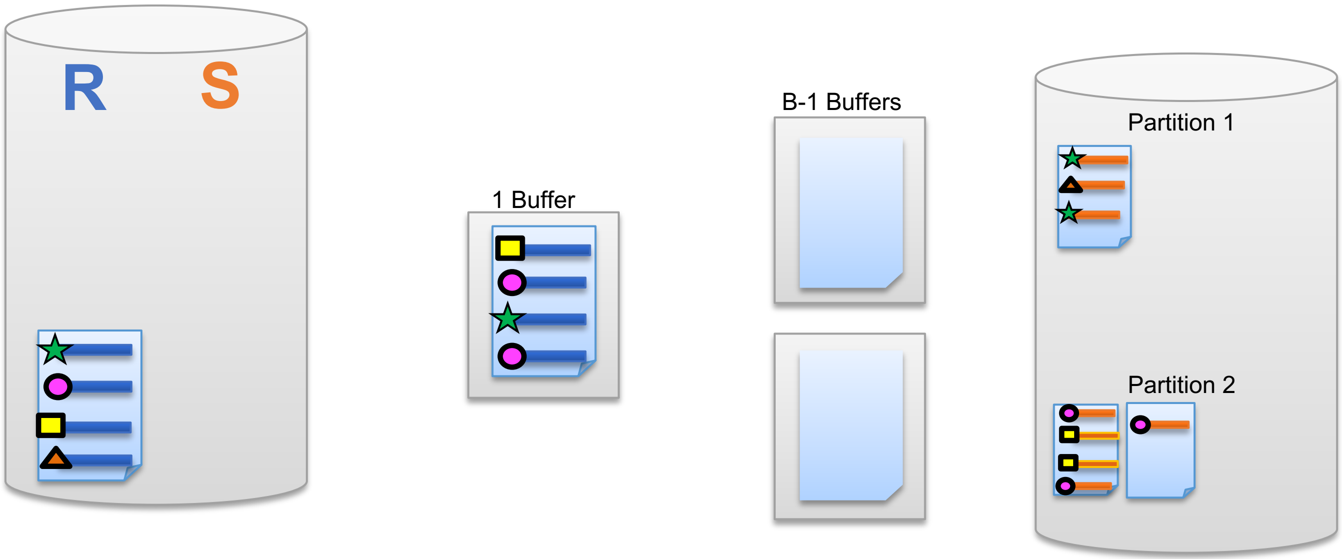
Grace Hash Join: *Partition Part 5*



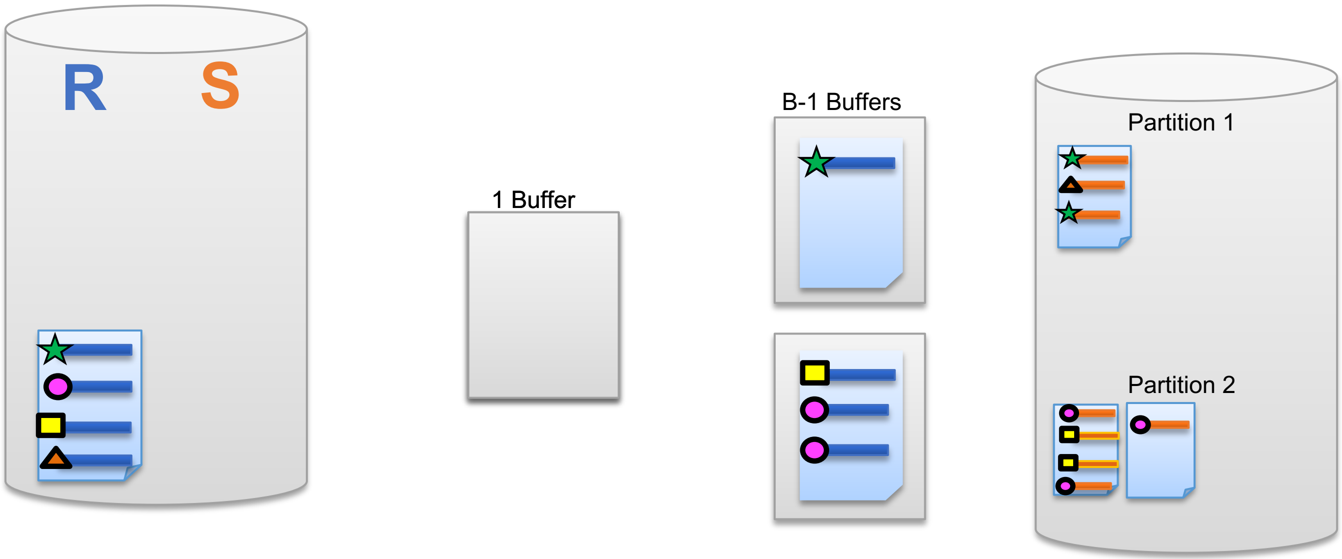
Grace Hash Join: *Partition Part 6*



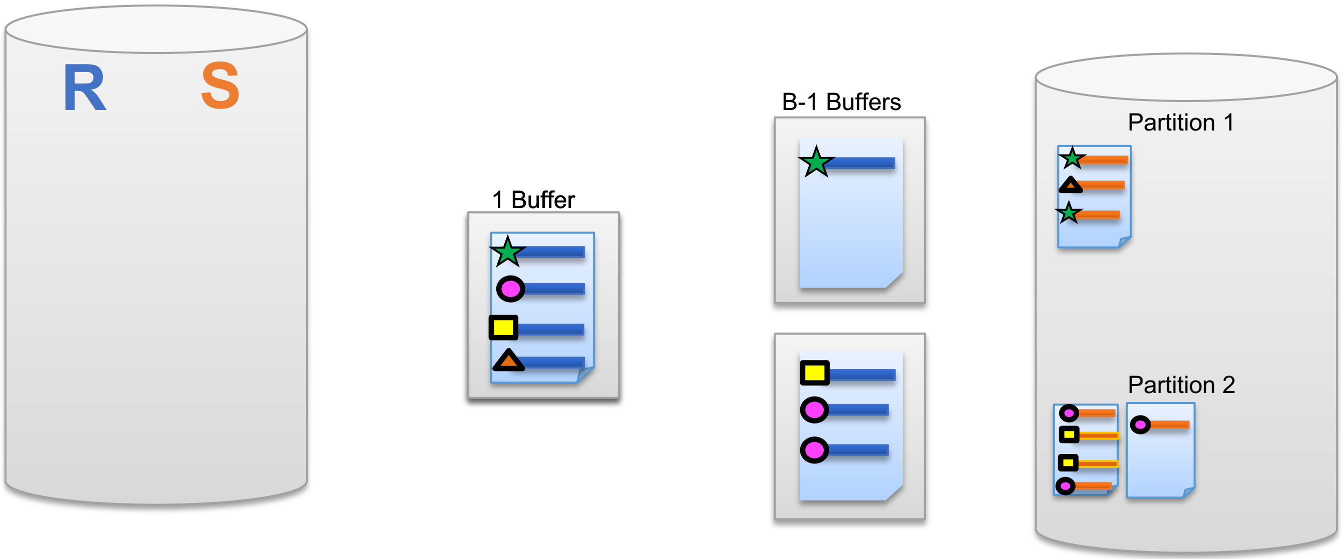
Grace Hash Join: *Partition Part 7*



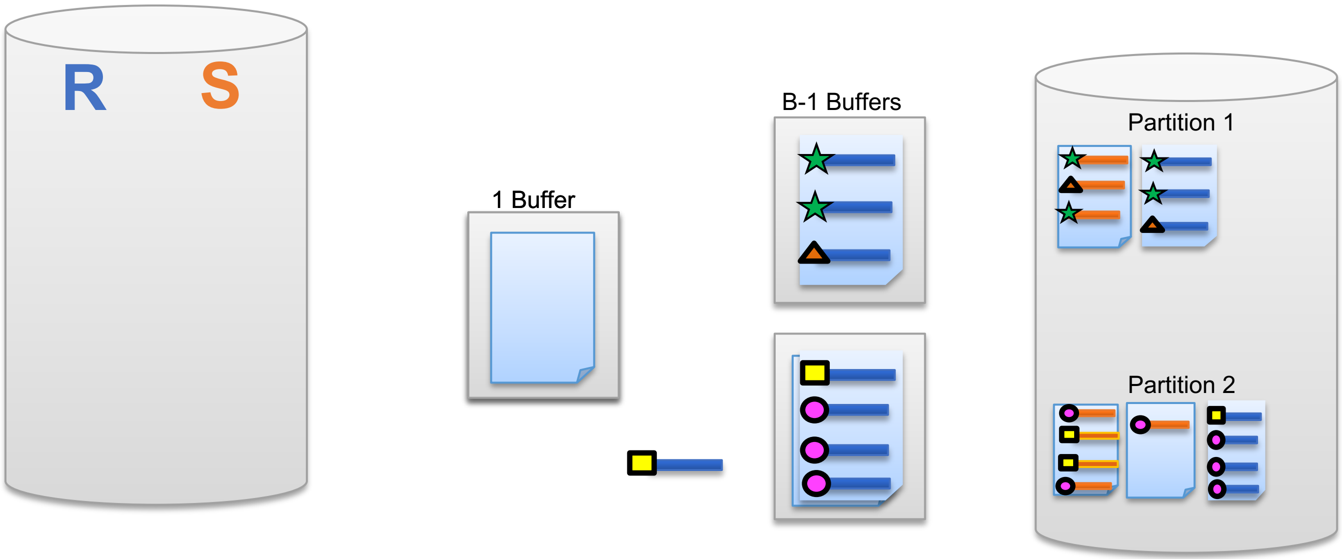
Grace Hash Join: ***Partition Part 8***



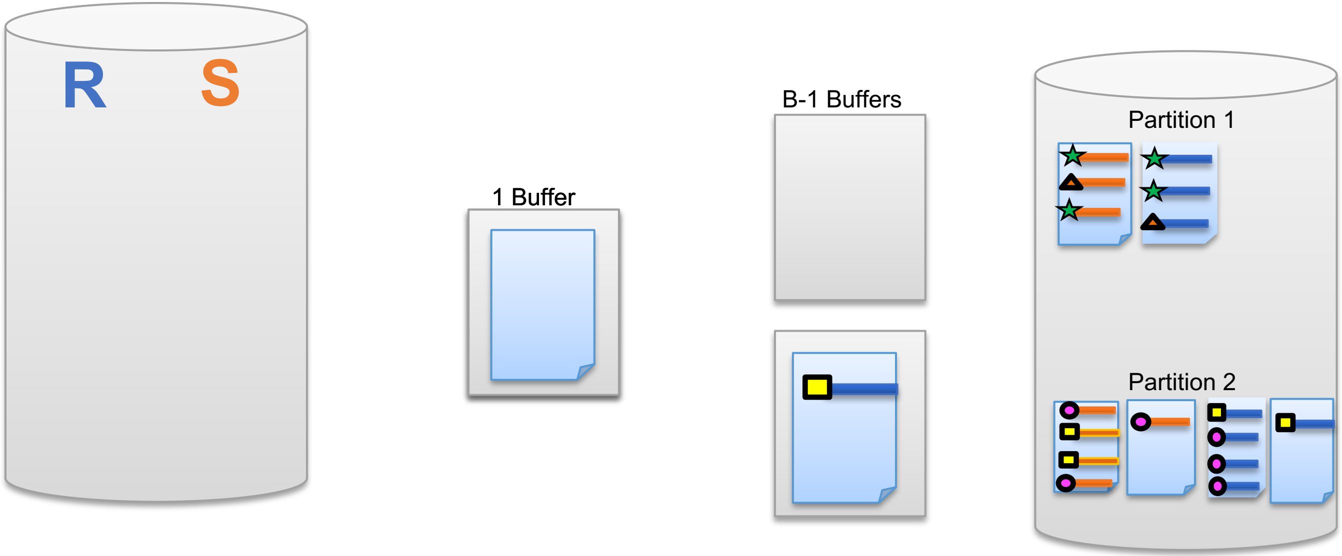
Grace Hash Join: *Partition 9*



Grace Hash Join: *Partition Part 10*

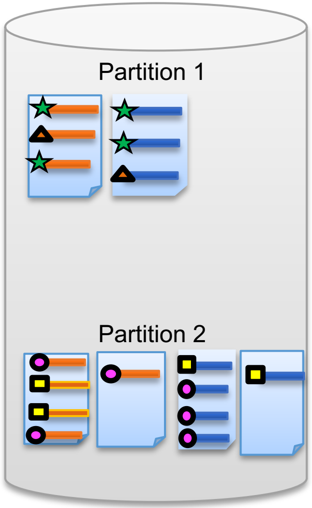


Grace Hash Join: *Partition Part 11*



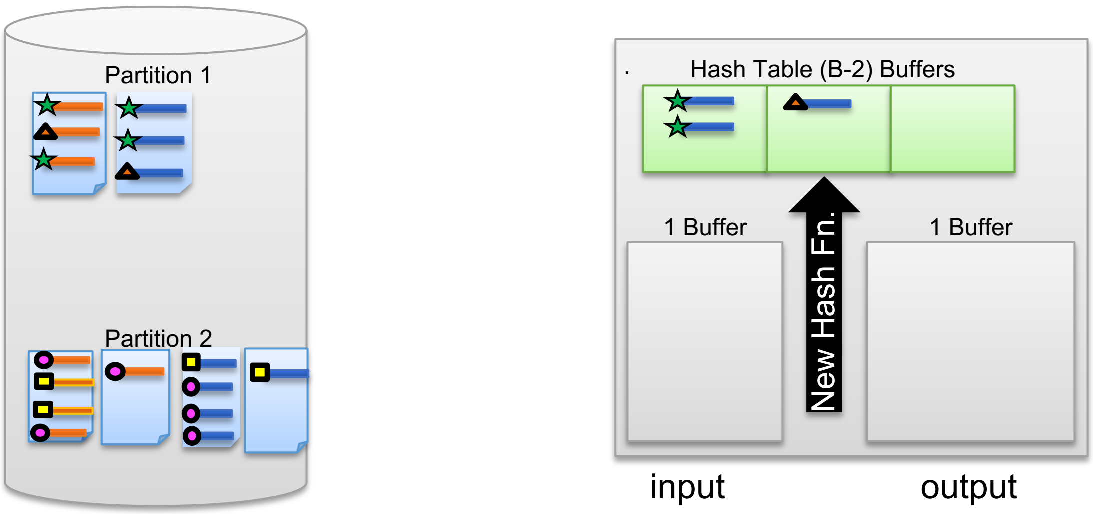
Grace Hash Join: *Partition Part 12*

- 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

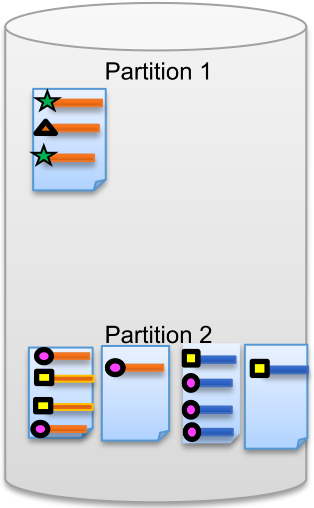


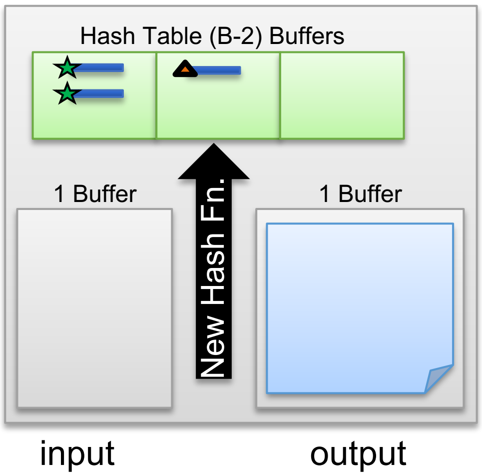
VIRGINIA TECH

Grace Hash Join: *Build* & *Probe*

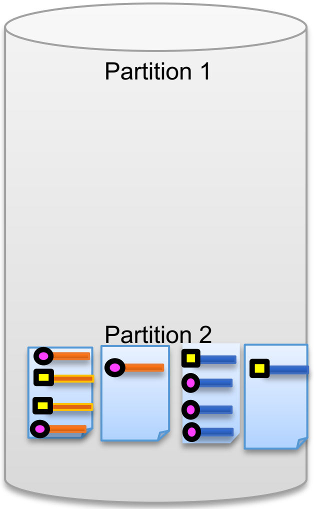


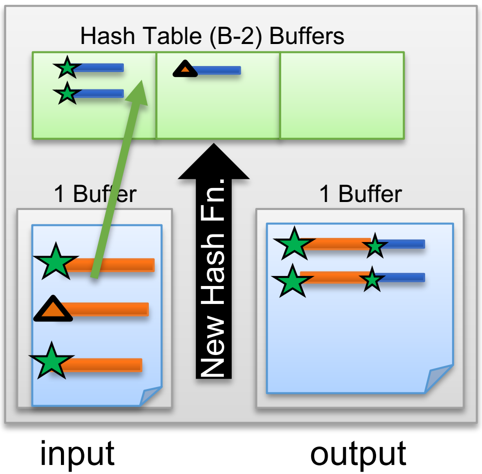
Grace Hash Join: *Build & Probe Part 2*



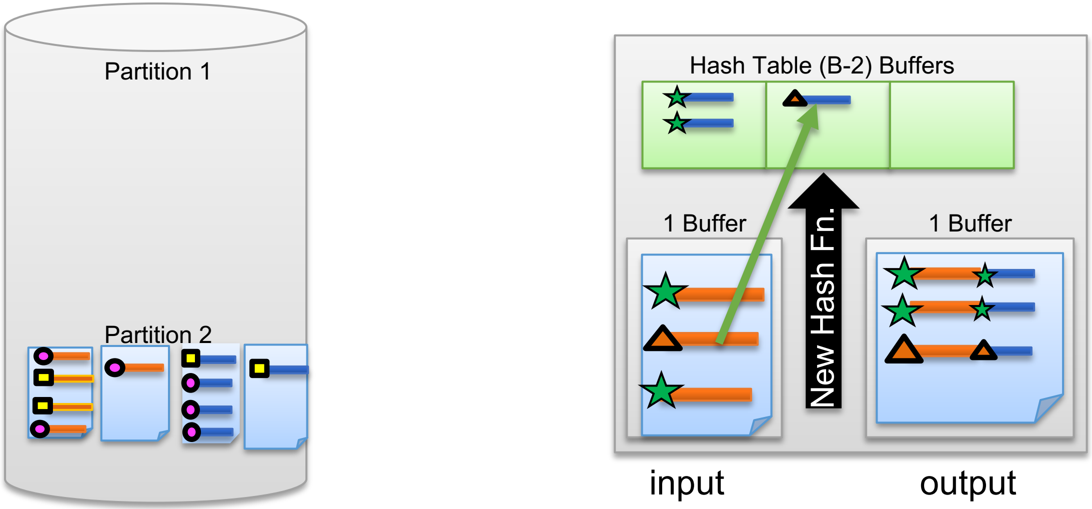


Grace Hash Join: *Build & <u>Probe Part 3</u>*

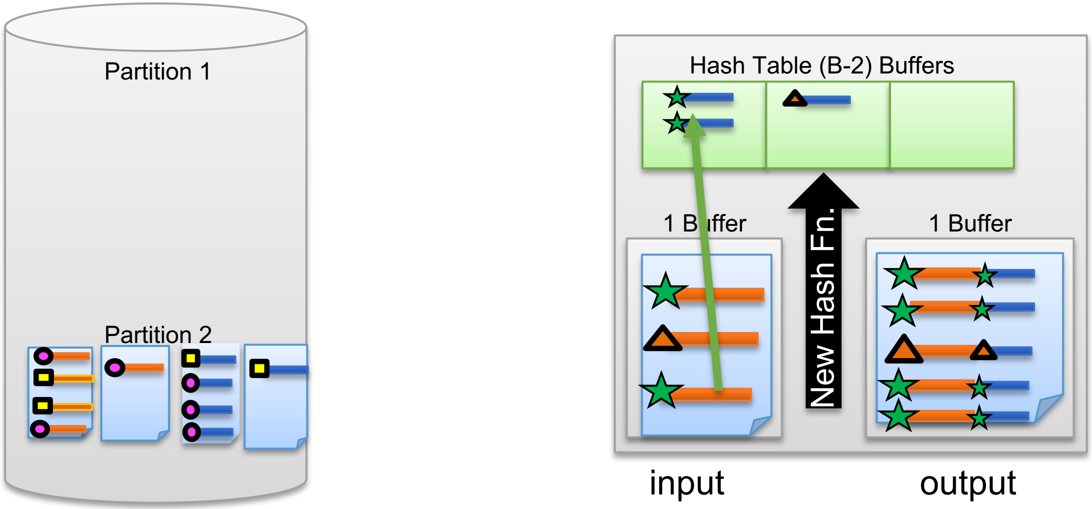




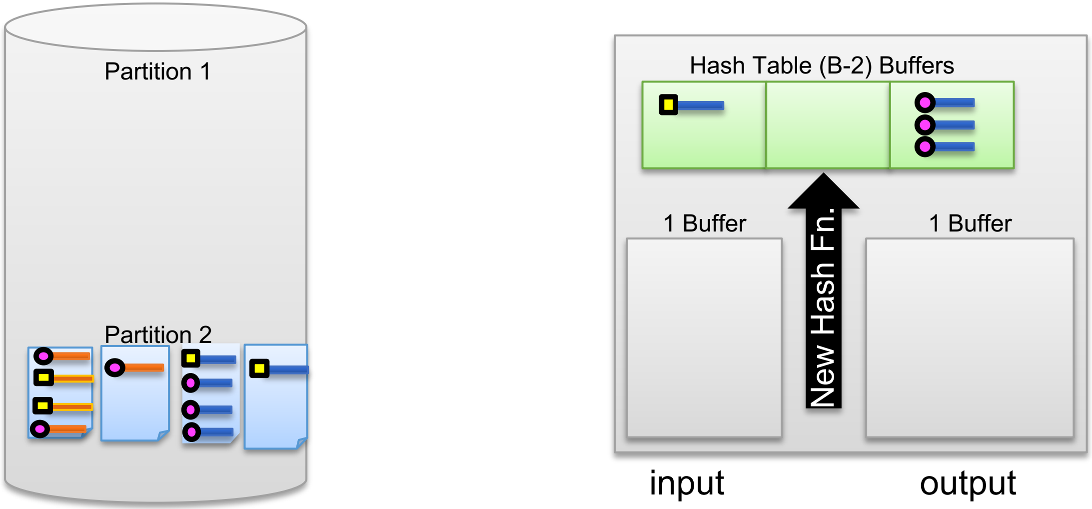
Grace Hash Join: *Build & Probe Part 4*



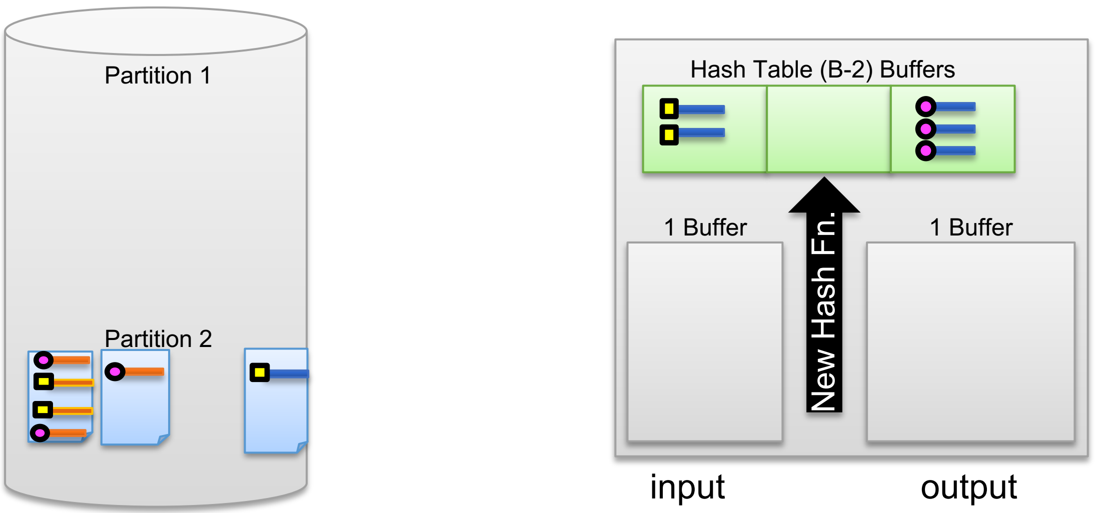
Grace Hash Join: *Build & <u>Probe Part 5</u>*



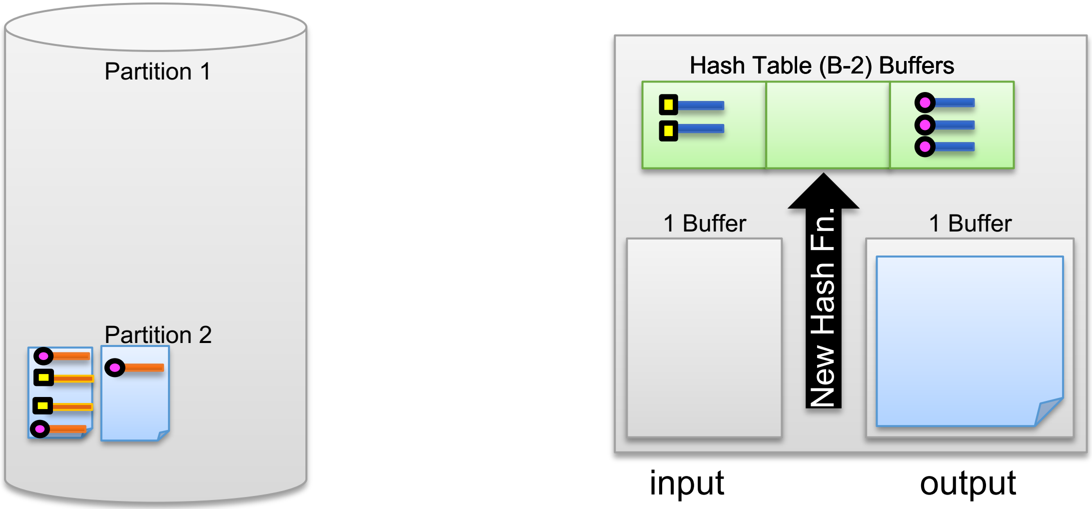
Grace Hash Join: *Build* & *Probe Part 6*



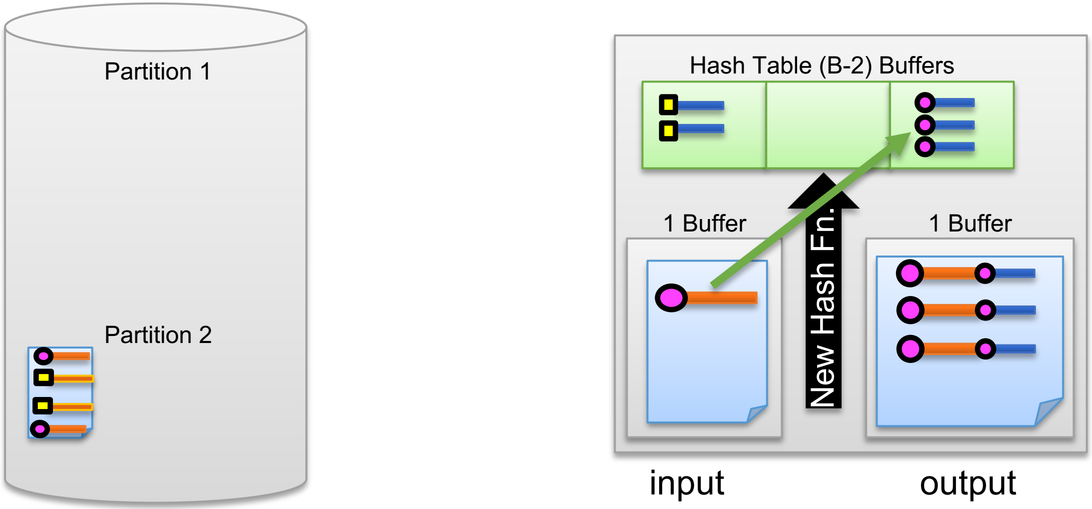
Grace Hash Join: *Build & Probe Part 7*



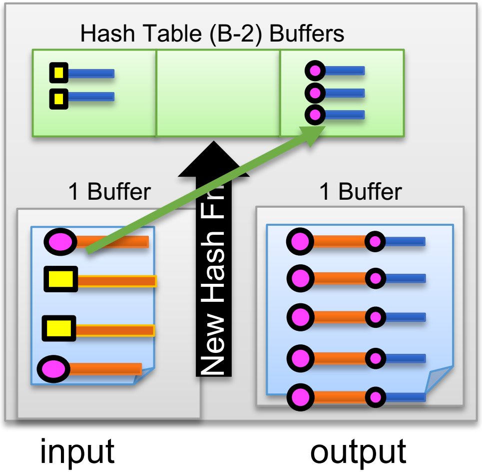
Grace Hash Join: *Build & Probe Part 8*



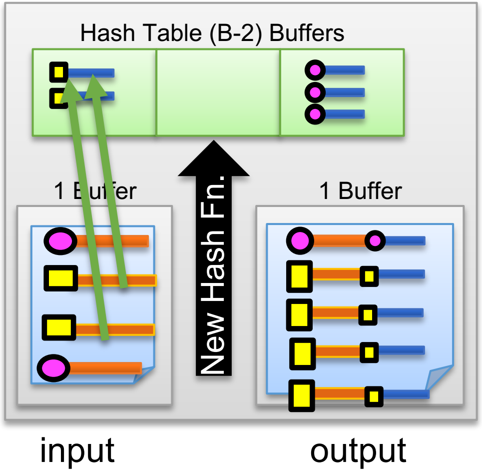
Grace Hash Join: *Build & Probe Part 9*



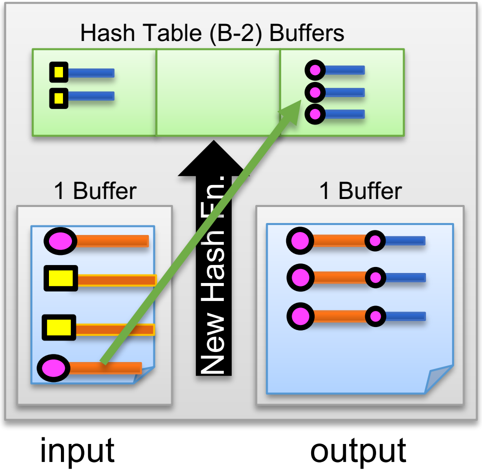
Grace Hash Join: *Build & Probe Part 10*



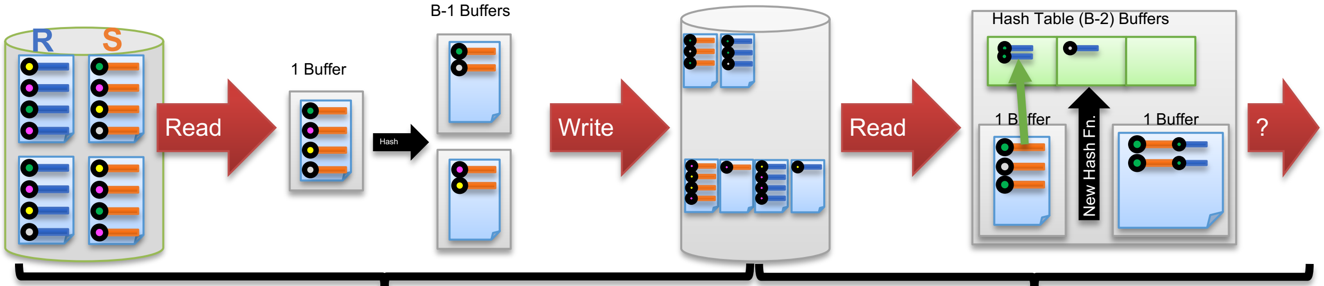
Grace Hash Join: *Build & Probe Part 11*



Grace Hash Join: *Build & Probe Part 12*



Summary of Grace Hash Join



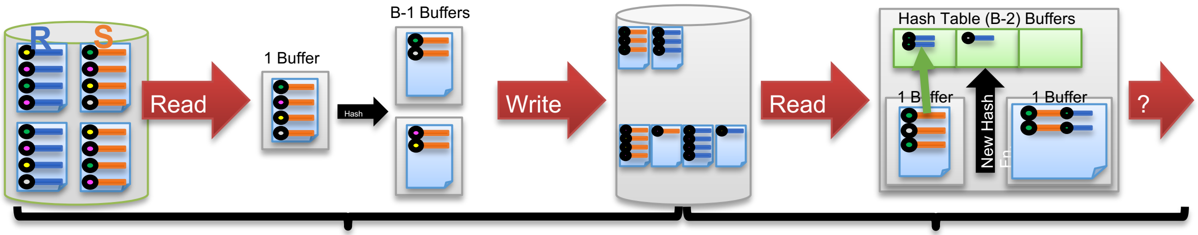
Partitioning Phase:
B-1 partitions

Build & Probe Phase
Each partition of R of size B-2

What is the Cost?

Cost of Hash Join

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



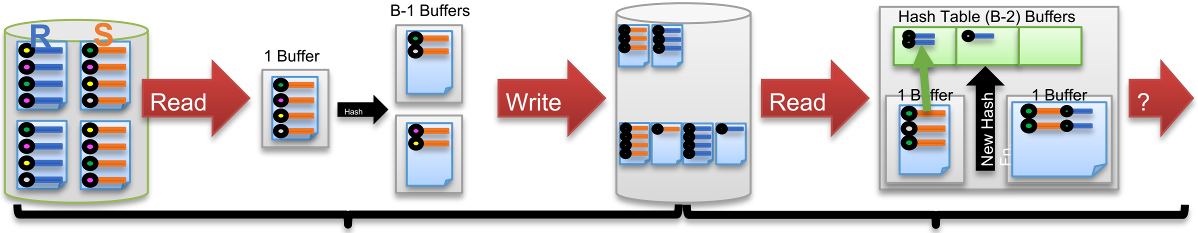
Partitioning Phase:
B-1 partitions

Build & Probe Phase
Each partition of R of size B-2

- Partitioning phase: read+write both relations
  ⇒2([R]+[S]) I/Os
- Matching phase: read both relations, forward output
  ⇒[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



Partitioning Phase:
B-1 partitions

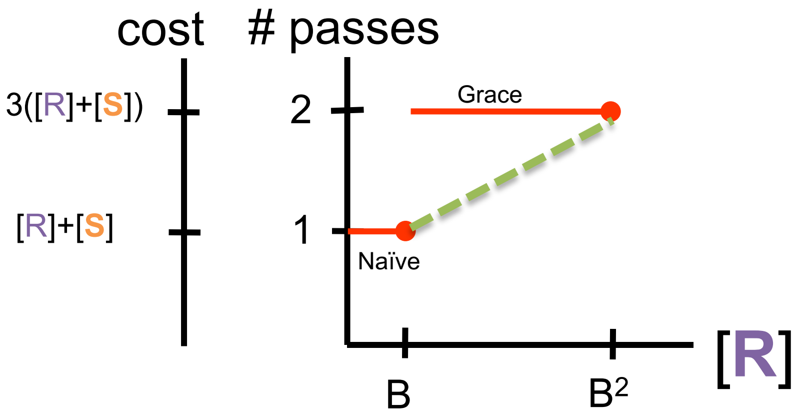
Build & Probe Phase
Each partition of R of size B-2

- **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) ≈ 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] **< B²**
- **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 Buffer is enough to hold smaller relation, 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!

Summary

- A virtue of relational DBMSs:
  - Queries are composed of a few basic operators
  - The implementation of these operators can be carefully tuned
- Many alternative implementation techniques for each operator
  - No universally superior technique for most operators
- Must consider available alternatives for each operation in a query and choose best one based on system statistics, etc.
  - Part of the broader task of optimizing a query composed of several ops

Reading and Next Class

- Query Processing: Ch 12, Ch14
- Next: Midterm review

Credits

- The animation Page 21-93 and 104-127 and some slides are adopted from UC Berkeley CS W 186.