# Welcome to CS 186, Section 3!

**TA:** Bryan Munar
**OH:** Mondays 11-12pm and Thursdays 2:30-3:30pm (651 Soda)
**DISC:** Tuesdays 11-12am (136 Barrows) and Wednesdays 10-11am (130 Wheeler)

# Announcements and Such

- Project/Homework 2 Released later this week!!

- Sign up to be partners with someone if you would like!

# Discussion 3: Join Algorithms!

# Overview:

1. Join Algorithms
2. Worksheet exercises
3. More Join Algorithms
4. Worksheet exercises

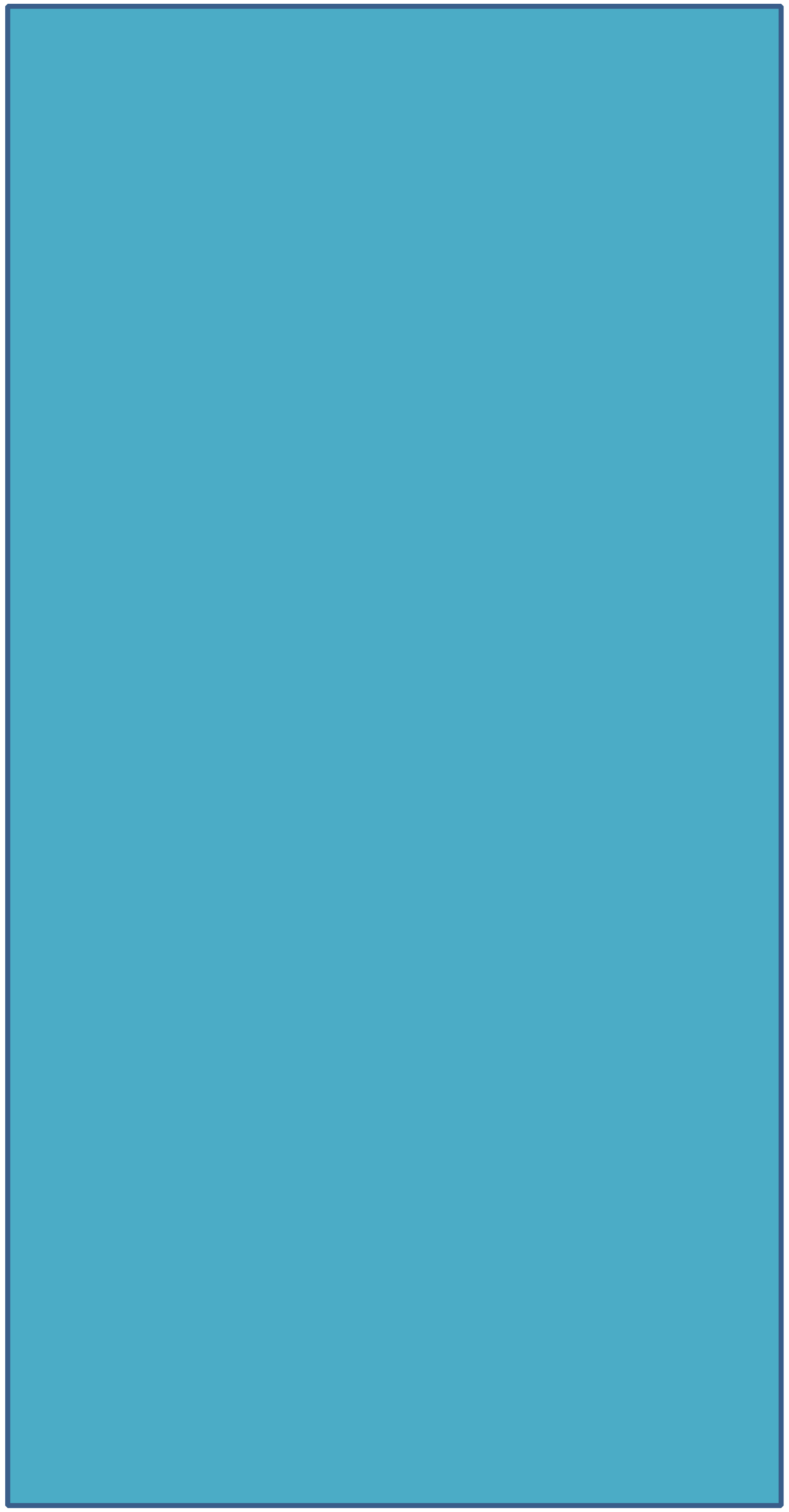
(A majority of the joins slides are from Michelle! Thank her!)

*Only going through second half of worksheet

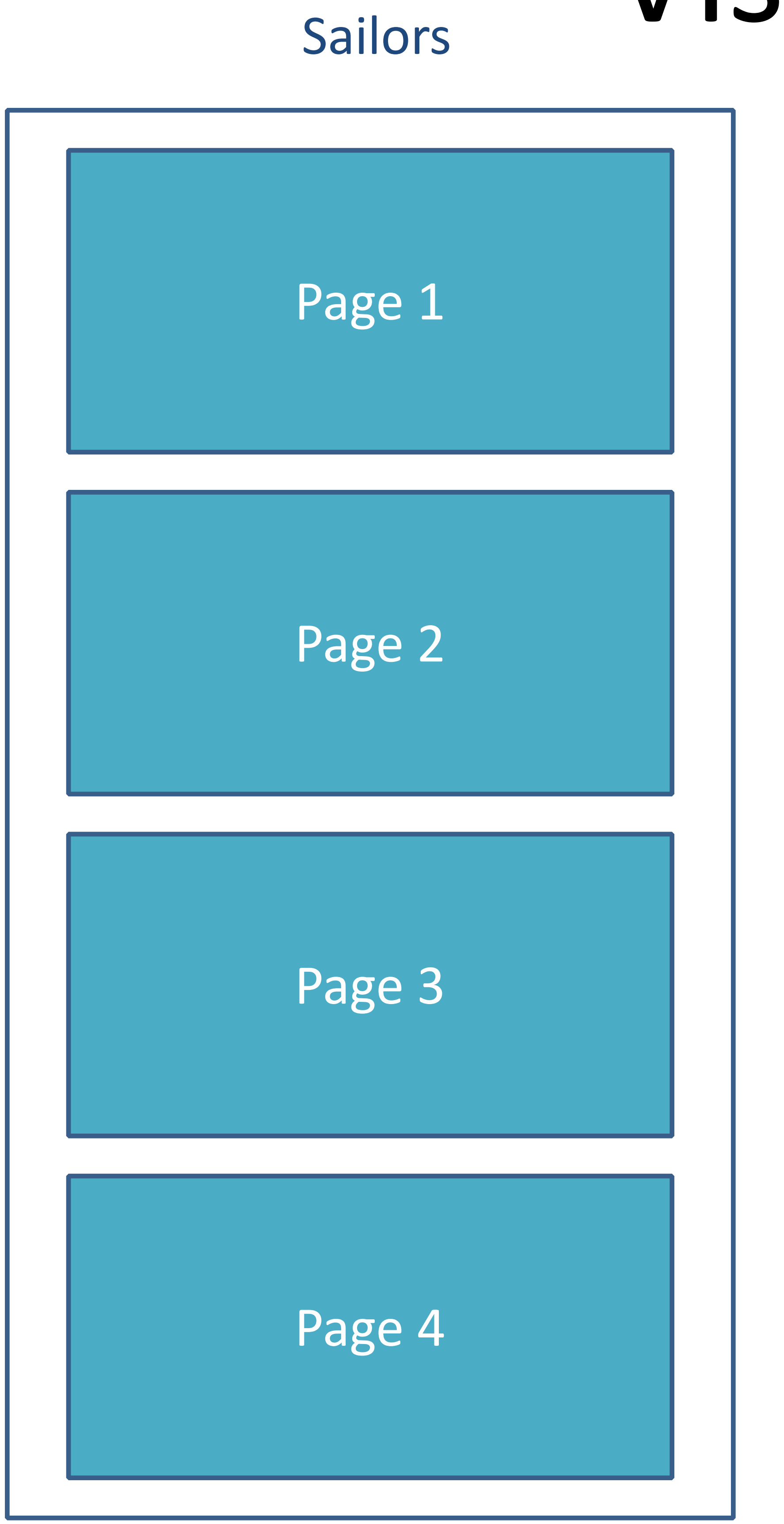
```sql
SELECT * FROM Sailors S, Reserves R
WHERE S.sid = R.sid;
```

# Visualizations

Sailors



# Visualizations



Sailors

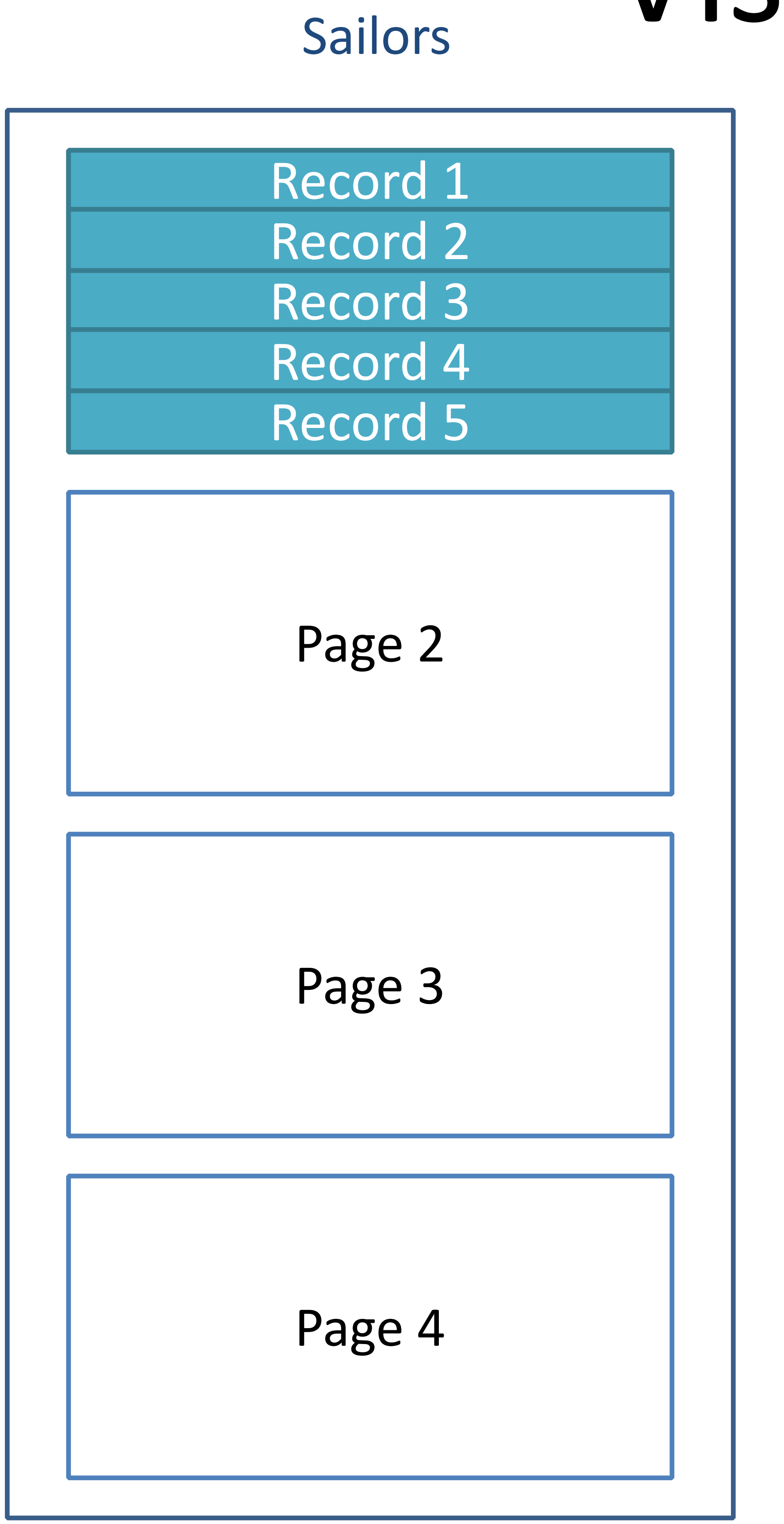
Page 1

Page 2

Page 3

Page 4

# Visualizations



Sailors

Record 1

Record 2

Record 3

Record 4

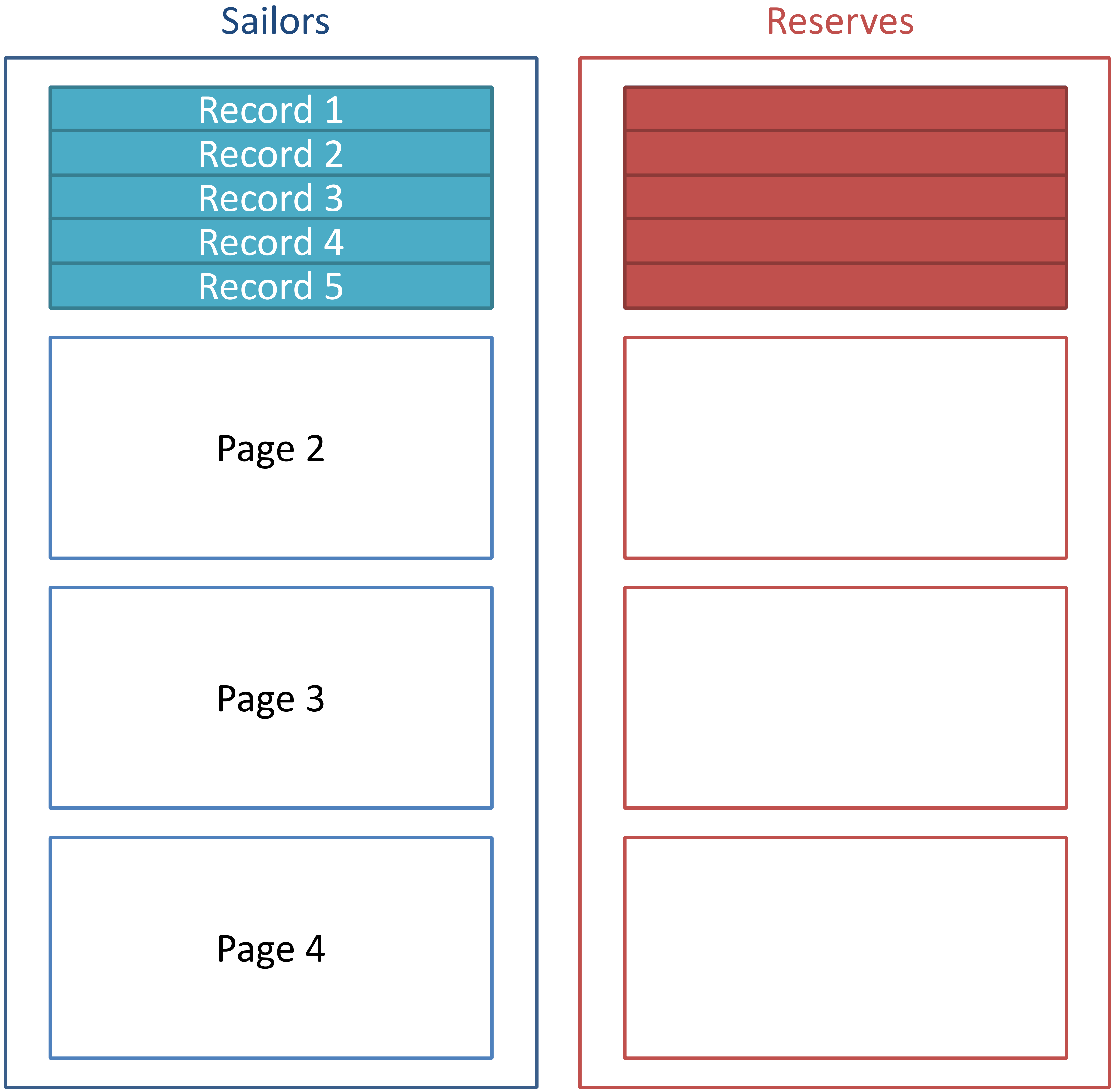
Record 5

Page 2

Page 3

Page 4

# Visualizations



Sailors

Reserves

Record 1

Record 2

Record 3

Record 4

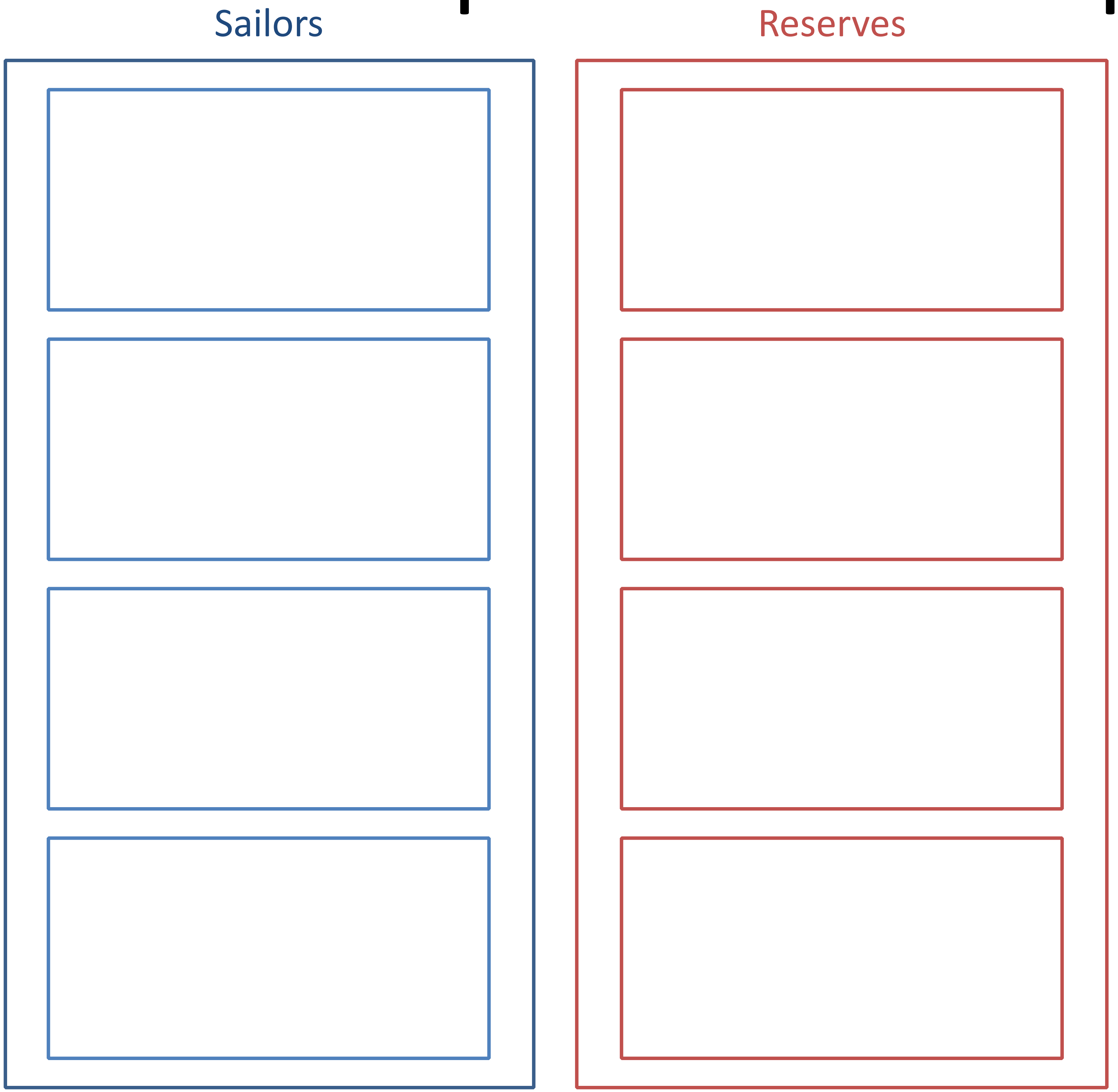
Record 5

Page 2

Page 3

Page 4

# Simple Nested Loops Join



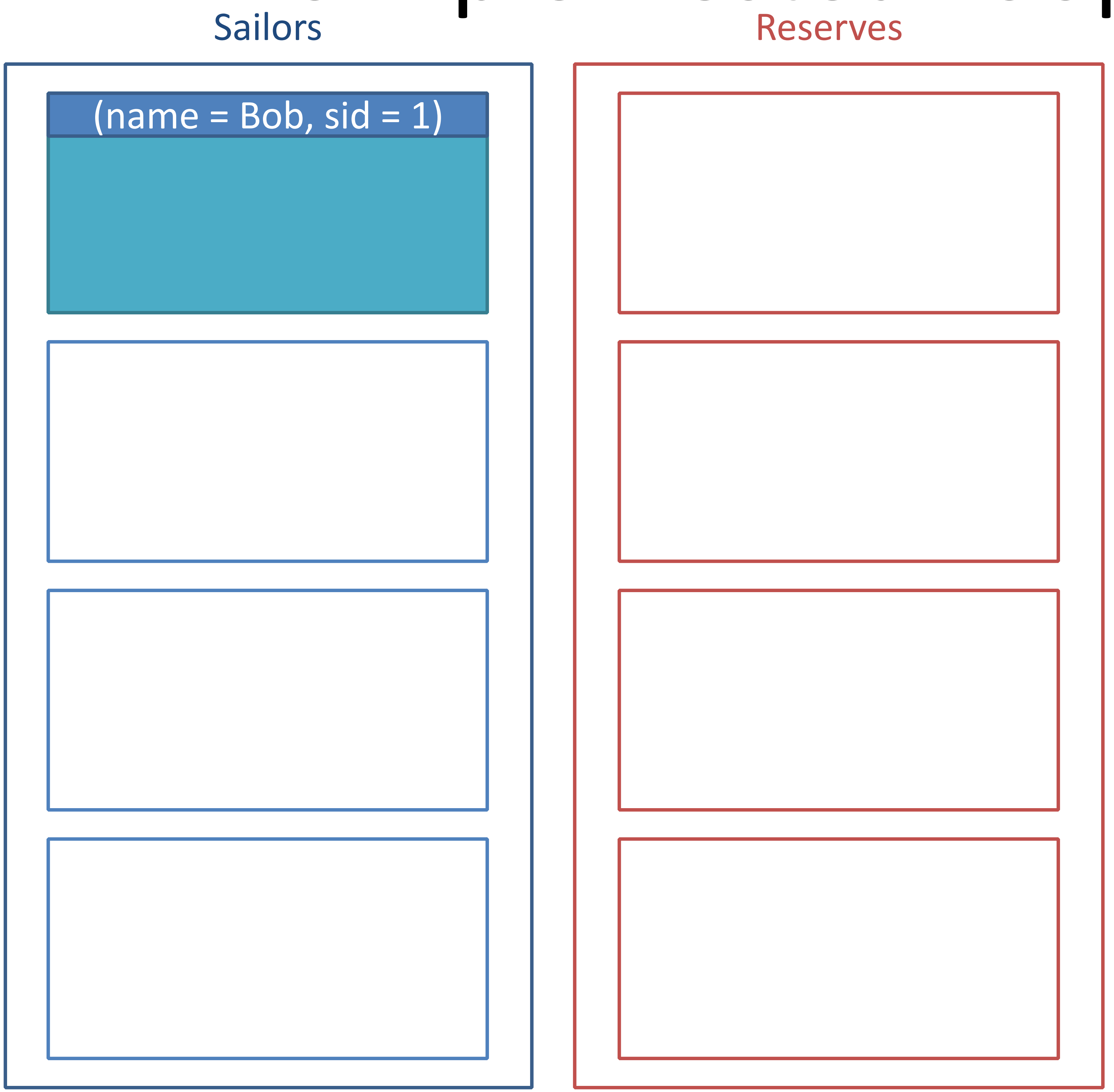
**Key idea:**

Take each record of S and match it with each record of R.

**Steps:**

1. Get tuple of S.
2. Iterate through each tuple in R.

# Simple Nested Loops Join



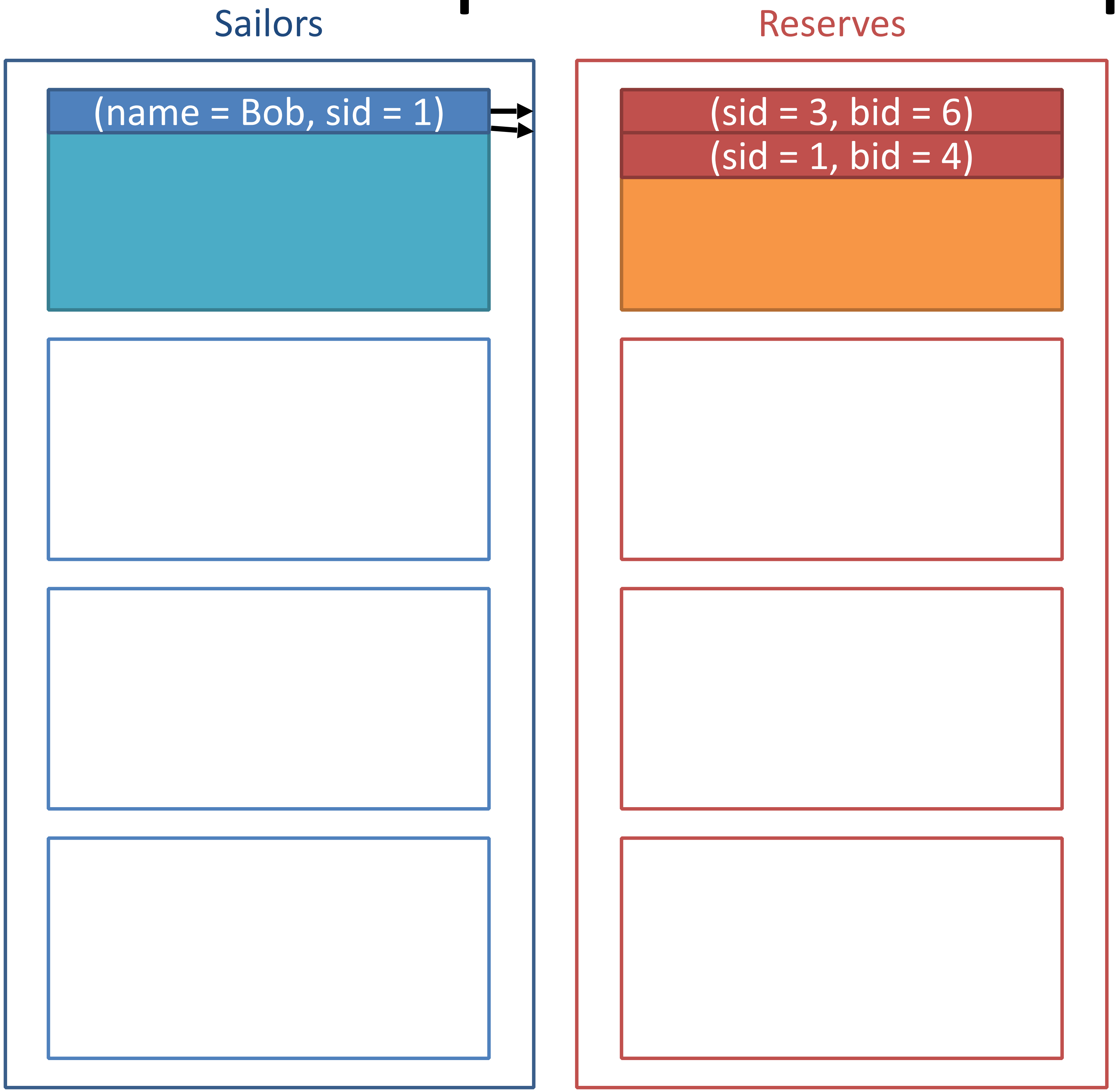
**Key idea:**

Take each record of S and match it with each record of R.

**Steps:**

1. Get tuple of S.
2. Iterate through each tuple in R.

# Simple Nested Loops Join



Sailors

(name = Bob, sid = 1)

Reserves

(sid = 3, bid = 6)

(sid = 1, bid = 4)

**Key idea:**

Take each record of S and match it with each record of R.

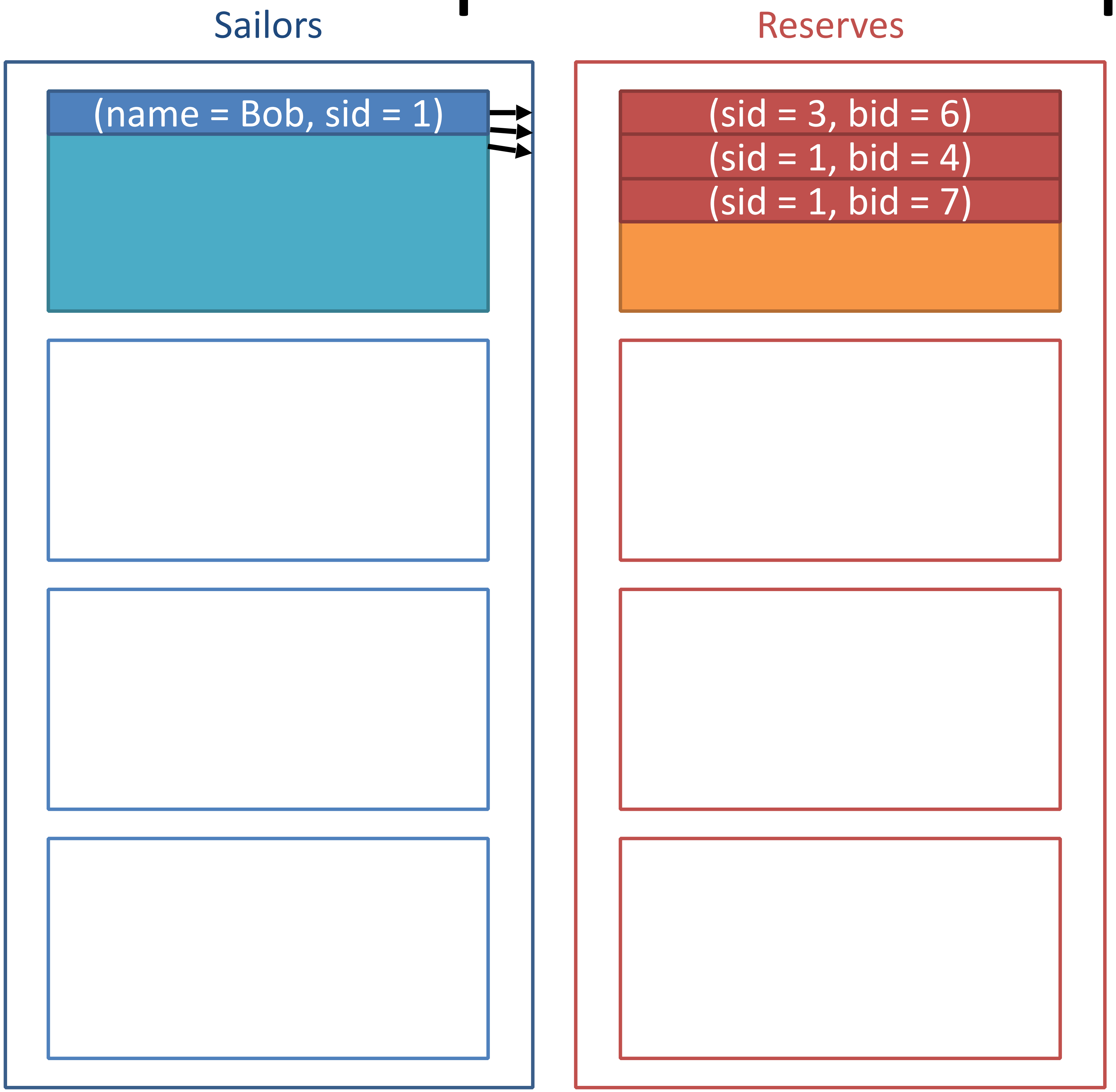
**Steps:**

1. Get tuple of S.
2. Iterate through each tuple in R.

**Output:**

(name = Bob, sid = 1, bid = 4)

# Simple Nested Loops Join



Sailors

(name = Bob, sid = 1)

Reserves

(sid = 3, bid = 6)

(sid = 1, bid = 4)

(sid = 1, bid = 7)

**Key idea:**

Take each record of S and match it with each record of R.

**Steps:**

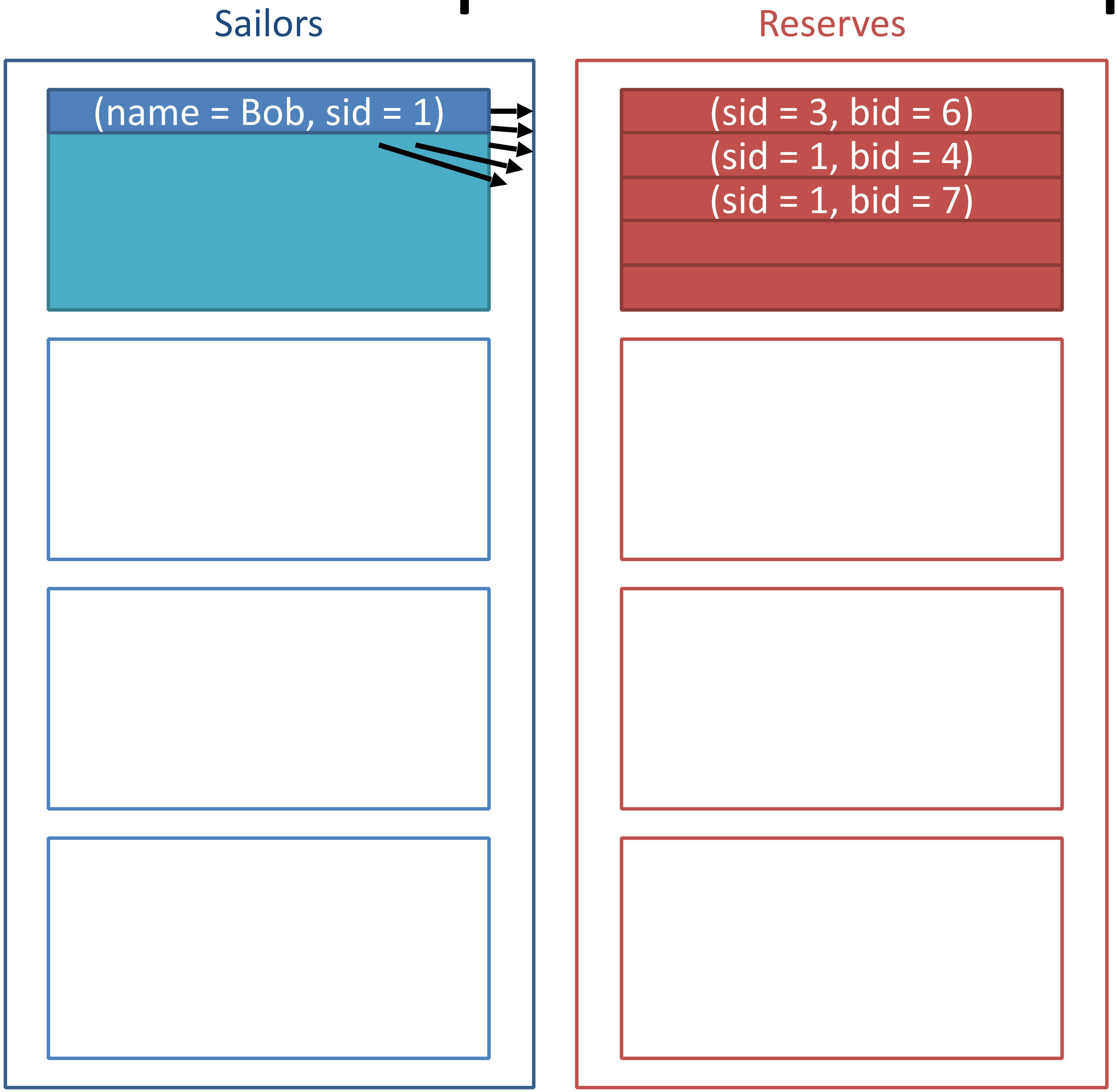
1. Get tuple of S.
2. Iterate through each tuple in R.

**Output:**

(name = Bob, sid = 1, bid = 4)

(name = Bob, sid = 1, bid = 7)

# Simple Nested Loops Join



**Key idea:**

Take each record of S and match it with each record of R.

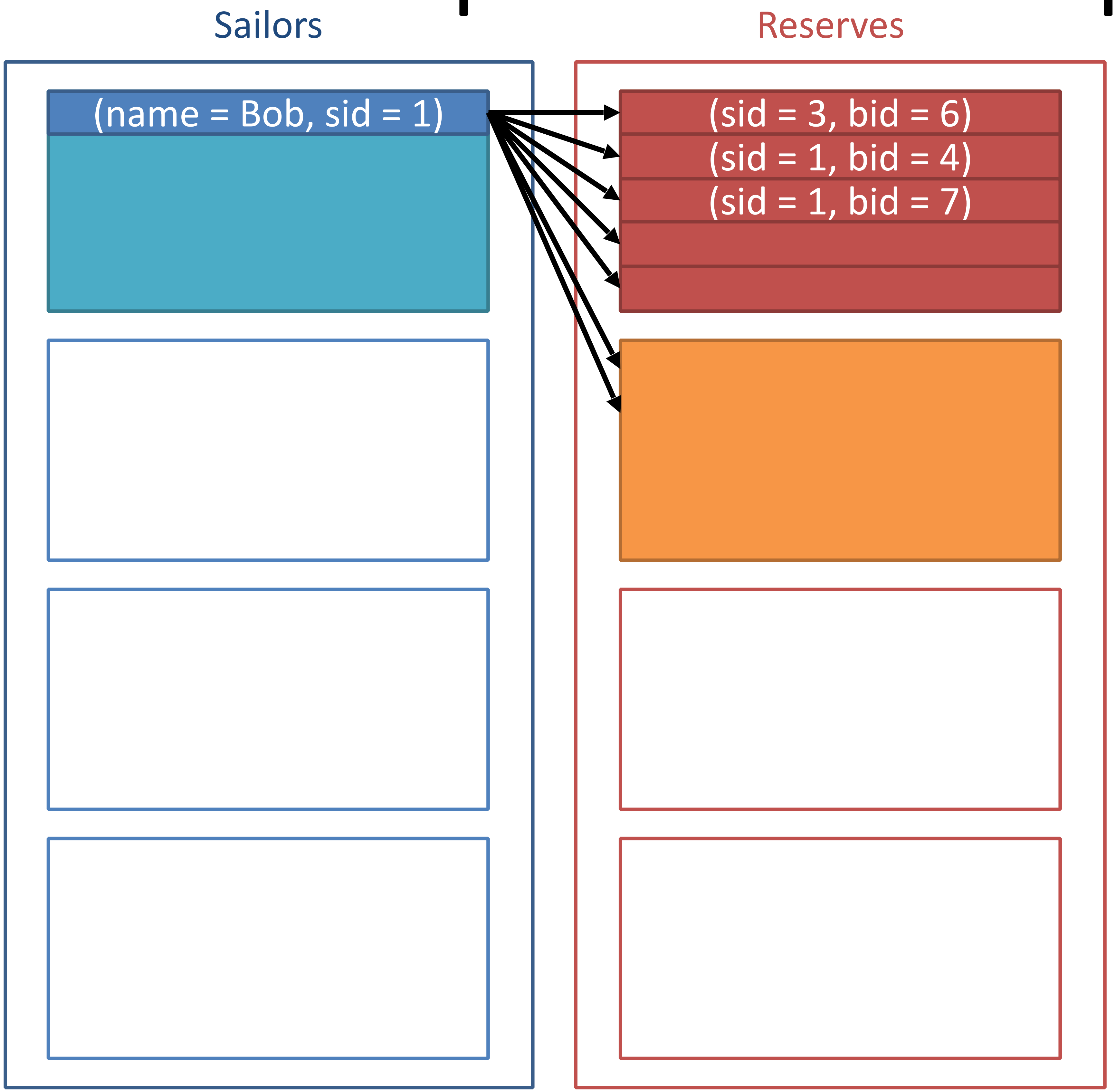
**Steps:**

1. Get tuple of S.
2. Iterate through each tuple in R.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)

# Simple Nested Loops Join



Sailors

(name = Bob, sid = 1)

Reserves

(sid = 3, bid = 6)
(sid = 1, bid = 4)
(sid = 1, bid = 7)

**Key idea:**

Take each record of S and match it with each record of R.

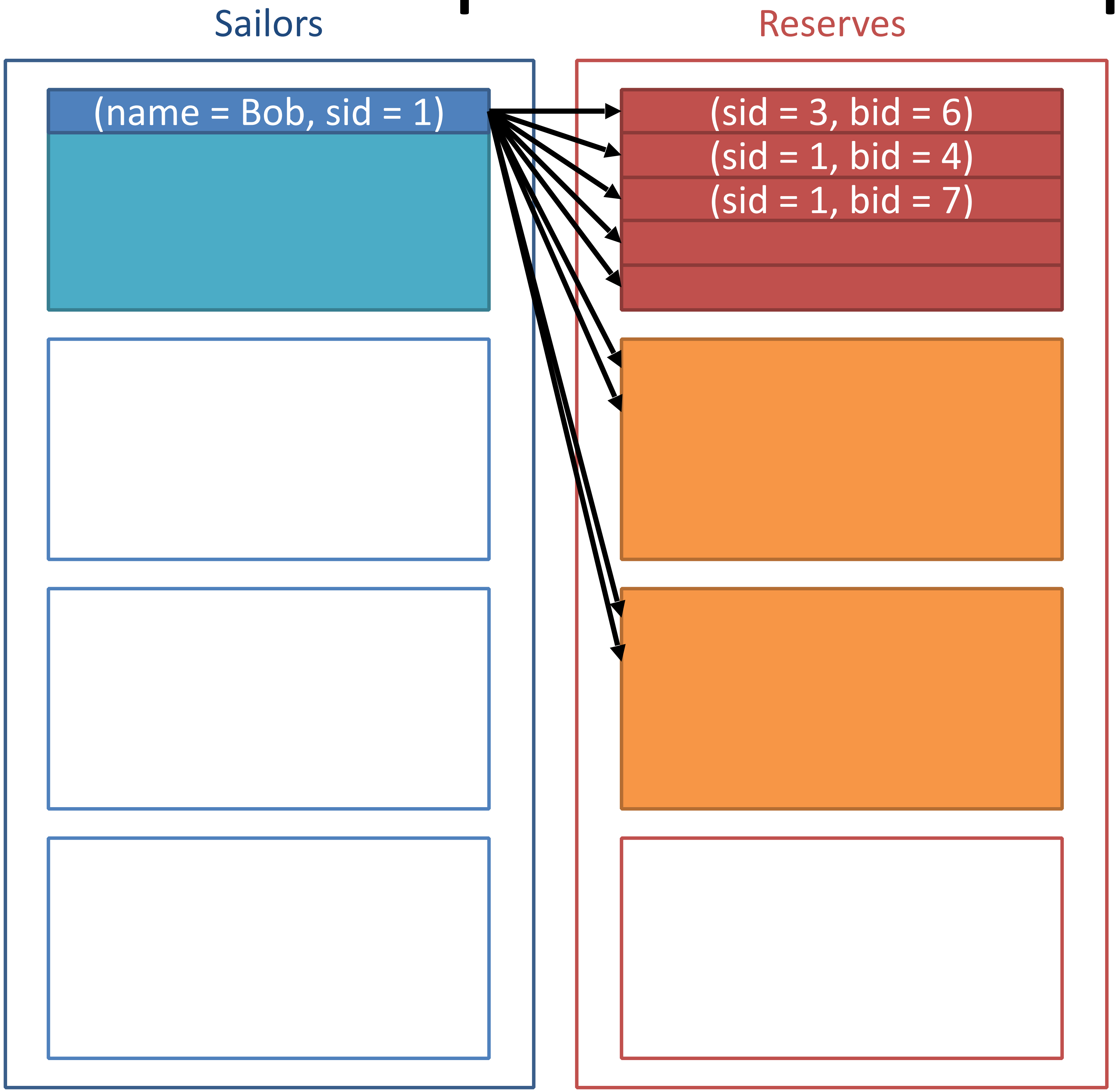
**Steps:**

1. Get tuple of S.
2. Iterate through each tuple in R.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)

# Simple Nested Loops Join



Sailors

(name = Bob, sid = 1)

Reserves

(sid = 3, bid = 6)

(sid = 1, bid = 4)

(sid = 1, bid = 7)

**Key idea:**

Take each record of S and match it with each record of R.

**Steps:**

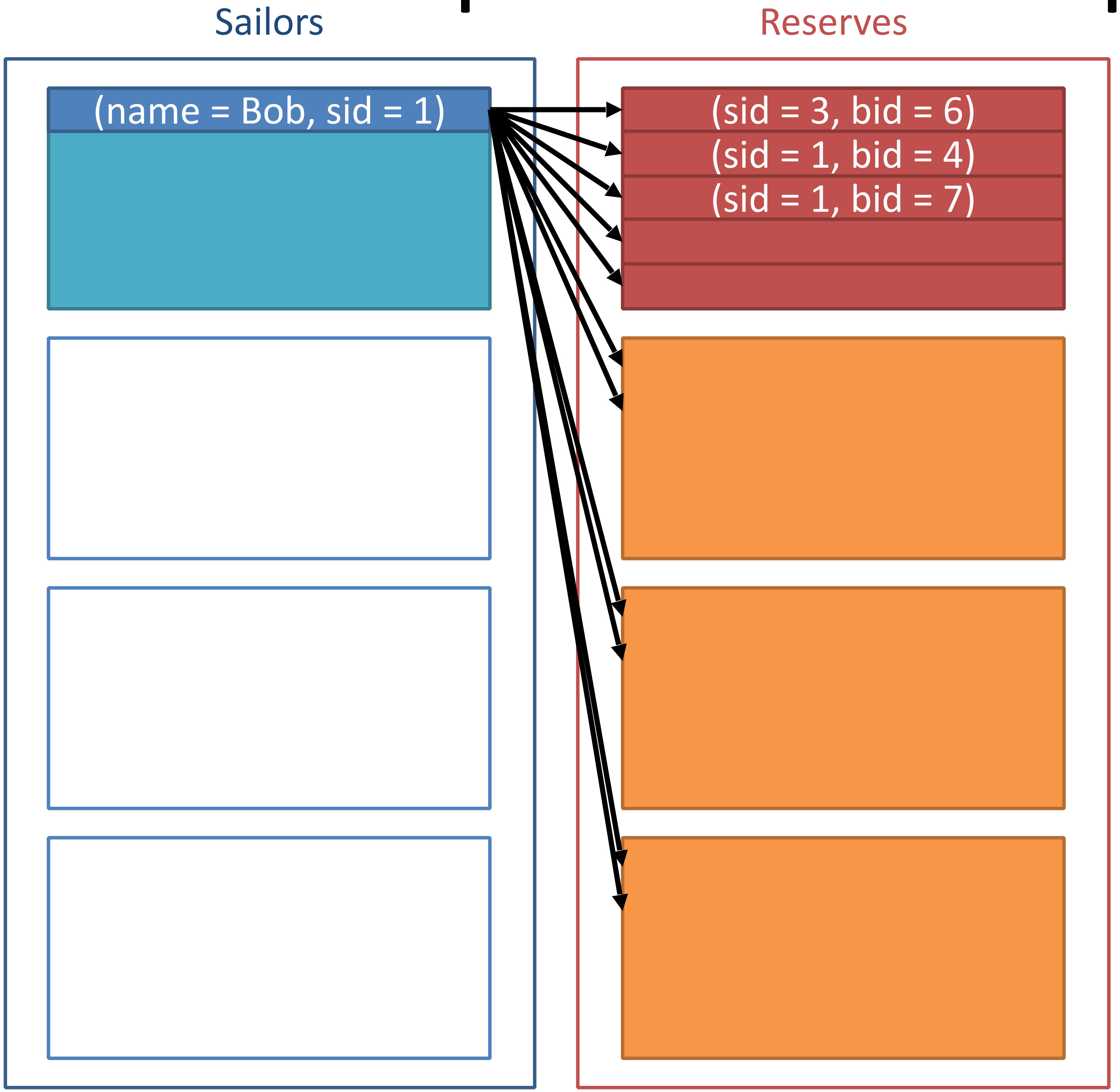
1. Get tuple of S.
2. Iterate through each tuple in R.

**Output:**

(name = Bob, sid = 1, bid = 4)

(name = Bob, sid = 1, bid = 7)

# Simple Nested Loops Join



**Key idea:**

Take each record of S and match it with each record of R.

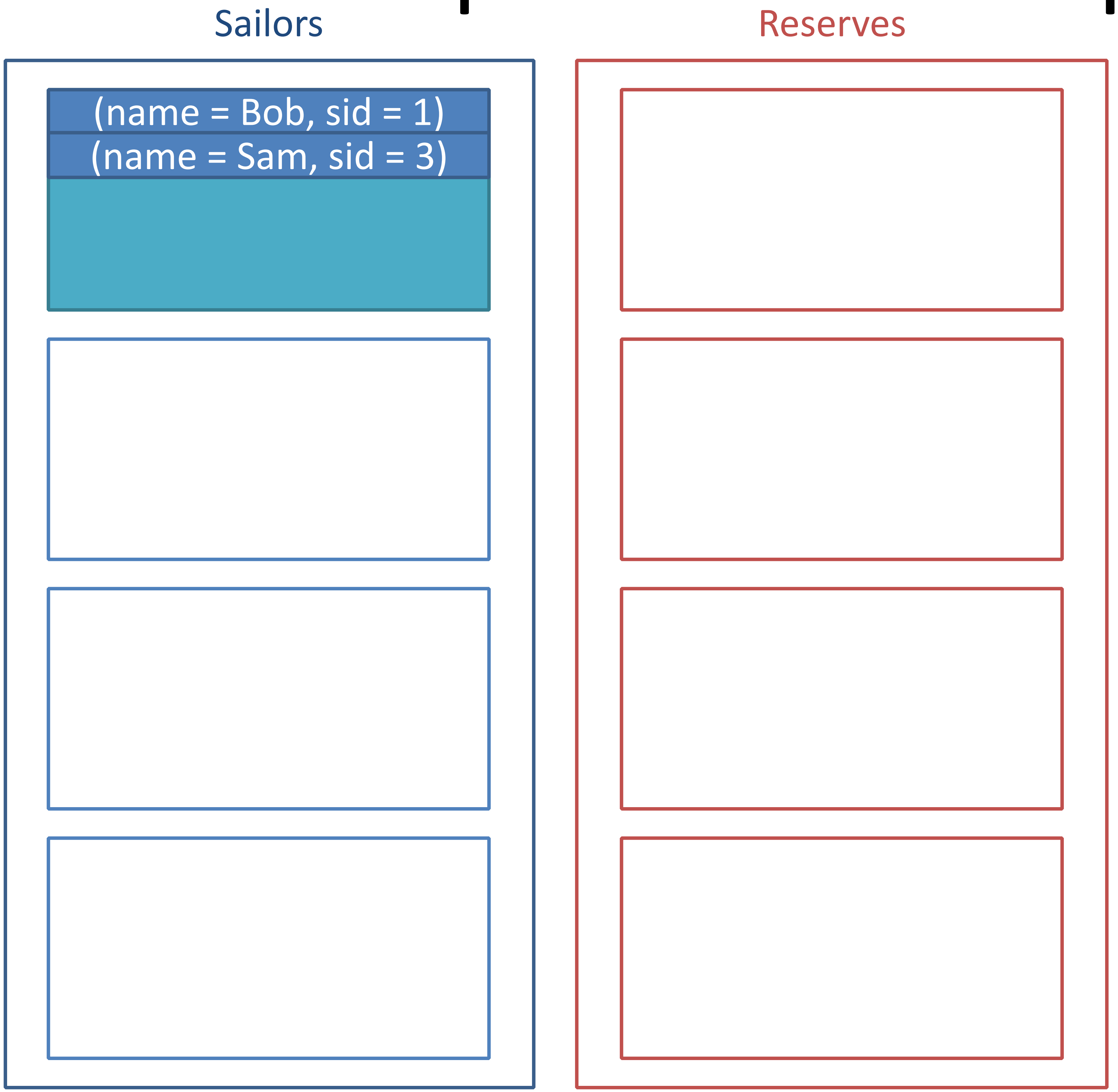
**Steps:**

1. Get tuple of S.
2. Iterate through each tuple in R.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)

# Simple Nested Loops Join



(name = Bob, sid = 1)

(name = Sam, sid = 3)

**Key idea:**

Take each record of S and match it with each record of R.

**Steps:**

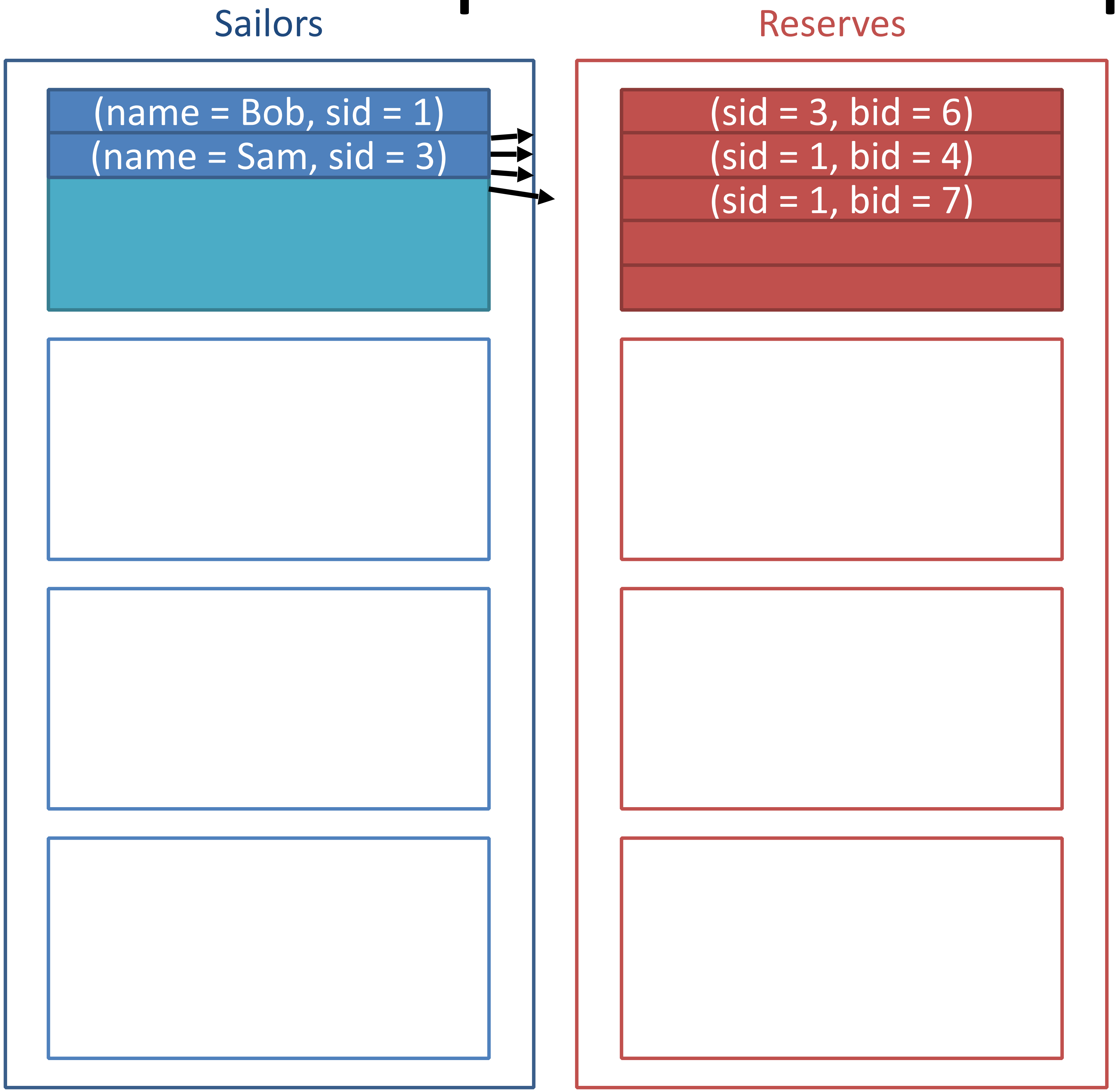
1. Get tuple of S.
2. Iterate through each tuple in R.

**Output:**

(name = Bob, sid = 1, bid = 4)

(name = Bob, sid = 1, bid = 7)

# Simple Nested Loops Join



Sailors

(name = Bob, sid = 1)

(name = Sam, sid = 3)

Reserves

(sid = 3, bid = 6)

(sid = 1, bid = 4)

(sid = 1, bid = 7)

**Key idea:**

Take each record of S and match it with each record of R.

**Steps:**

1. Get tuple of S.
2. Iterate through each tuple in R.

**Output:**

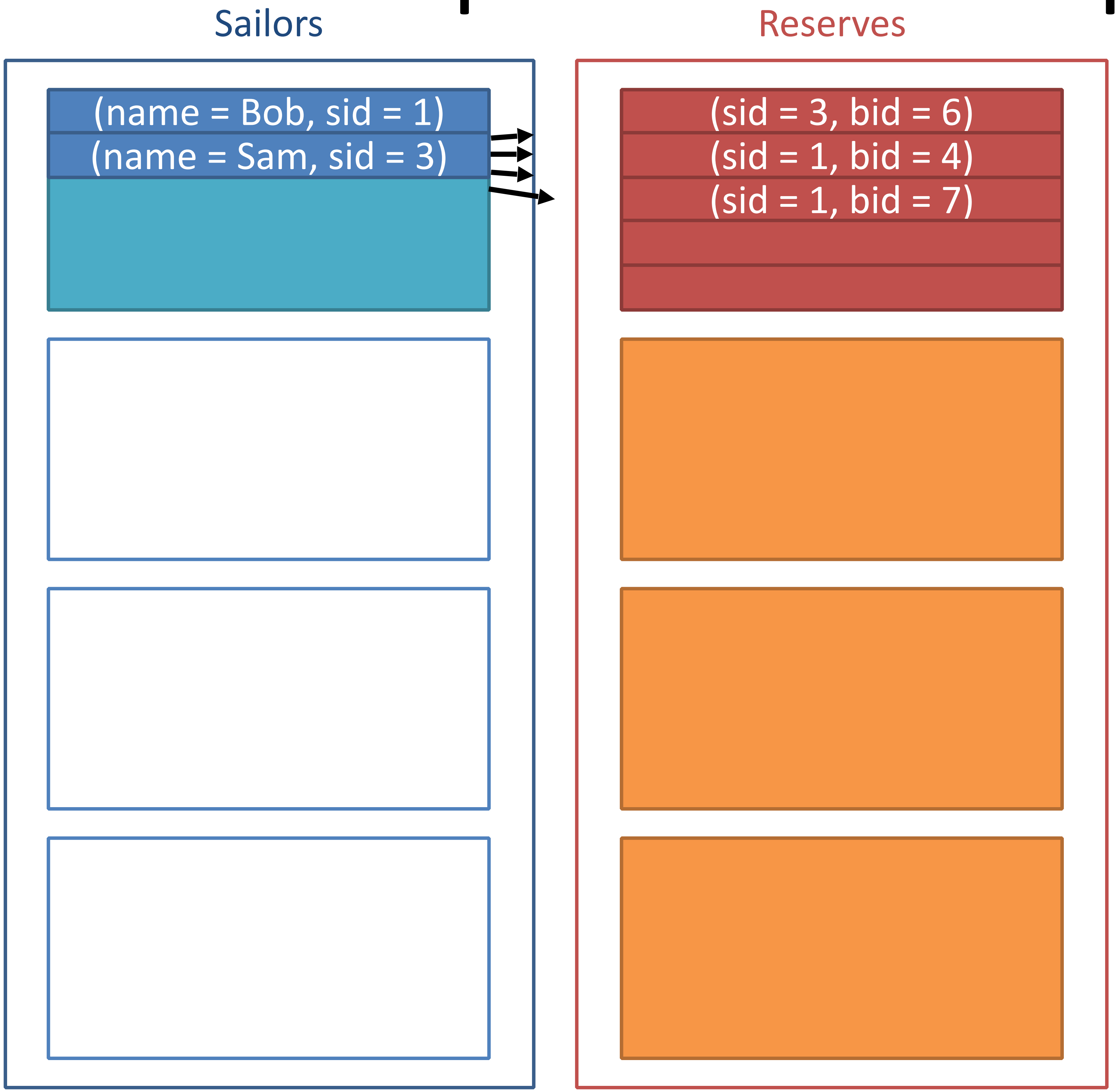
(name = Bob, sid = 1, bid = 4)

(name = Bob, sid = 1, bid = 7)

(name = Sam, sid = 3, bid = 6)

**Notation: [S] == “# pages in S” ; |S| == “# tuples in S”**

# Simple Nested Loops Join



**Key idea:**

Take each record of S and match it with each record of R.

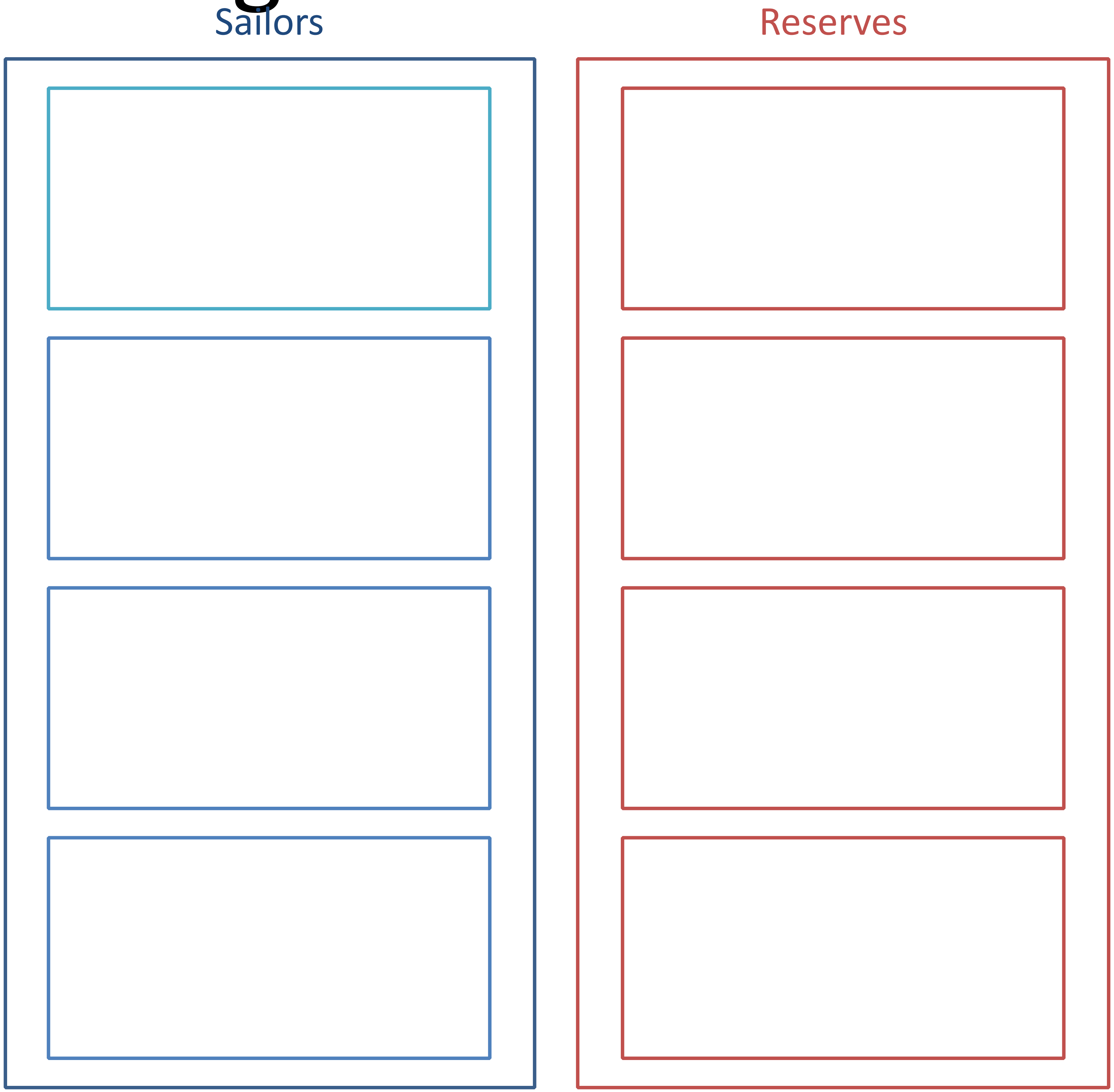
**Steps:**

1. Get tuple of S.
2. Iterate through each tuple in R.

**I/Os:**

[S] + |S|*[R]

# Page-Oriented Nested Loops Join



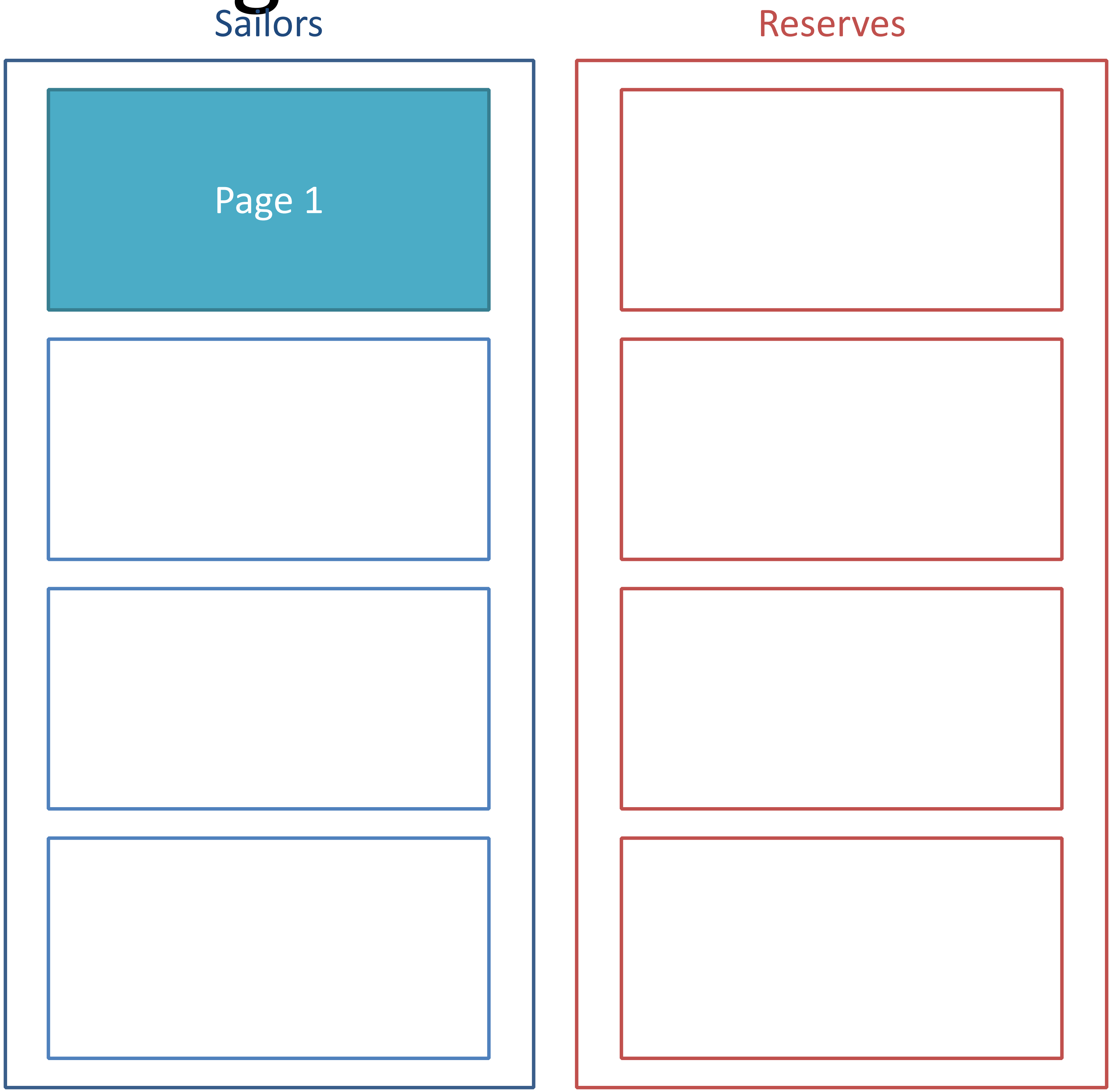
**Key idea:**

Take each page of S and match with each page of R.

**Steps:**

1. Get page of S.
2. Iterate through each page in R.
3. Compare tuples in each.

# Page-Oriented Nested Loops Join



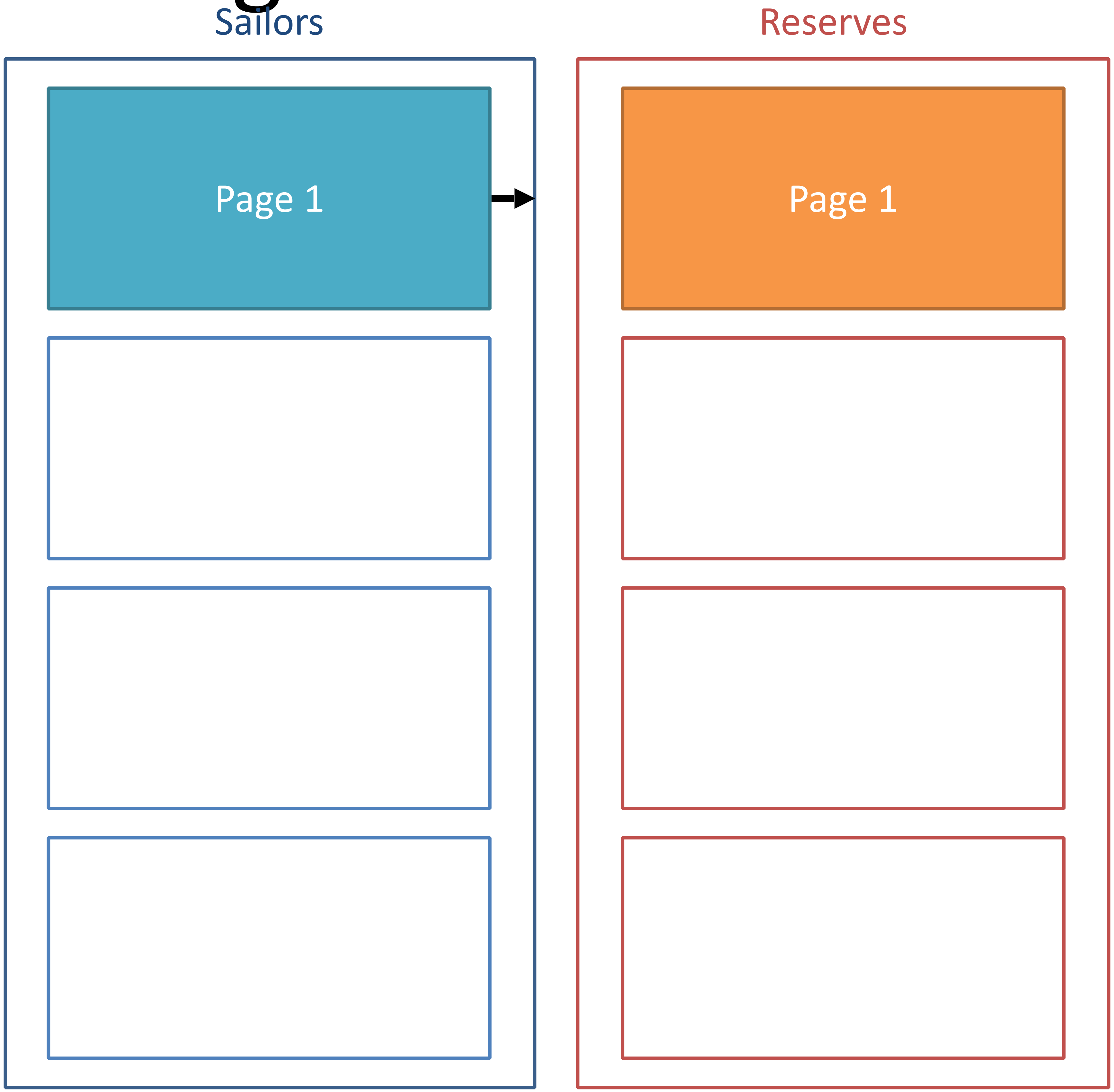
**Key idea:**

Take each page of S and match with each page of R.

**Steps:**

1. Get page of S.
2. Iterate through each page in R.
3. Compare tuples in each.

# Page-Oriented Nested Loops Join



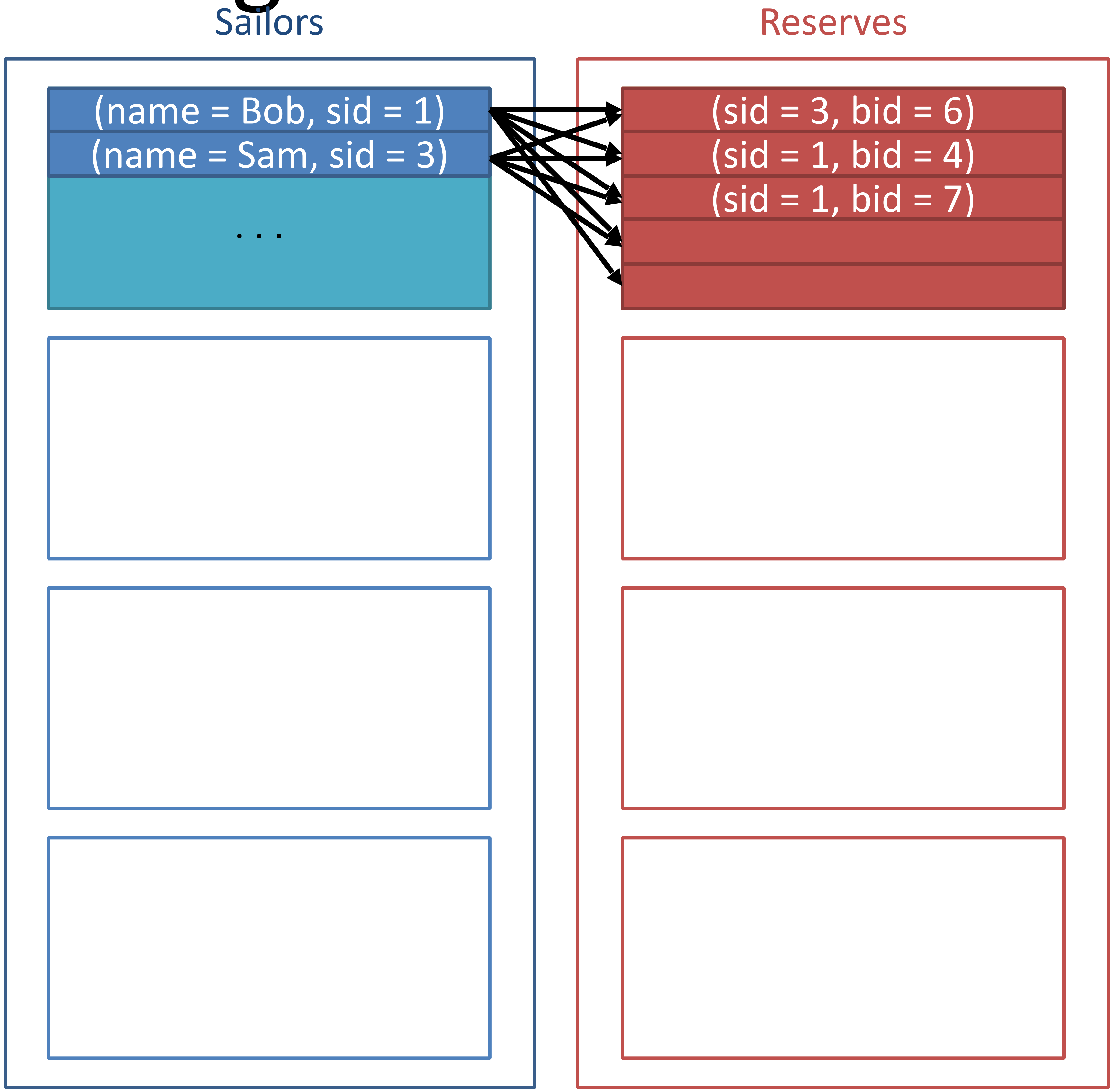
**Key idea:**

Take each page of S and match with each page of R.

**Steps:**

1. Get page of S.
2. Iterate through each page in R.
3. Compare tuples in each.

# Page-Oriented Nested Loops Join



**Key idea:**

Take each page of S and match with each page of R.

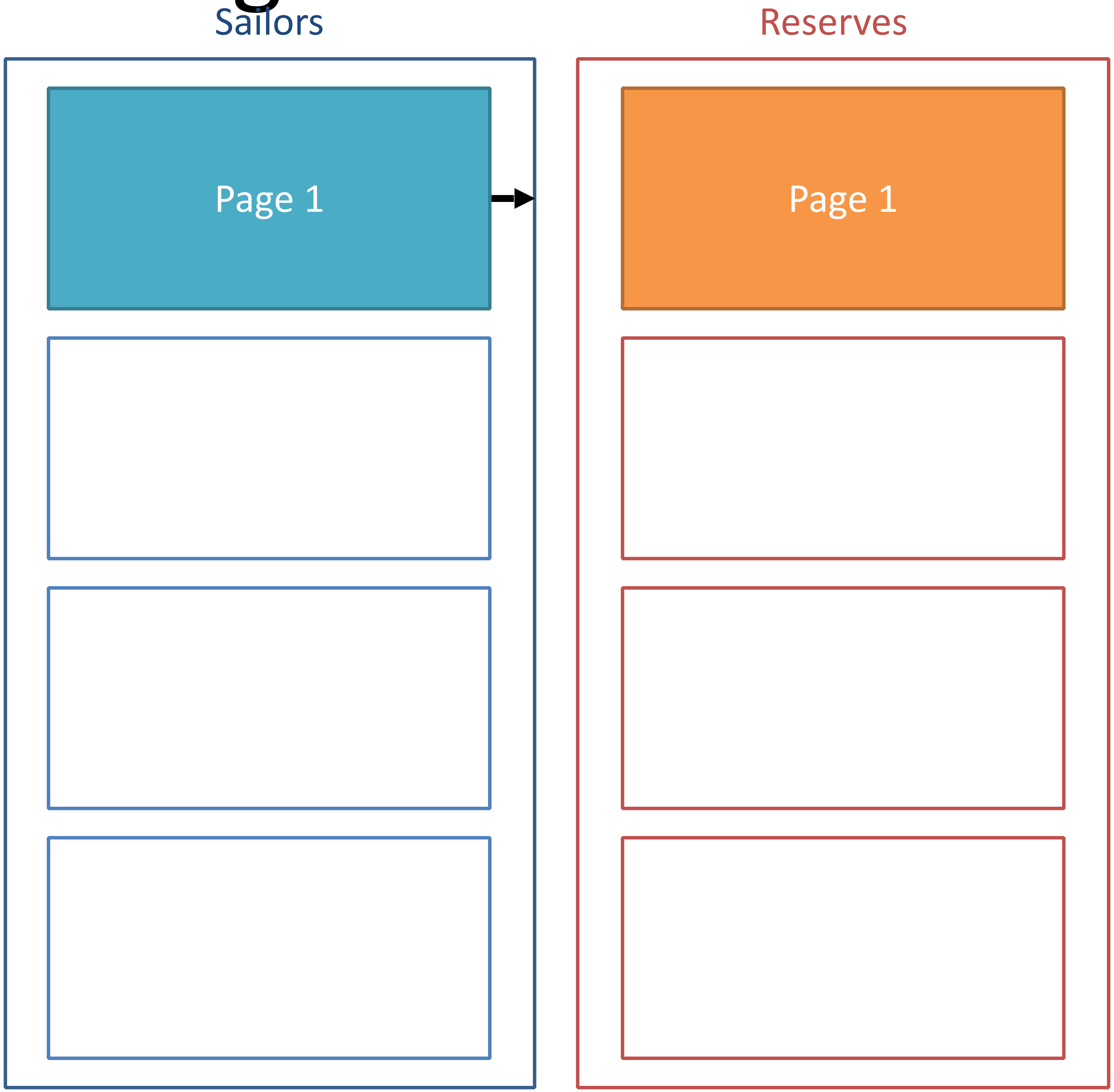
**Steps:**

1. Get page of S.
2. Iterate through each page in R.
3. Compare tuples in each.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)

# Page-Oriented Nested Loops Join



**Key idea:**

Take each page of S and match with each page of R.

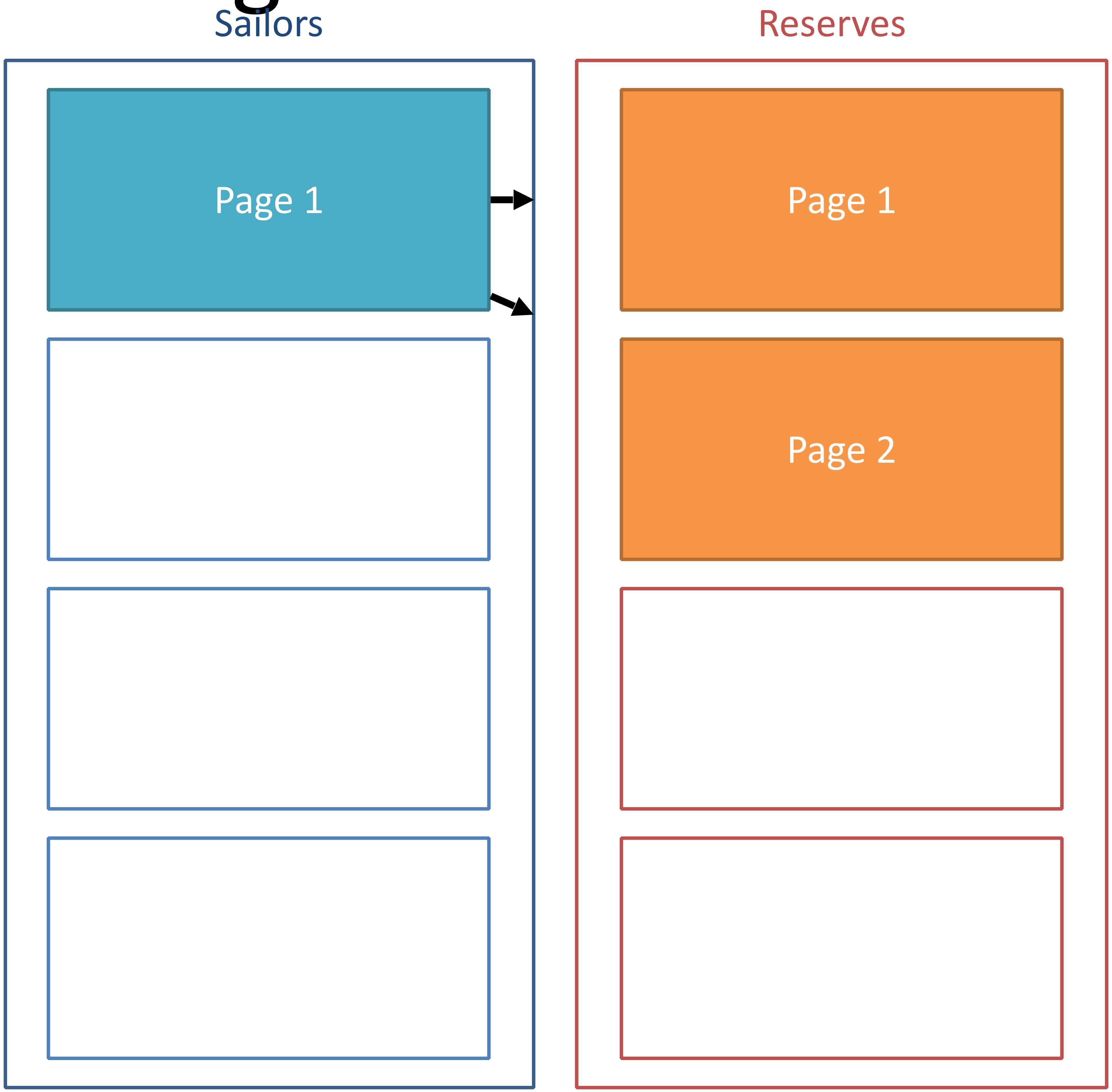
**Steps:**

1. Get page of S.
2. Iterate through each page in R.
3. Compare tuples in each.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)

# Page-Oriented Nested Loops Join



**Key idea:**

Take each page of S and match with each page of R.

**Steps:**

1. Get page of S.
2. Iterate through each page in R.
3. Compare tuples in each.

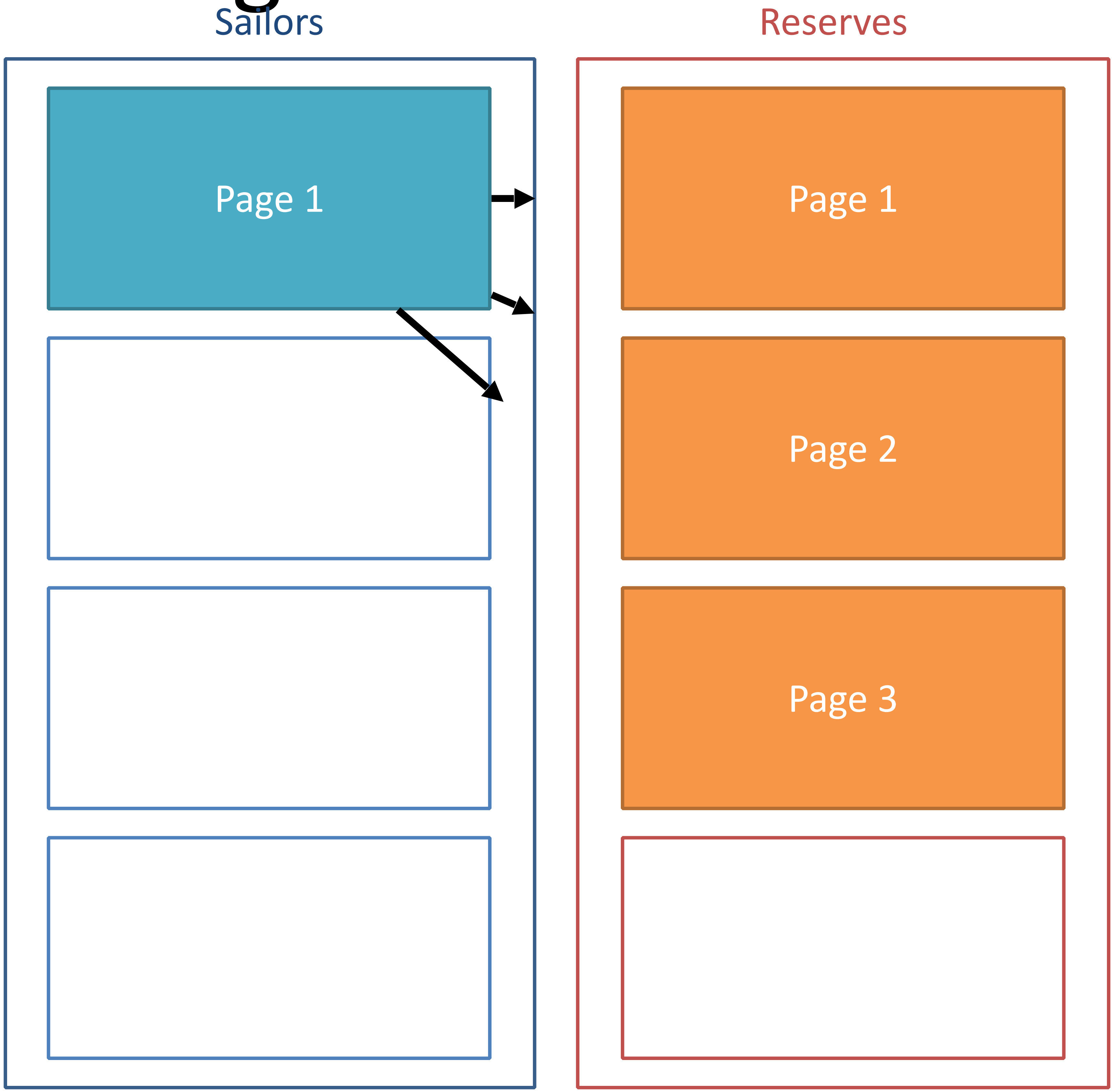
**Output:**

(name = Bob, sid = 1, bid = 4)

(name = Bob, sid = 1, bid = 7)

(name = Sam, sid = 3, bid = 6)

# Page-Oriented Nested Loops Join



**Key idea:**

Take each page of S and match with each page of R.

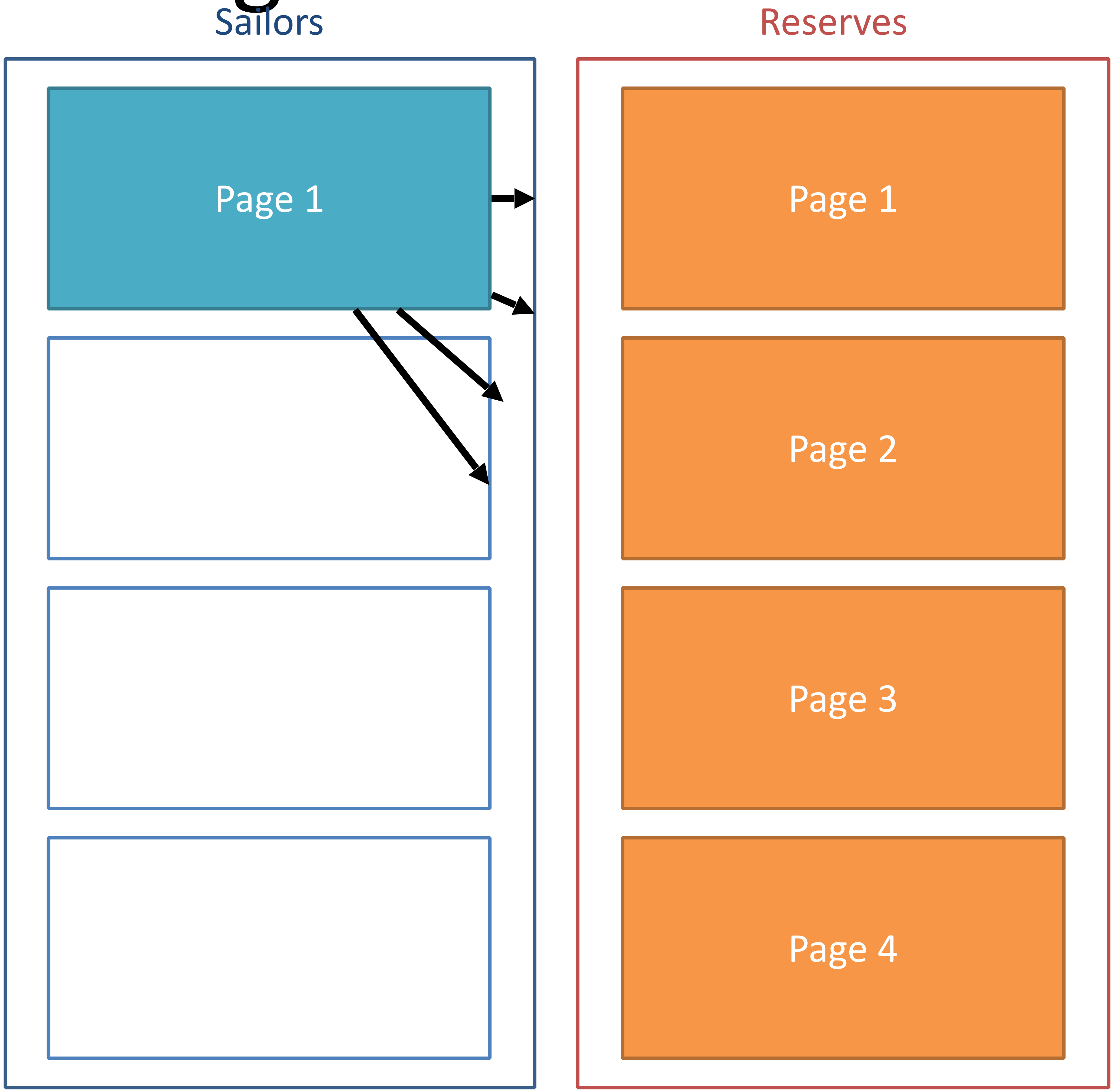
**Steps:**

1. Get page of S.
2. Iterate through each page in R.
3. Compare tuples in each.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)

# Page-Oriented Nested Loops Join



**Key idea:**

Take each page of S and match with each page of R.

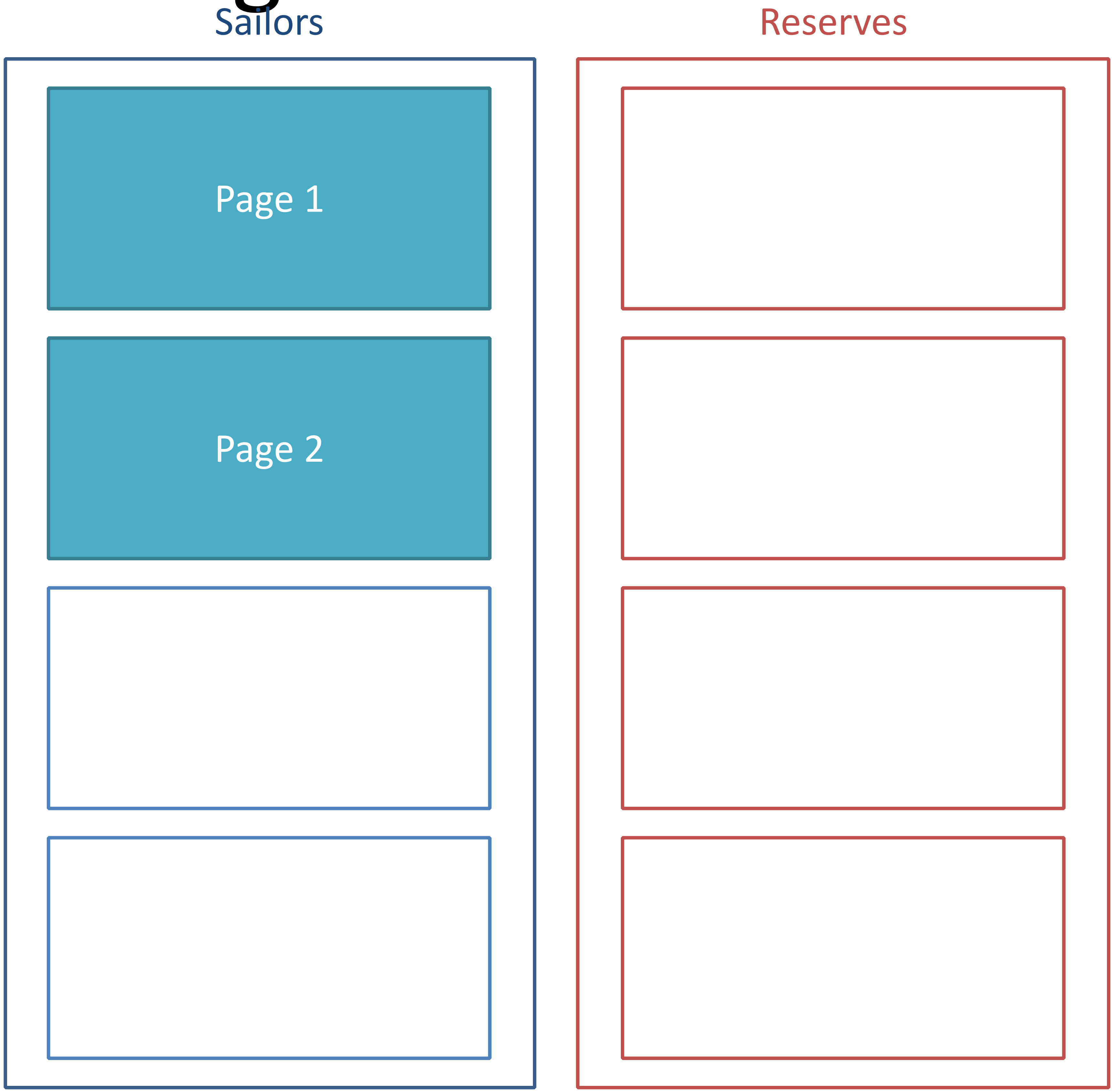
**Steps:**

1. Get page of S.
2. Iterate through each page in R.
3. Compare tuples in each.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)

# Page-Oriented Nested Loops Join



**Key idea:**

Take each page of S and match with each page of R.

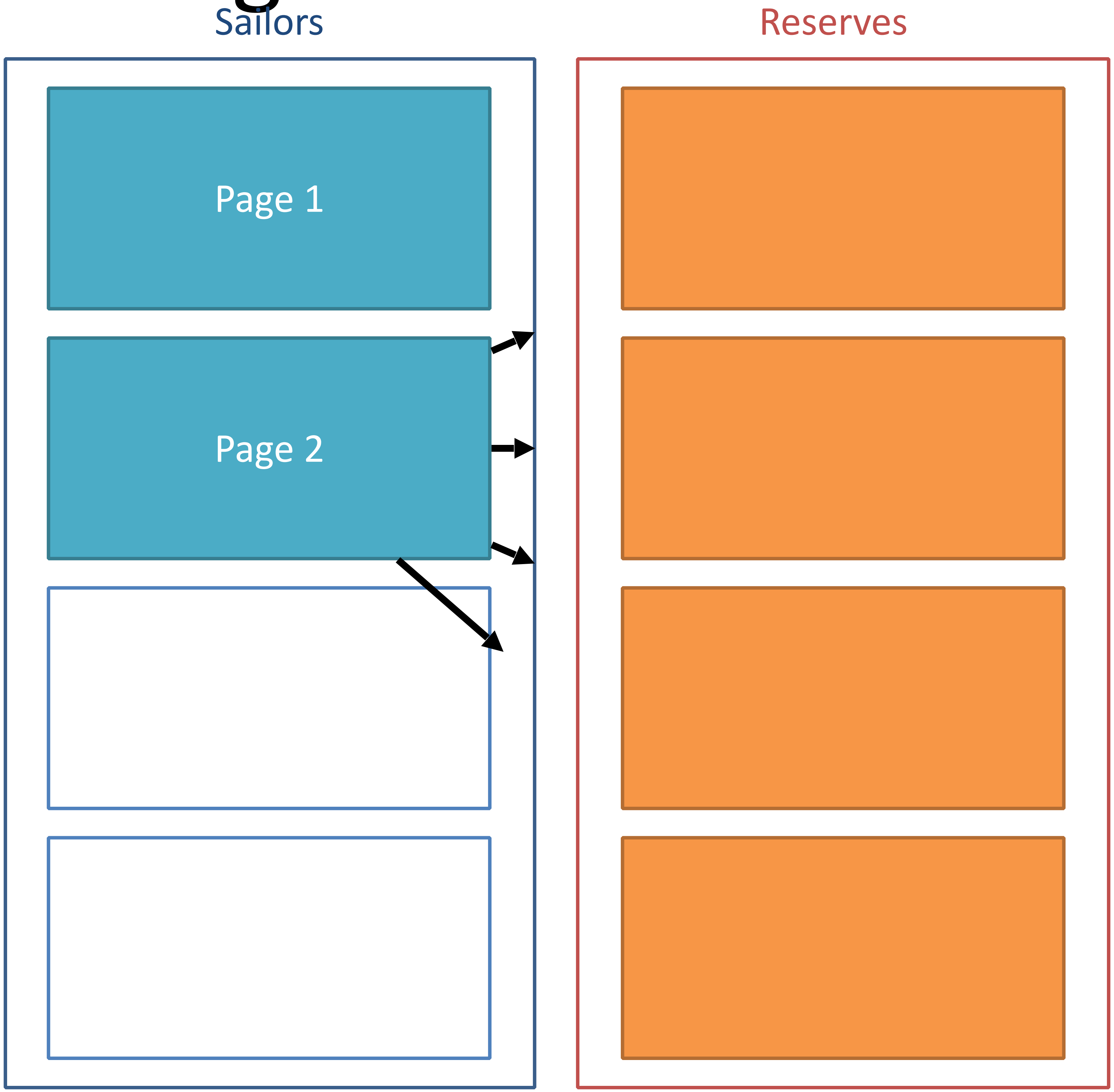
**Steps:**

1. Get page of S.
2. Iterate through each page in R.
3. Compare tuples in each.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)

# Page-Oriented Nested Loops Join



**Key idea:**

Take each page of S and match with each page of R.

**Steps:**

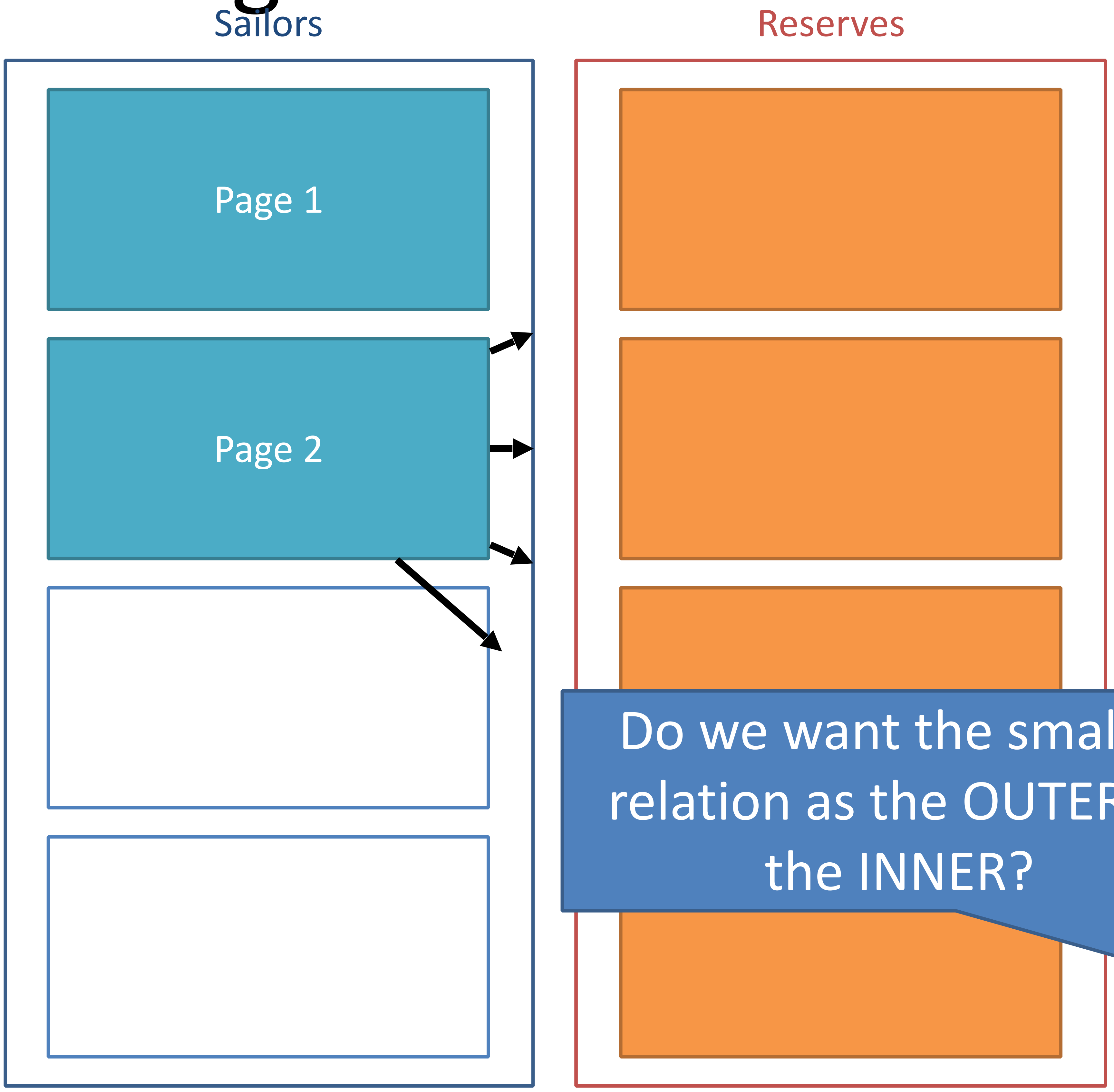
1. Get page of S.
2. Iterate through each page in R.
3. Compare tuples in each.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)

**Notation: [S] == “# pages in S” ; |S| == “# tuples in S”**

# Page-Oriented Nested Loops Join



Do we want the smaller relation as the OUTER or the INNER?

**Key idea:**

Take each page of S and match with each page of R.

**Steps:**

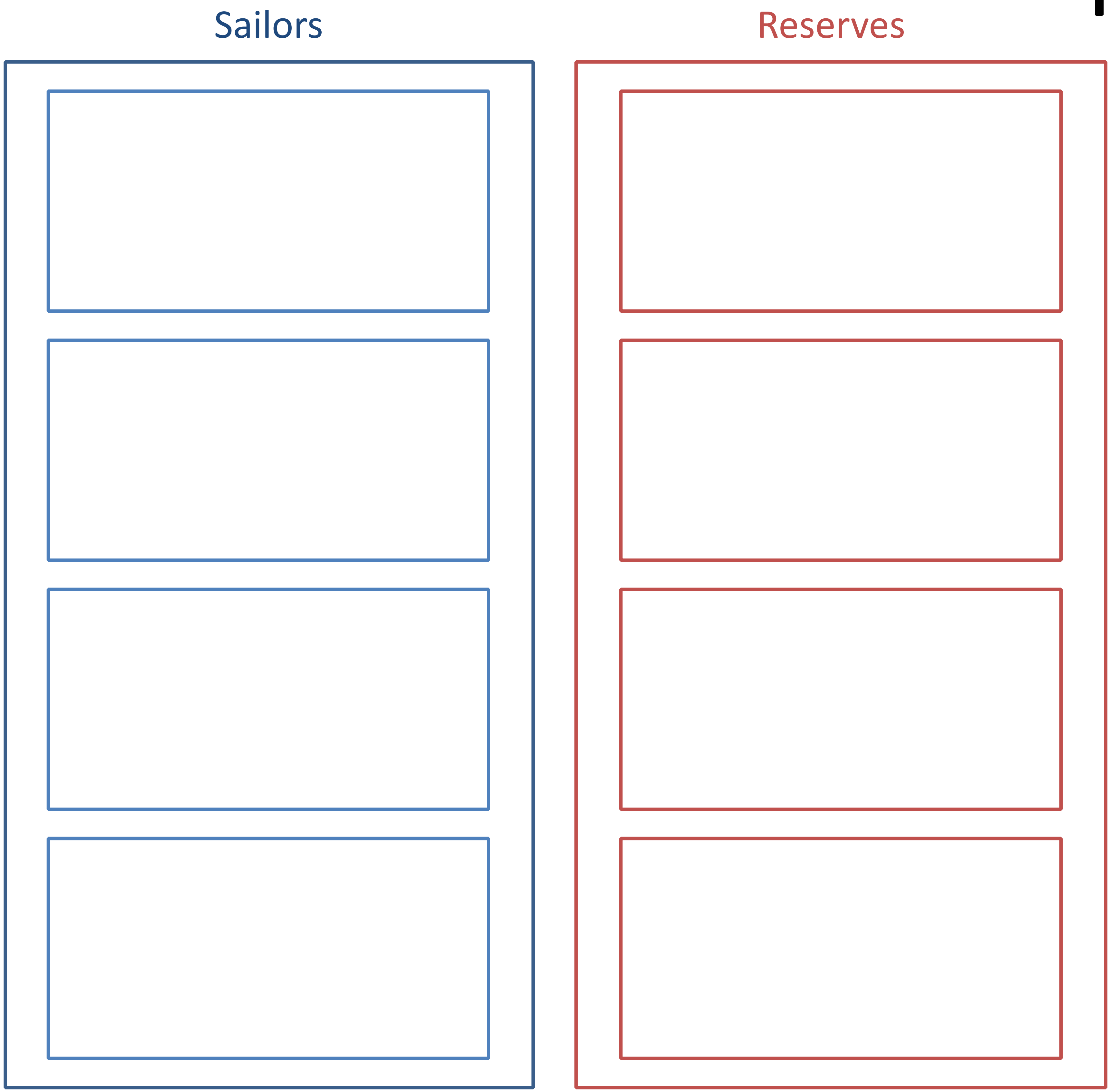
1. Get page of S.
2. Iterate through each page in R.

ompare tuples in ach.

:

[S] + [S]*[R]

# Chunk Nested Loops Join



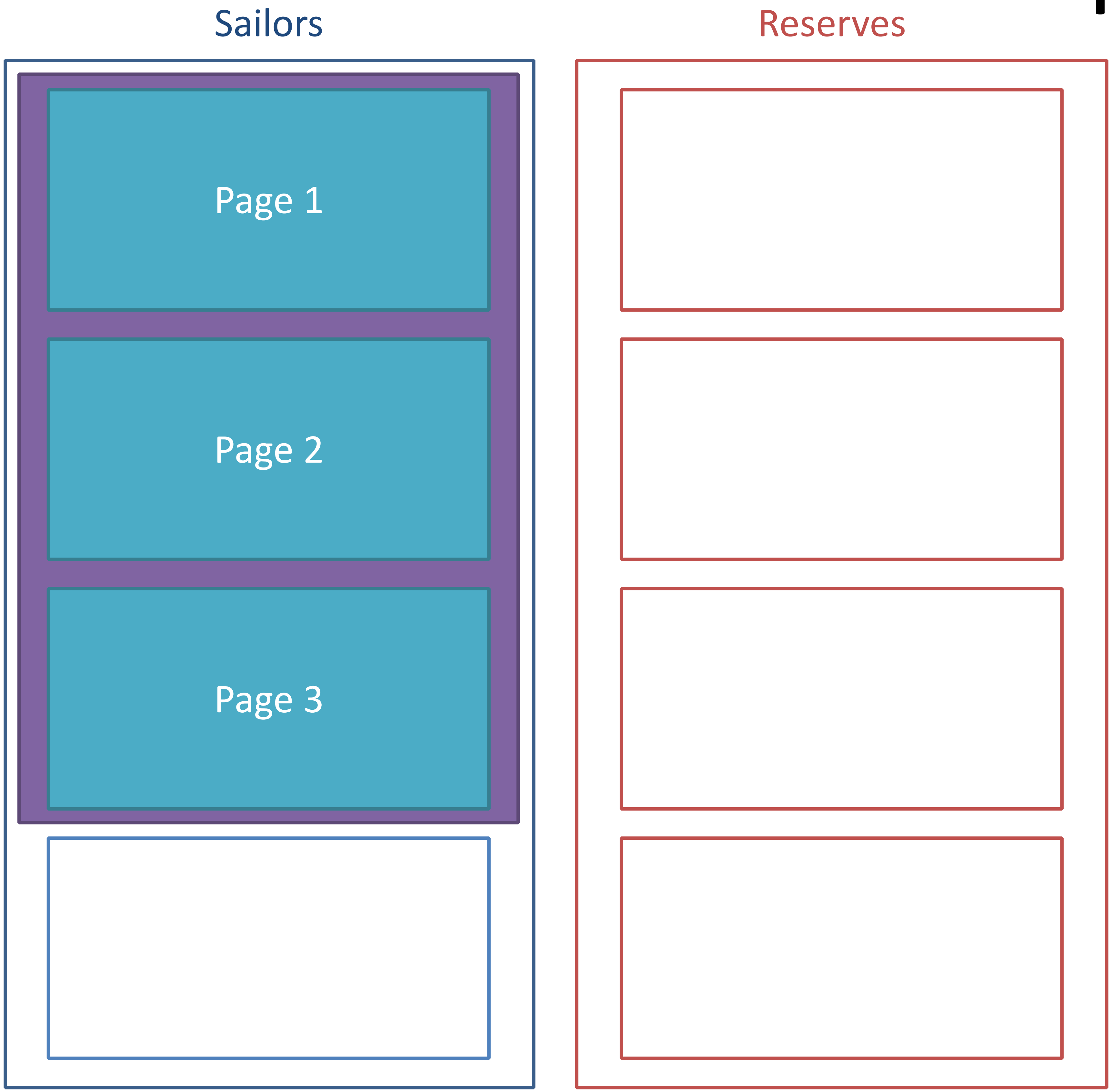
**Key idea:**

Take **k pages** of S and match with each page of R.

**Steps:**

1. Get **k** pages of S.
2. Iterate through each page in R.
3. Compare tuples in each.

# Chunk Nested Loops Join



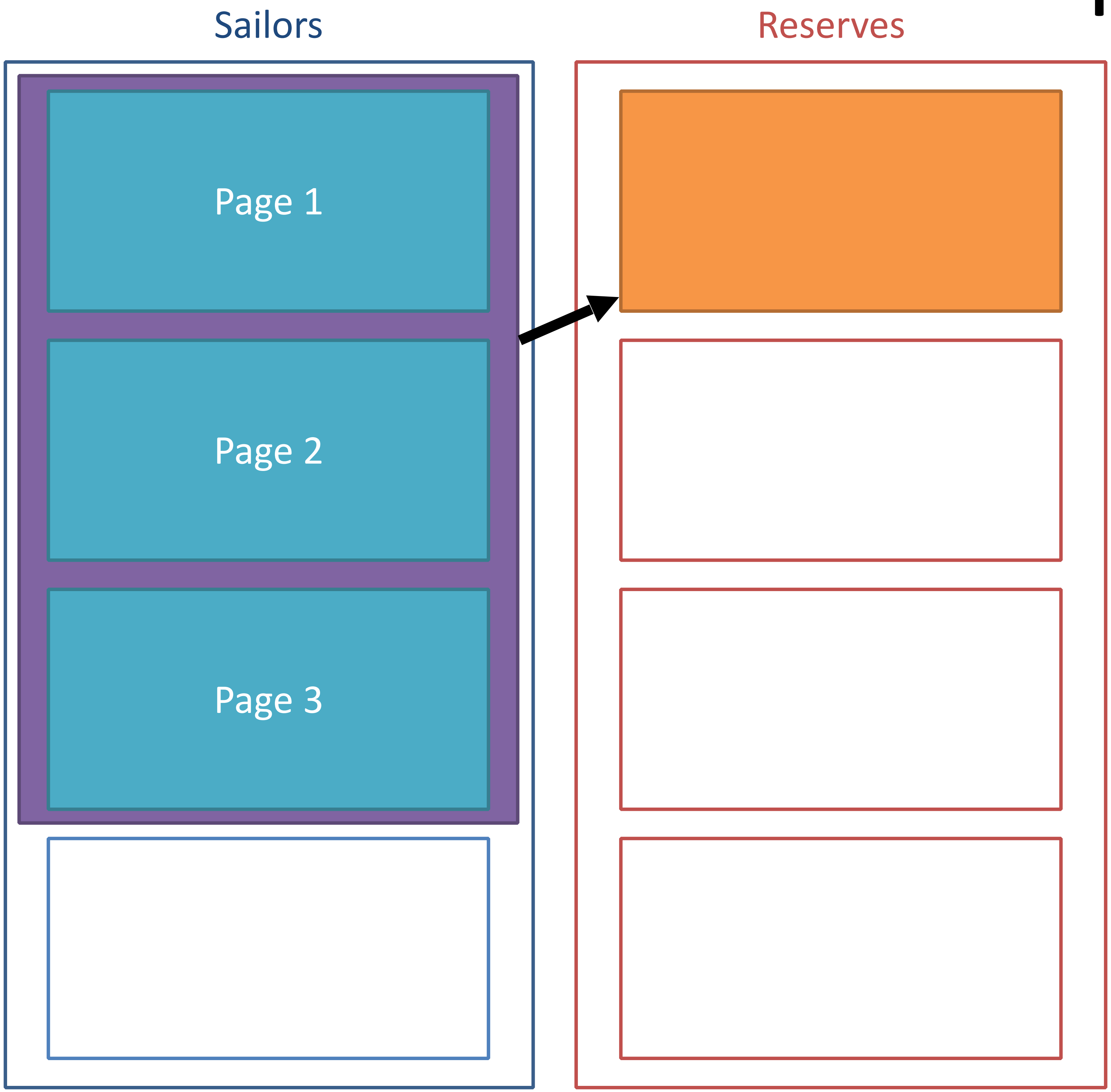
**Key idea:**

Take **k pages** of S and match with each page of R.

**Steps:**

1. Get **k** pages of S.
2. Iterate through each page in R.
3. Compare tuples in each.

# Chunk Nested Loops Join



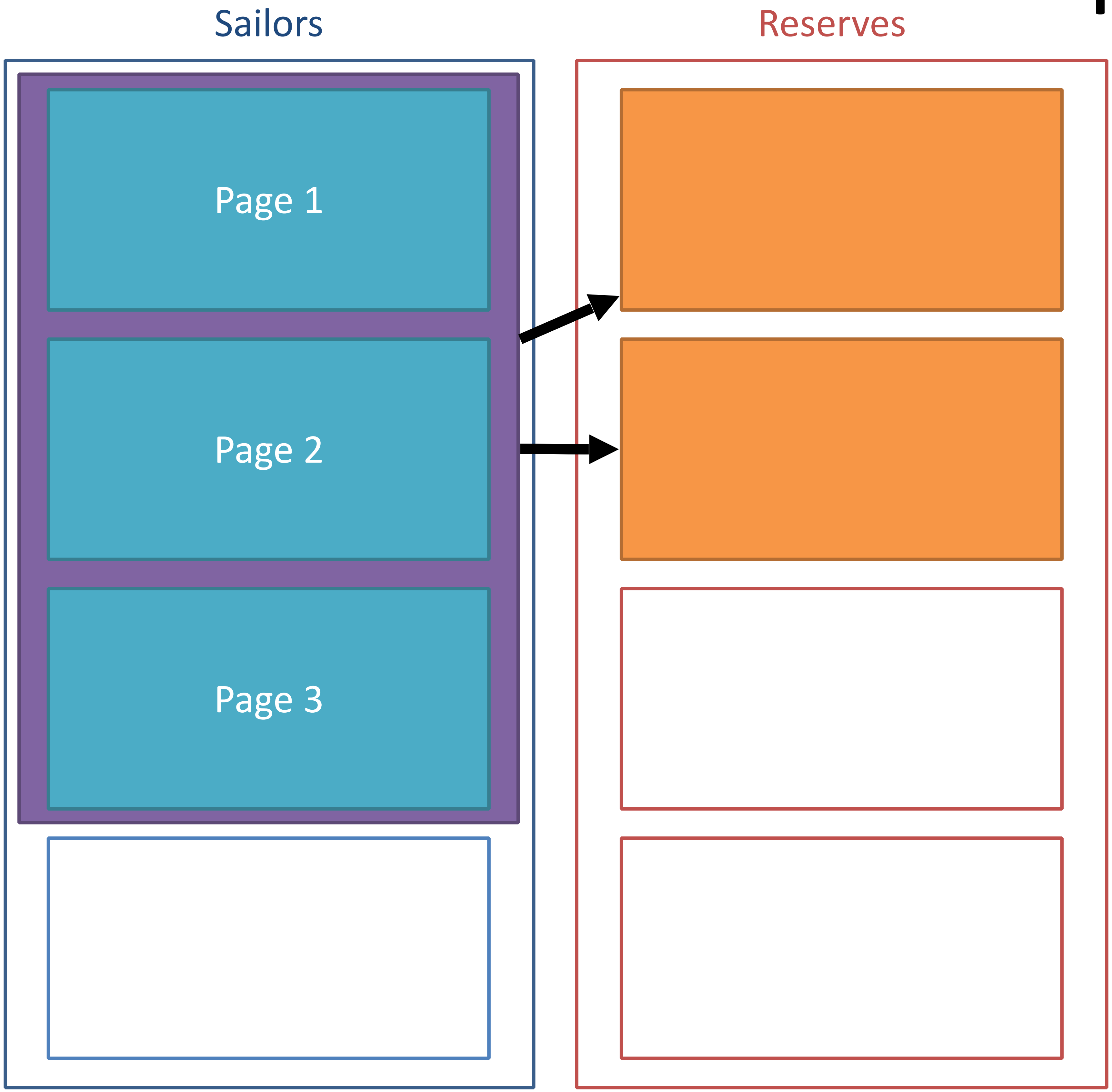
**Key idea:**

Take **k pages** of S and match with each page of R.

**Steps:**

1. Get **k** pages of S.
2. Iterate through each page in R.
3. Compare tuples in each.

# Chunk Nested Loops Join



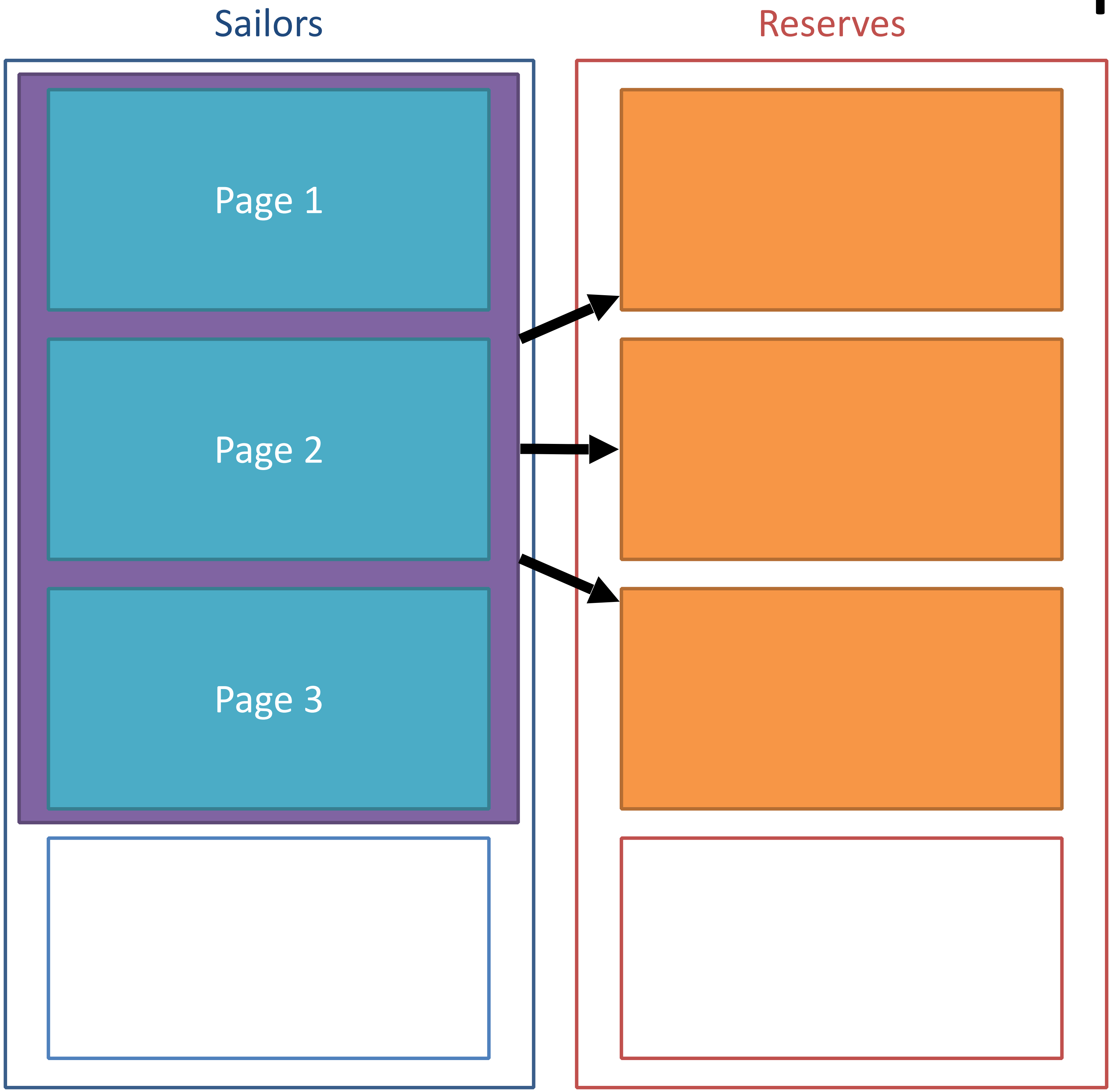
**Key idea:**

Take **k pages** of S and match with each page of R.

**Steps:**

1. Get **k** pages of S.
2. Iterate through each page in R.
3. Compare tuples in each.

# Chunk Nested Loops Join



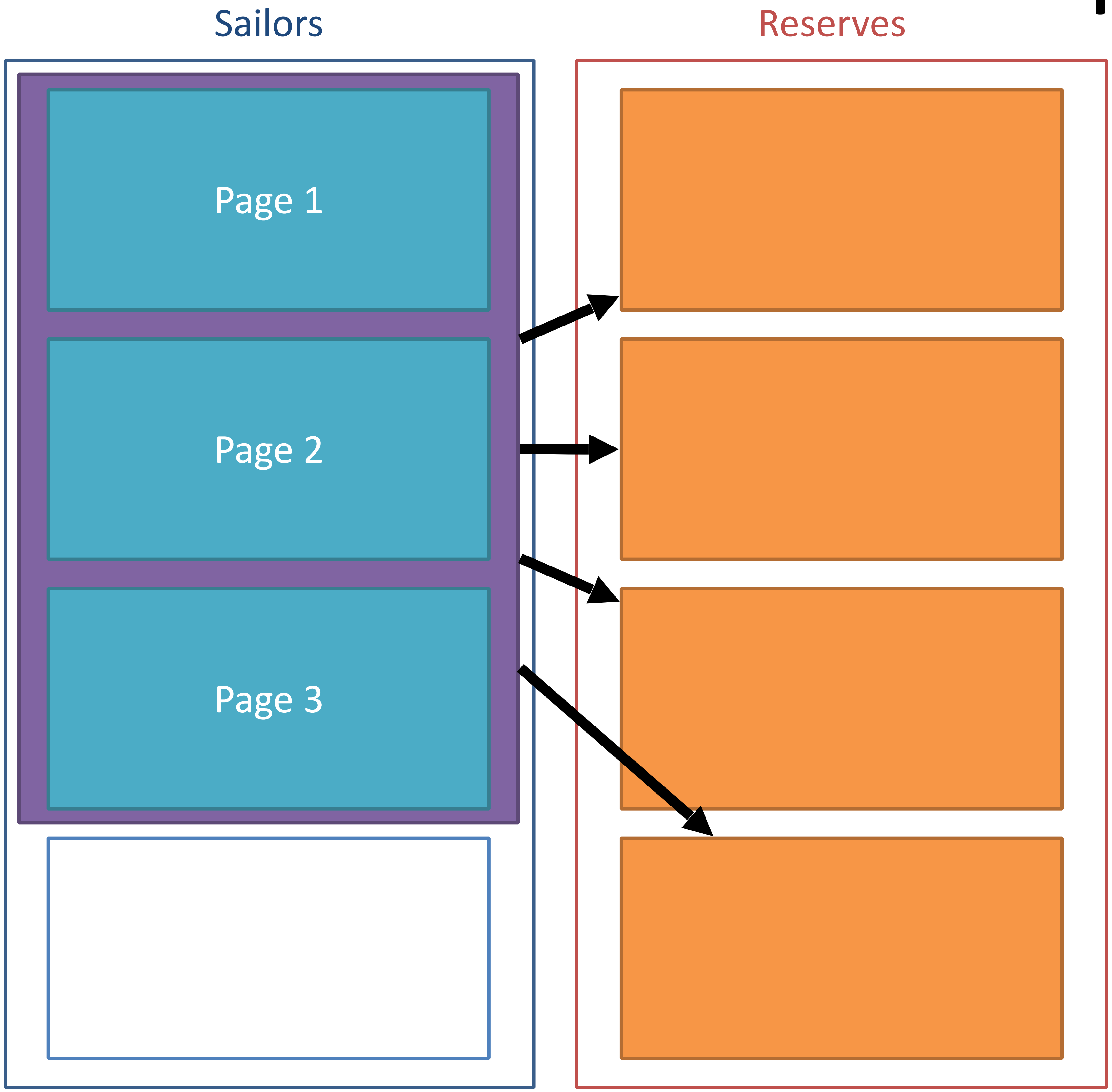
**Key idea:**

Take **k pages** of S and match with each page of R.

**Steps:**

1. Get **k** pages of S.
2. Iterate through each page in R.
3. Compare tuples in each.

# Chunk Nested Loops Join



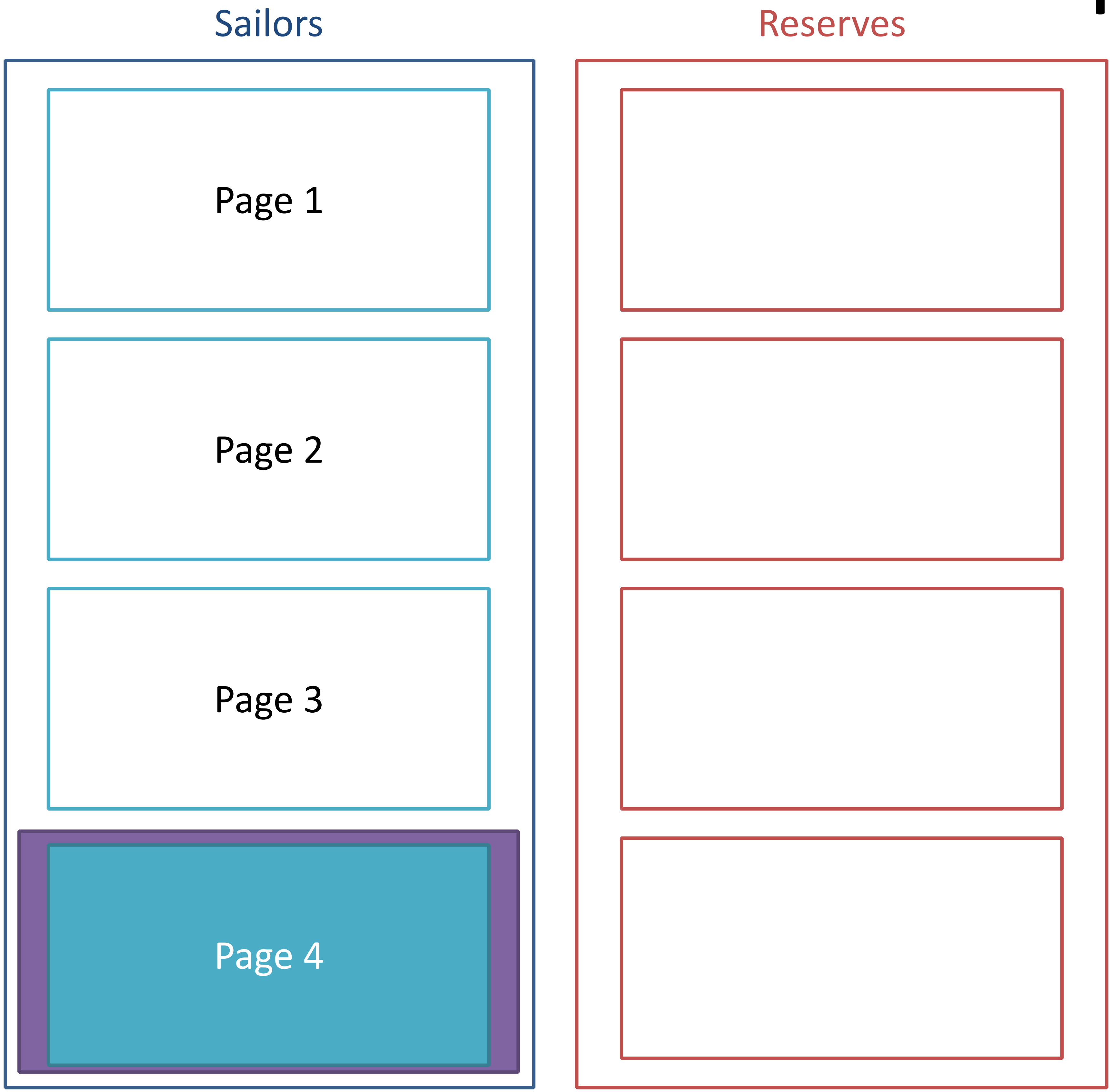
**Key idea:**

Take **k pages** of S and match with each page of R.

**Steps:**

1. Get **k** pages of S.
2. Iterate through each page in R.
3. Compare tuples in each.

# Chunk Nested Loops Join



**Key idea:**

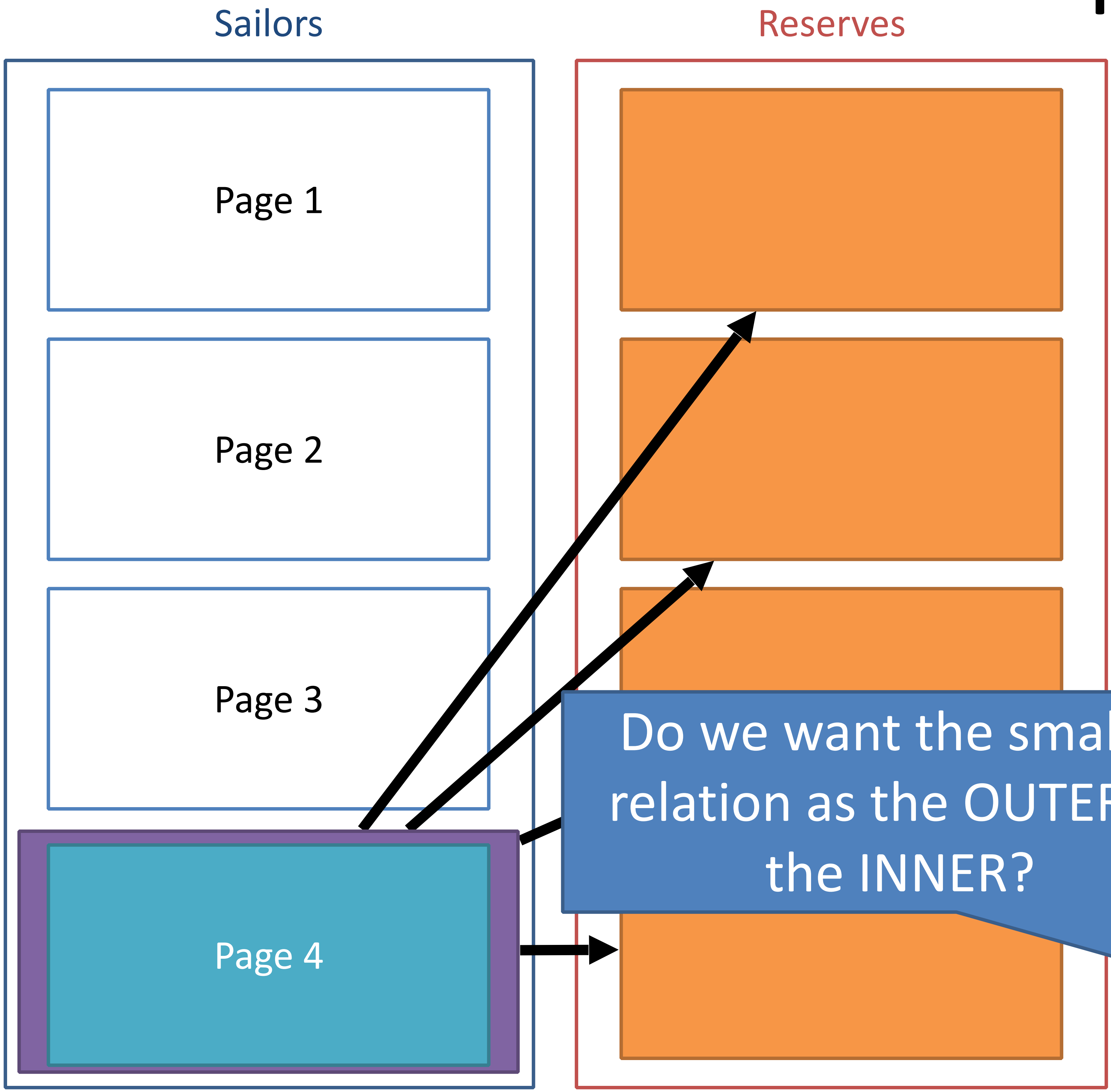
Take **k pages** of S and match with each page of R.

**Steps:**

1. Get **k** pages of S.
2. Iterate through each page in R.
3. Compare tuples in each.

Notation: [S] == “# pages in S” ; |S| == “# tuples in S”

# Chunk Nested Loops Join



Do we want the smaller relation as the OUTER or the INNER?

**Key idea:**

Take **k pages** of S and match with each page of R.

**Steps:**

1. Get **k** pages of S.
2. Iterate through each page in R.

ompare tuples in ach.

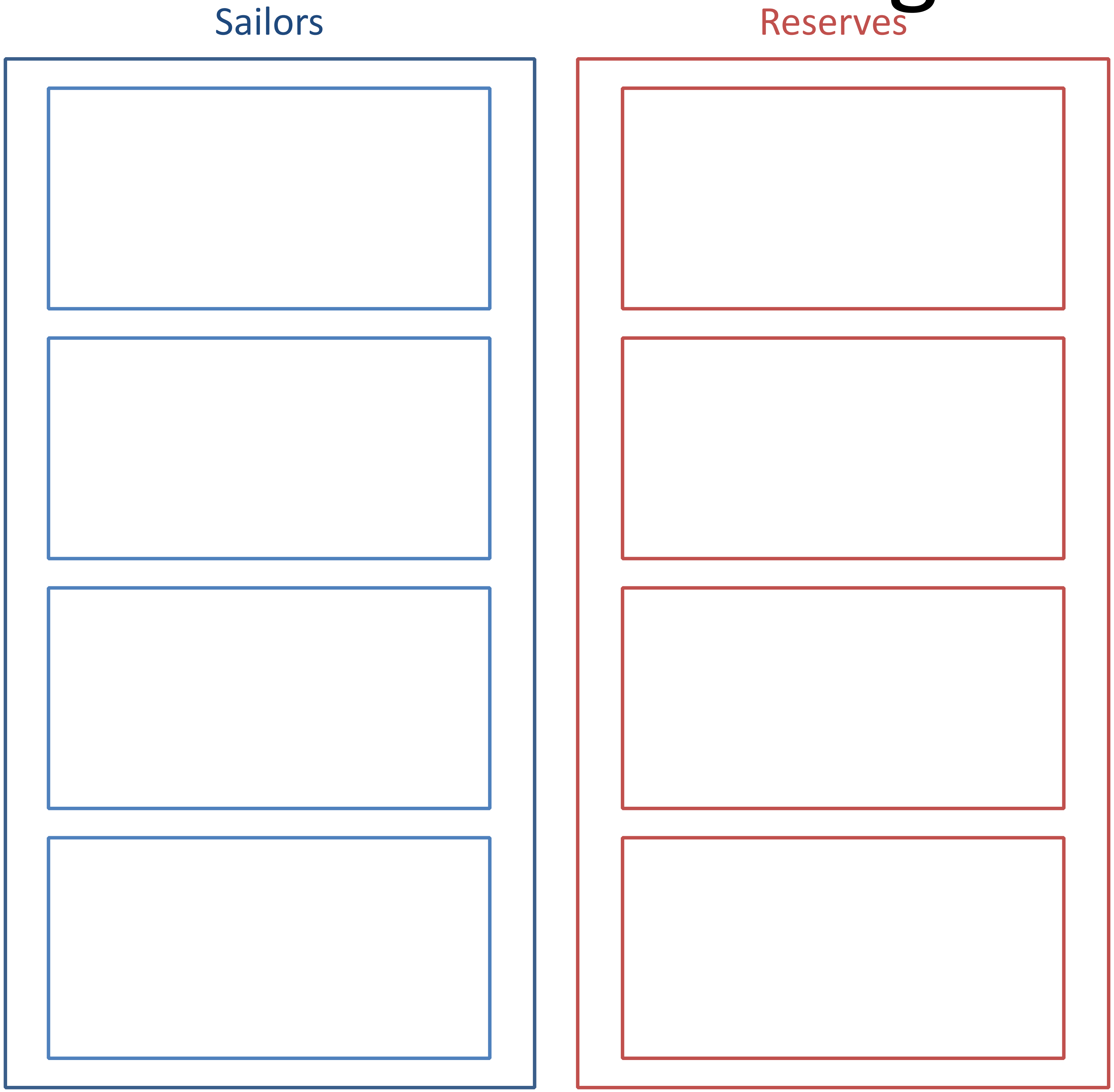
:

[S] + ([S] / k)*[R]

Do question #2 (a & b)

IMPORTANT

# Sort-Merge Join



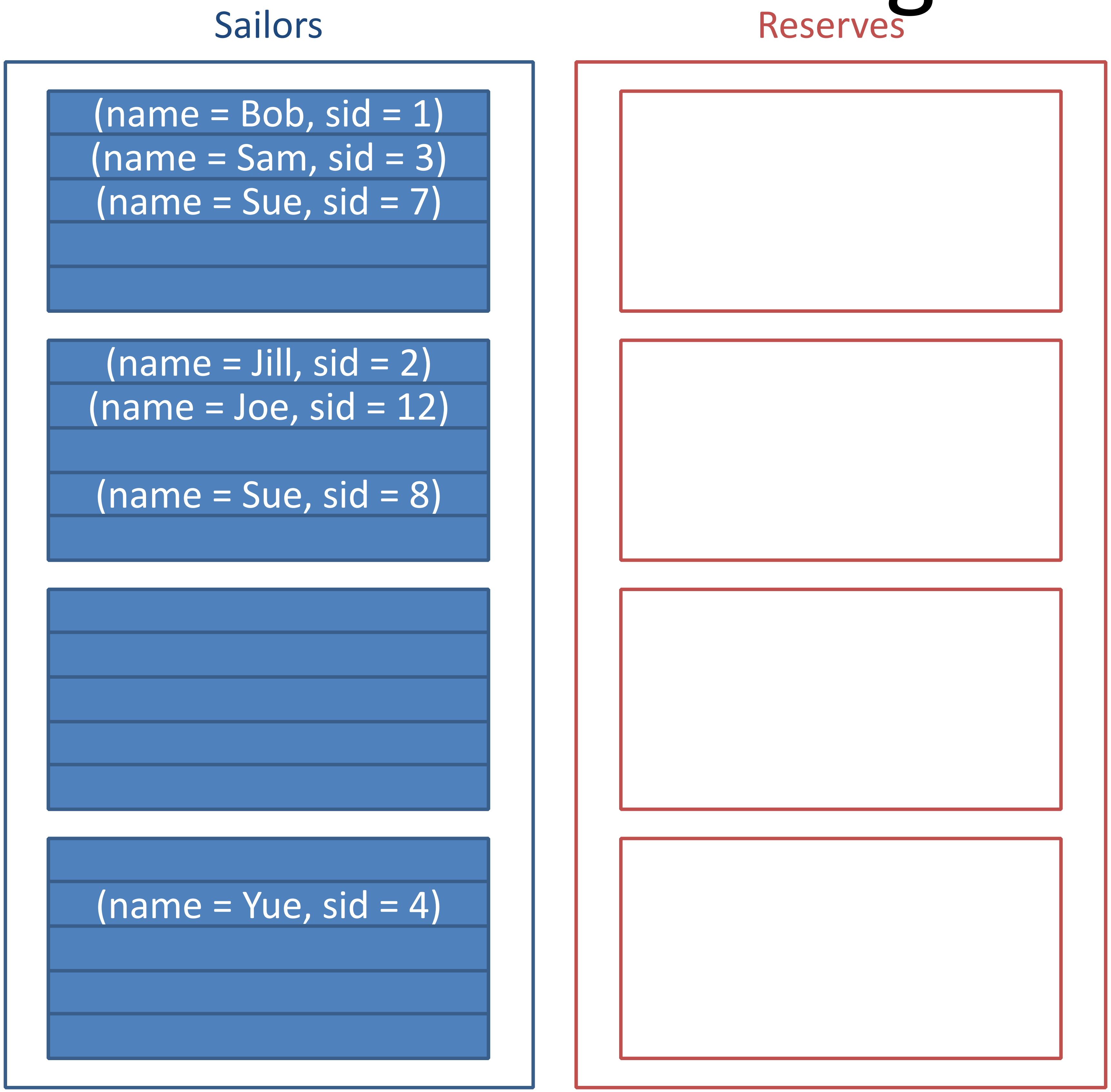
**Key idea:**

Sort S and R, then merge them!

**Steps:**

1. Sort S and R.
2. “Zip” or merge.

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)<br>
(name = Sam, sid = 3)<br>
(name = Sue, sid = 7)

(name = Jill, sid = 2)<br>
(name = Joe, sid = 12)<br>
(name = Sue, sid = 8)

(name = Yue, sid = 4)

Reserves

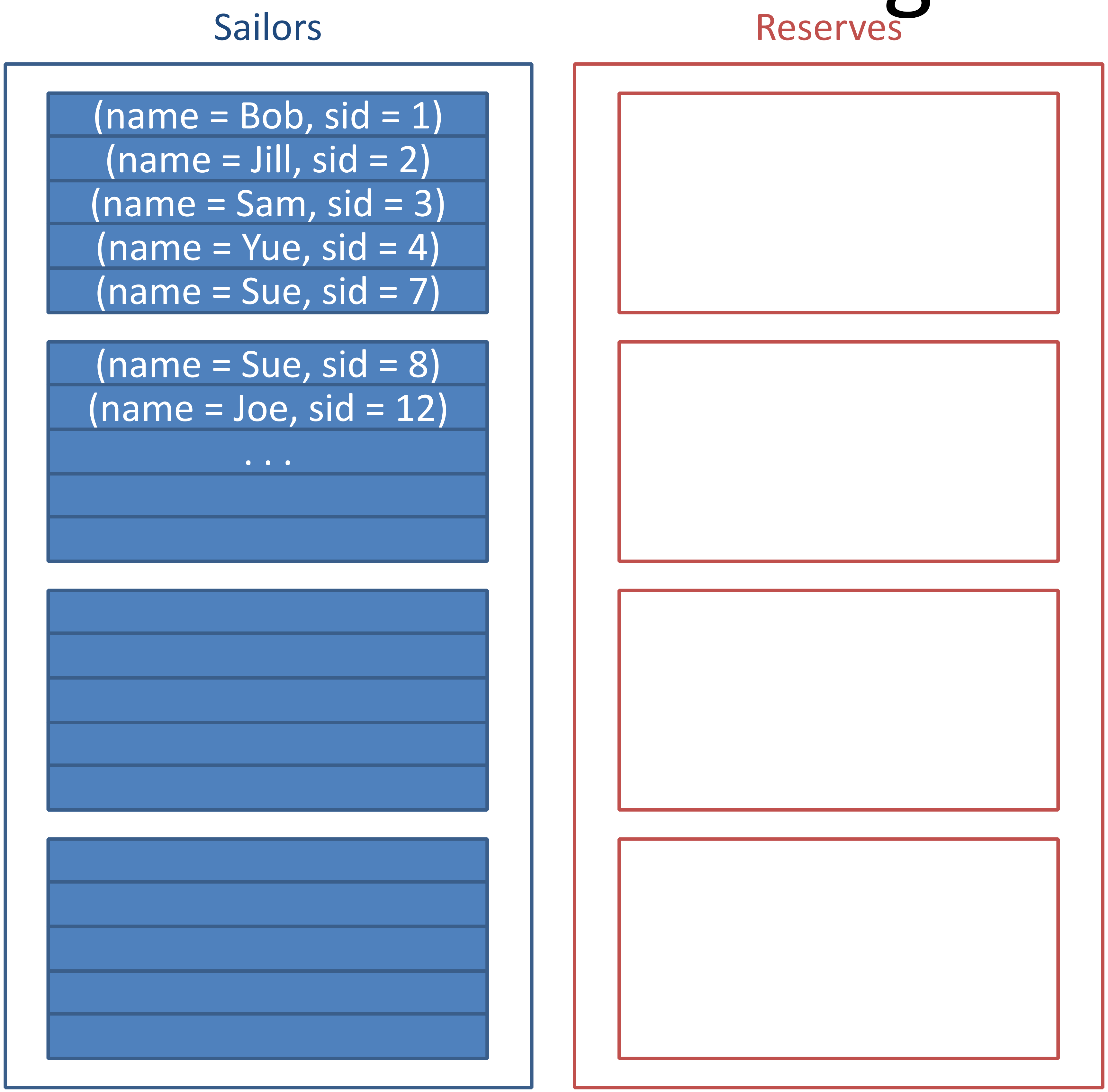
**Key idea:**

Sort S and R **on join column**, then merge them!

**Steps:**

1. Sort S and R.
2. “Zip” or merge.

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

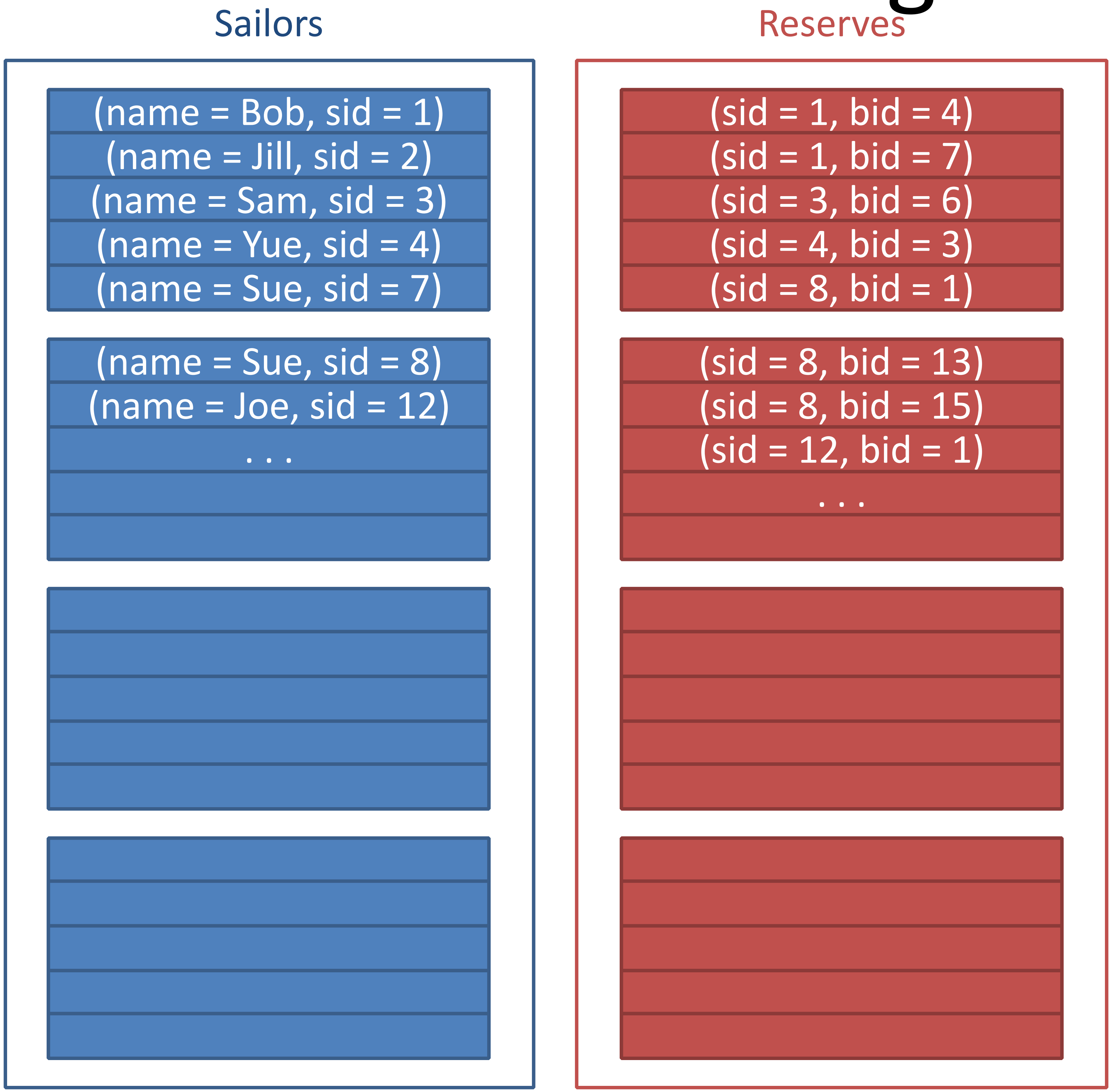
**Key idea:**

Sort S and R **on join column**, then merge them!

**Steps:**

1. Sort S and R.
2. “Zip” or merge.

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

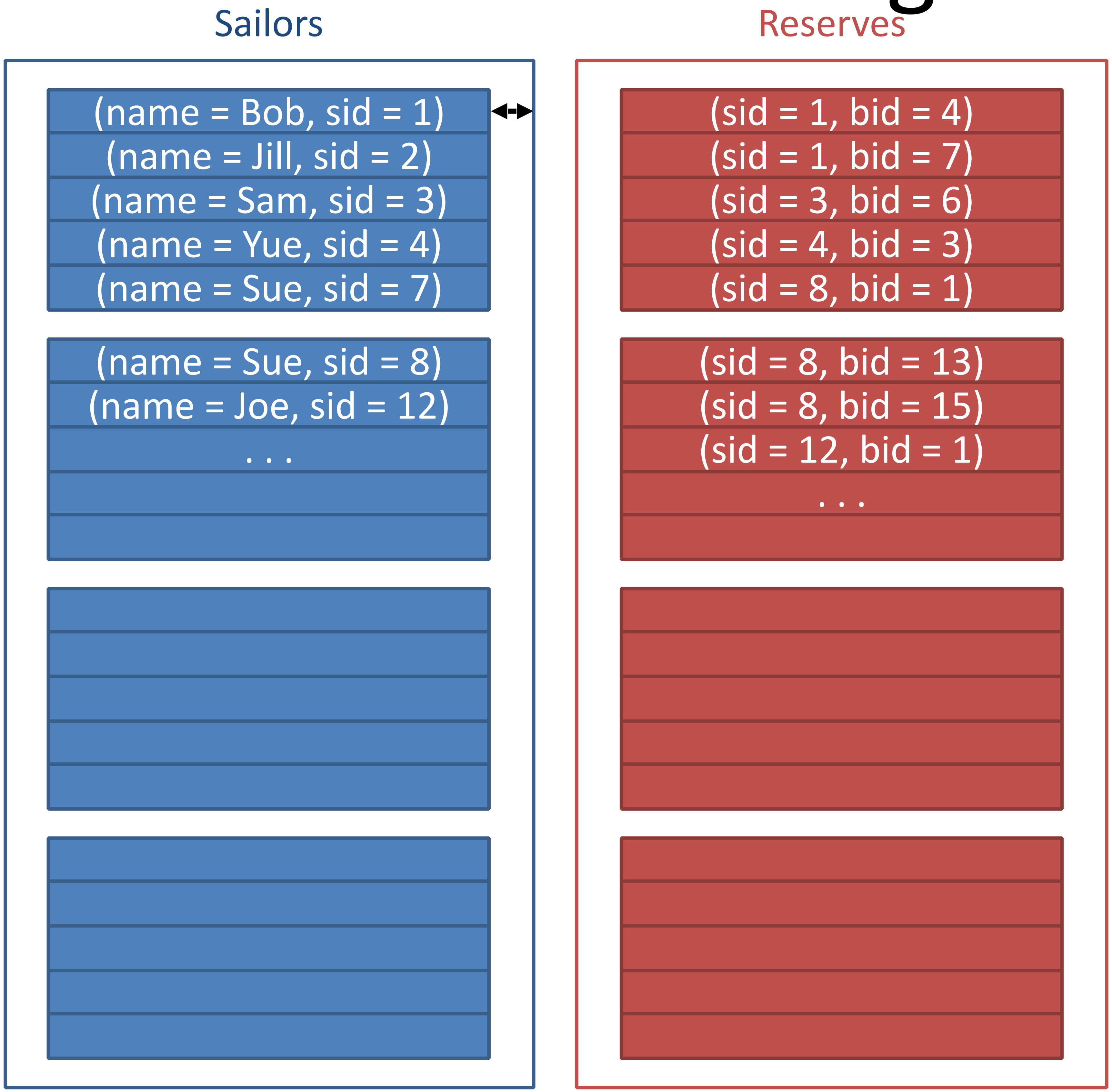
**Key idea:**

Sort S and R **on join column**, then merge them!

**Steps:**

1. Sort S and R.
2. “Zip” or merge.

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

**Key idea:**

Sort S and R **on join column**, then merge them!

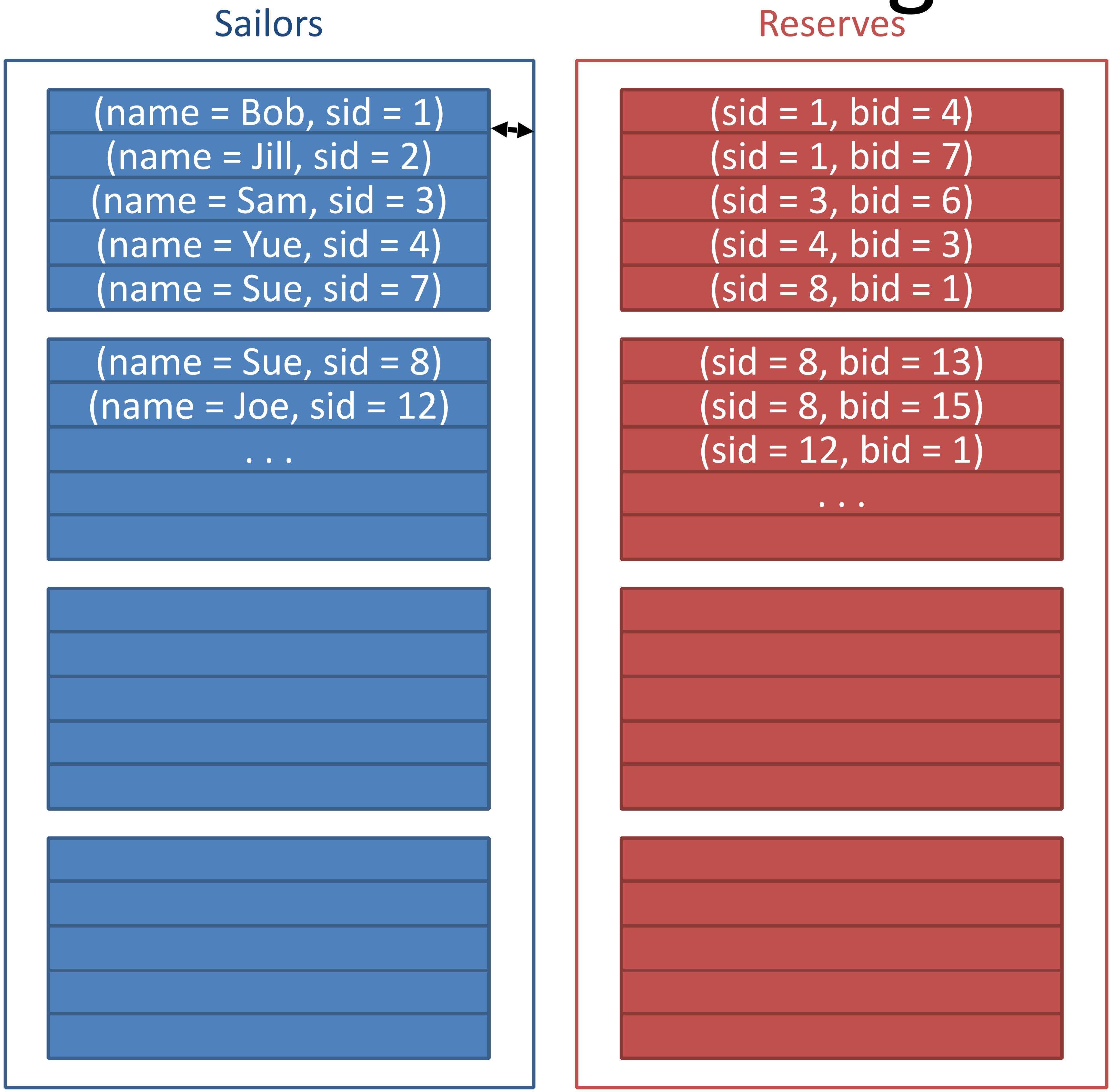
**Steps:**

1. Sort S and R.
2. “Zip” or merge.

**Output:**

(name = Bob, sid = 1, bid = 4)

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

**Key idea:**

Sort S and R **on join column**, then merge them!

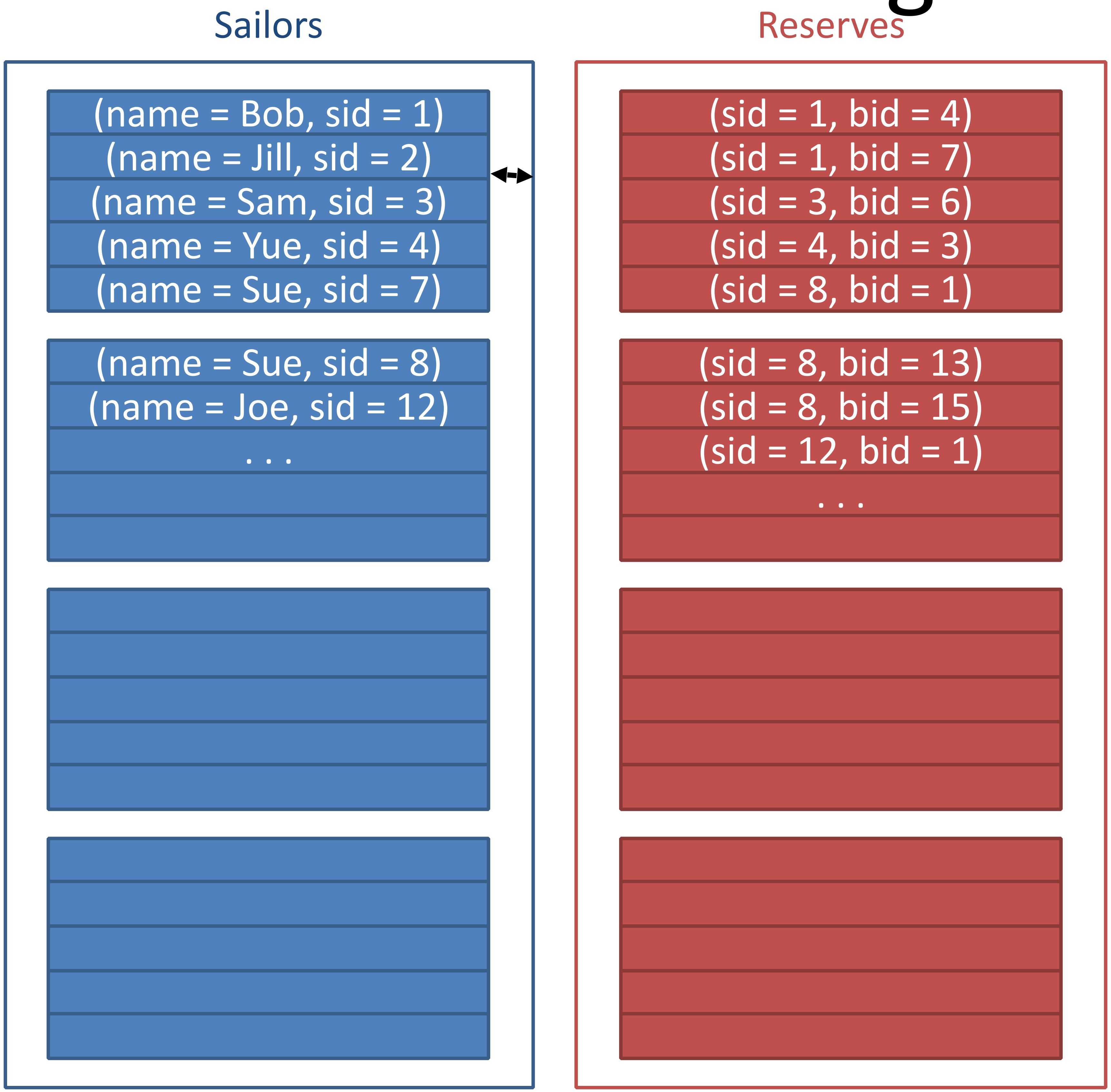
**Steps:**

1. Sort S and R.
2. “Zip” or merge.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

**Key idea:**

Sort S and R **on join column**, then merge them!

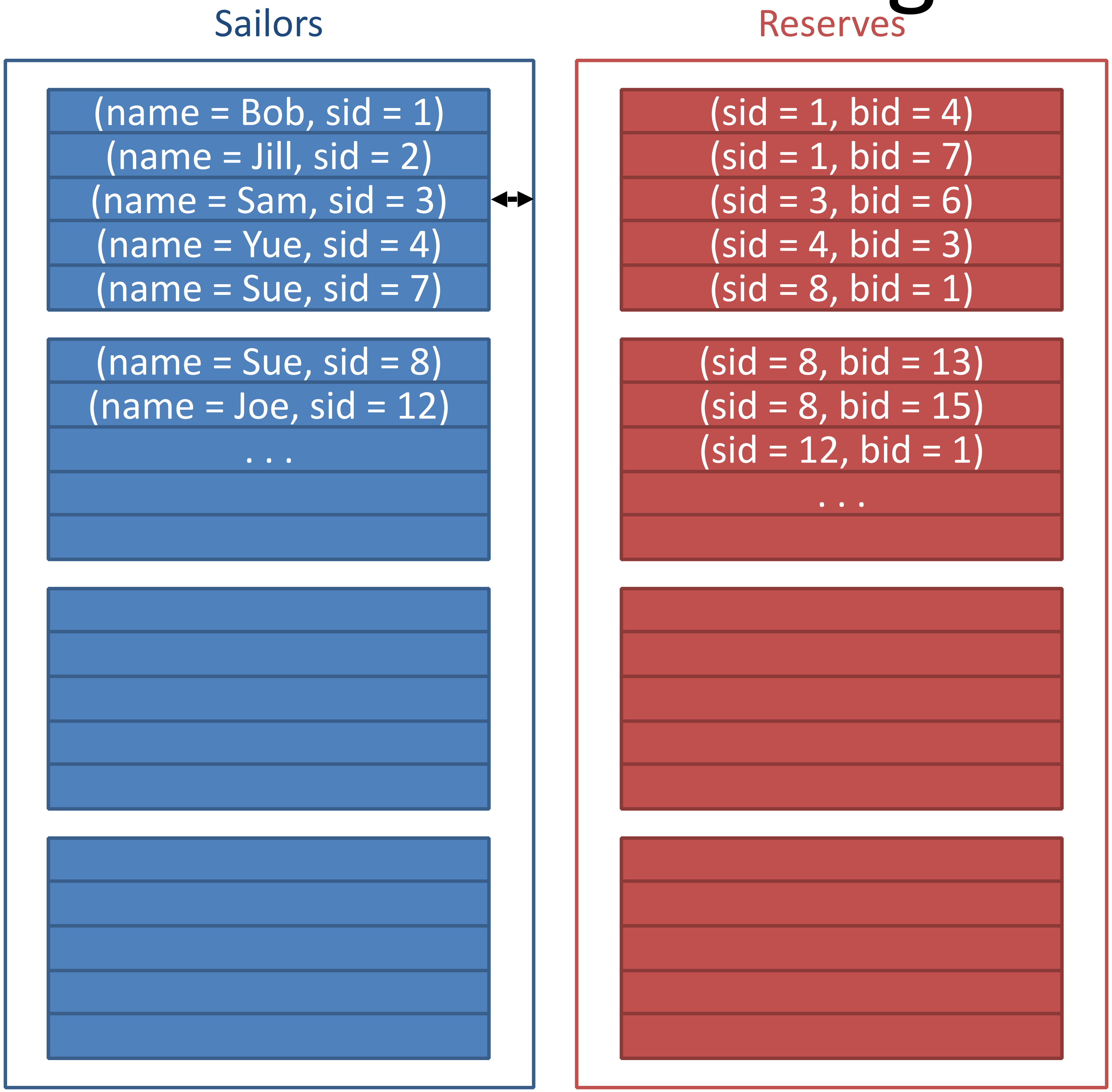
**Steps:**

1. Sort S and R.
2. "Zip" or merge.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

**Key idea:**

Sort S and R **on join column**, then merge them!

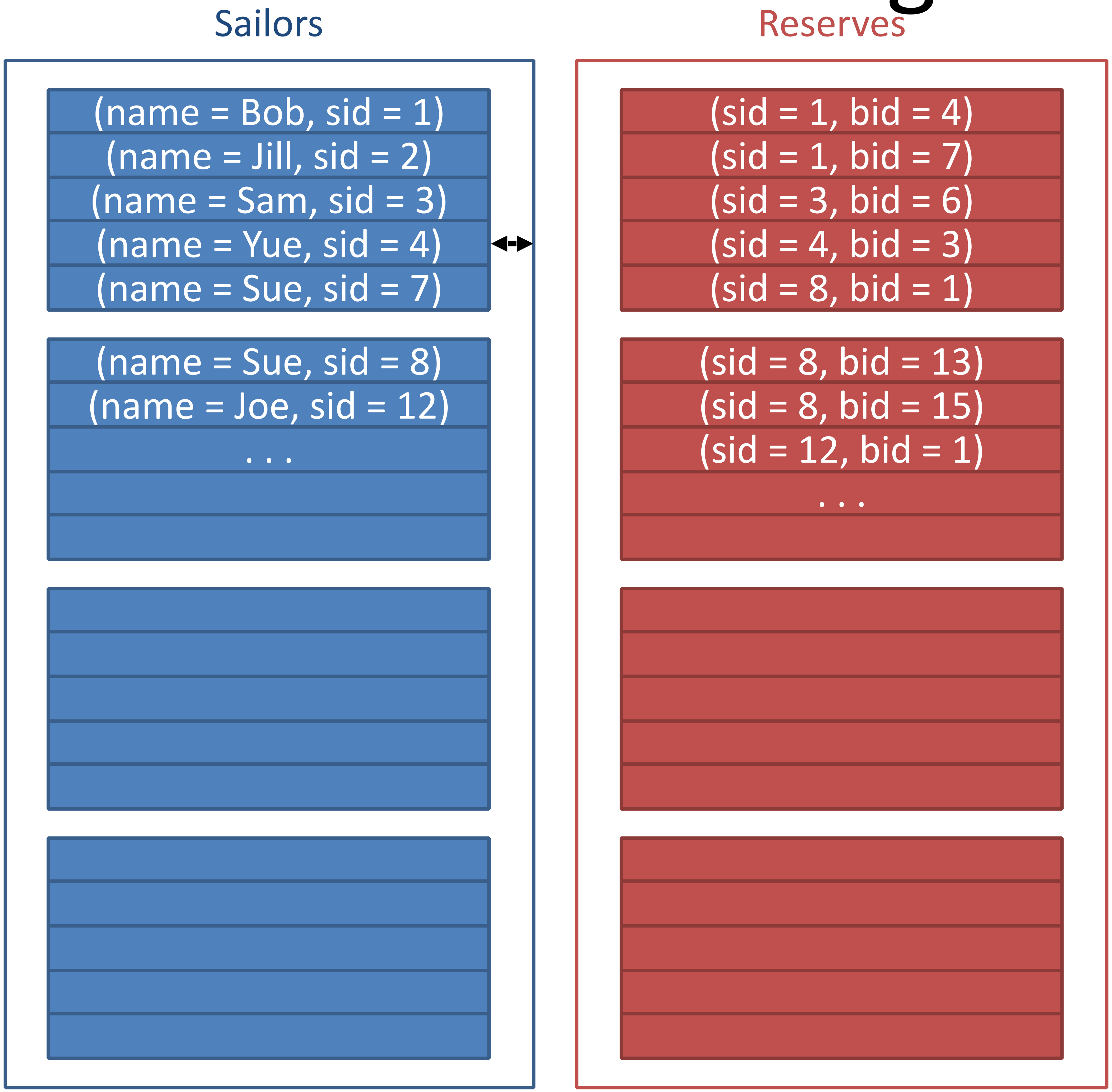
**Steps:**

1. Sort S and R.
2. “Zip” or merge.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

**Key idea:**

Sort S and R **on join column**, then merge them!

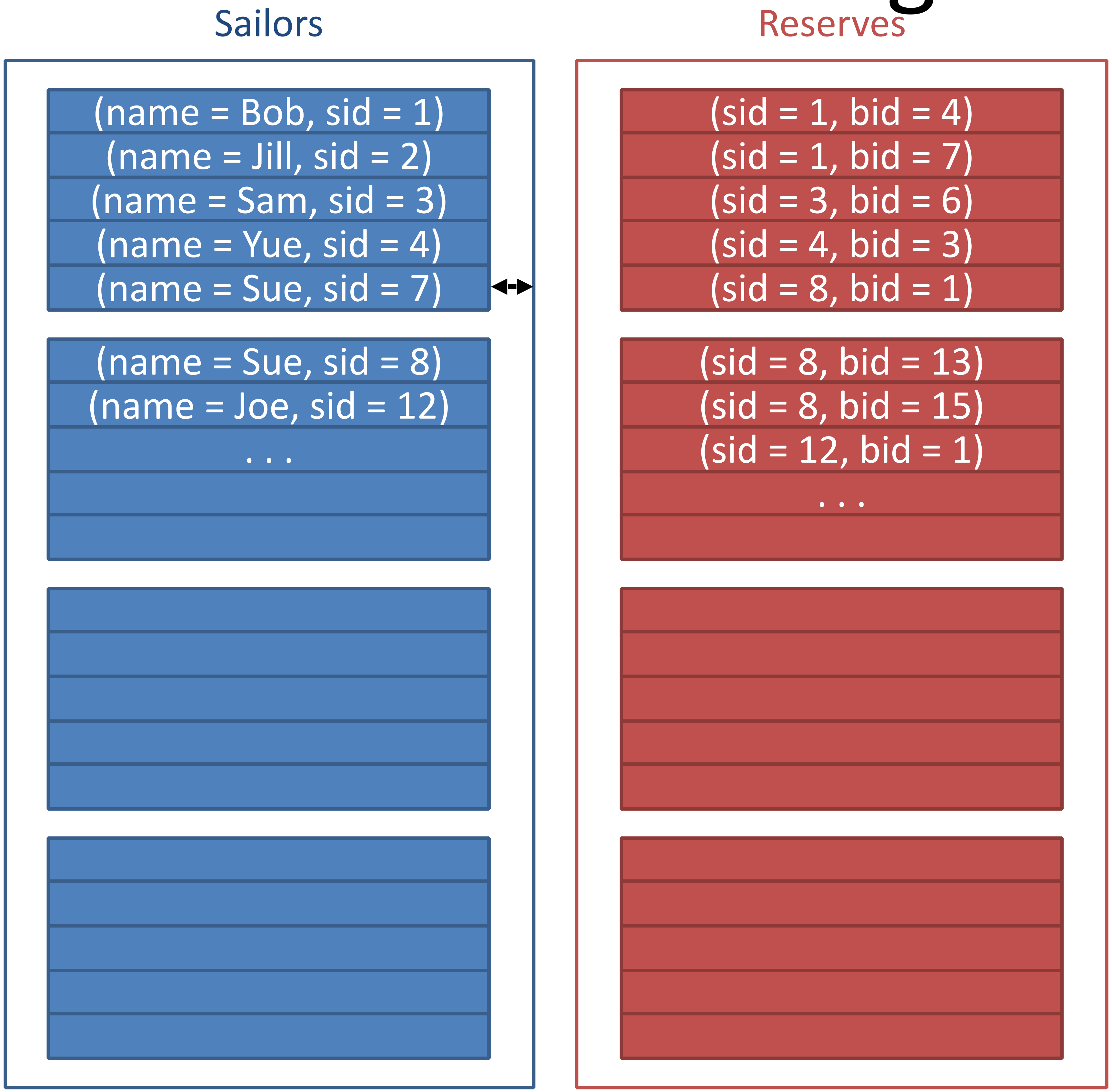
**Steps:**

1. Sort S and R.
2. "Zip" or merge.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)
. . .

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

**Key idea:**

Sort S and R **on join column**, then merge them!

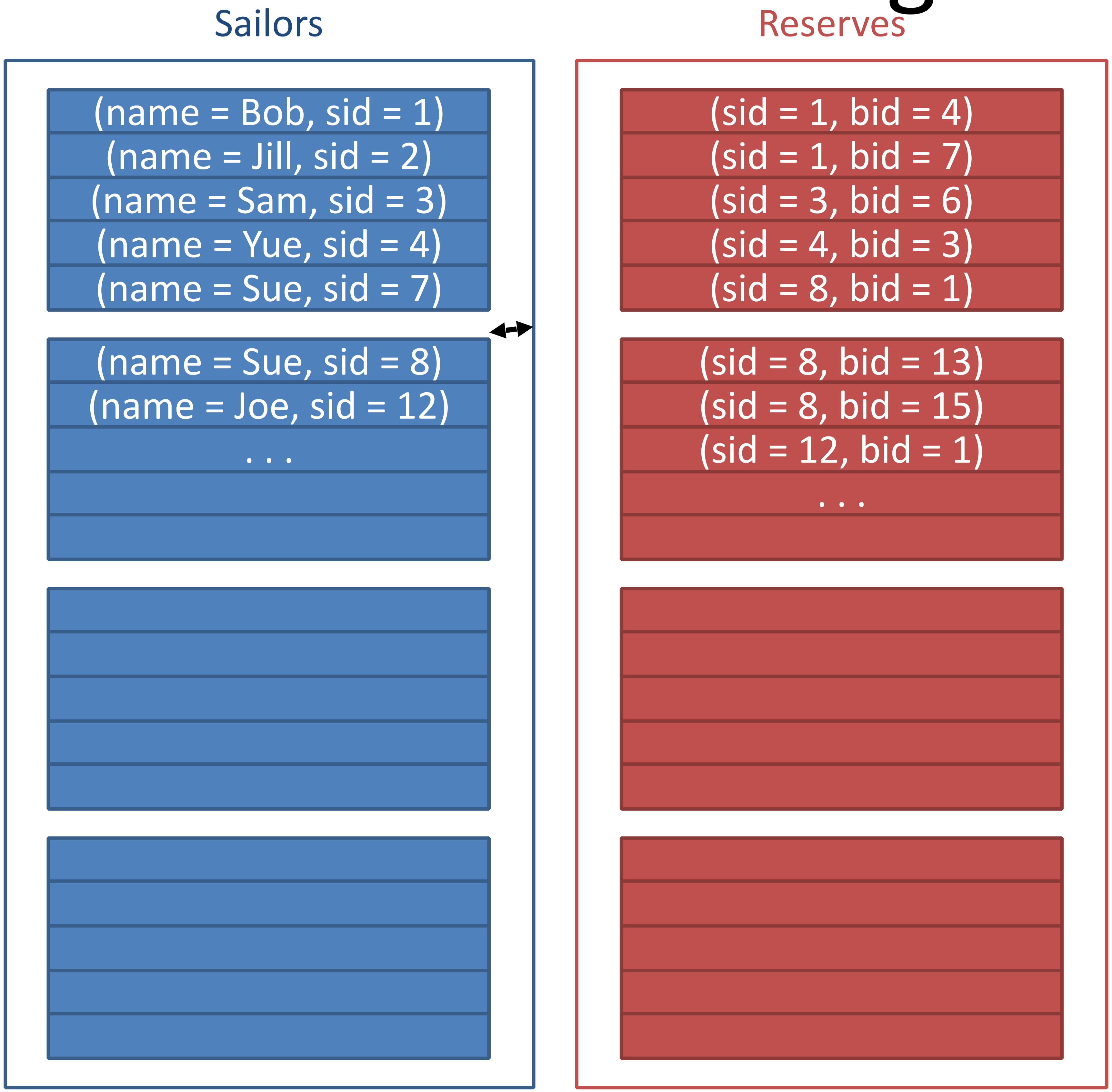
**Steps:**

1. Sort S and R.
2. “Zip” or merge.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)
. . .

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

**Key idea:**

Sort S and R **on join column**, then merge them!

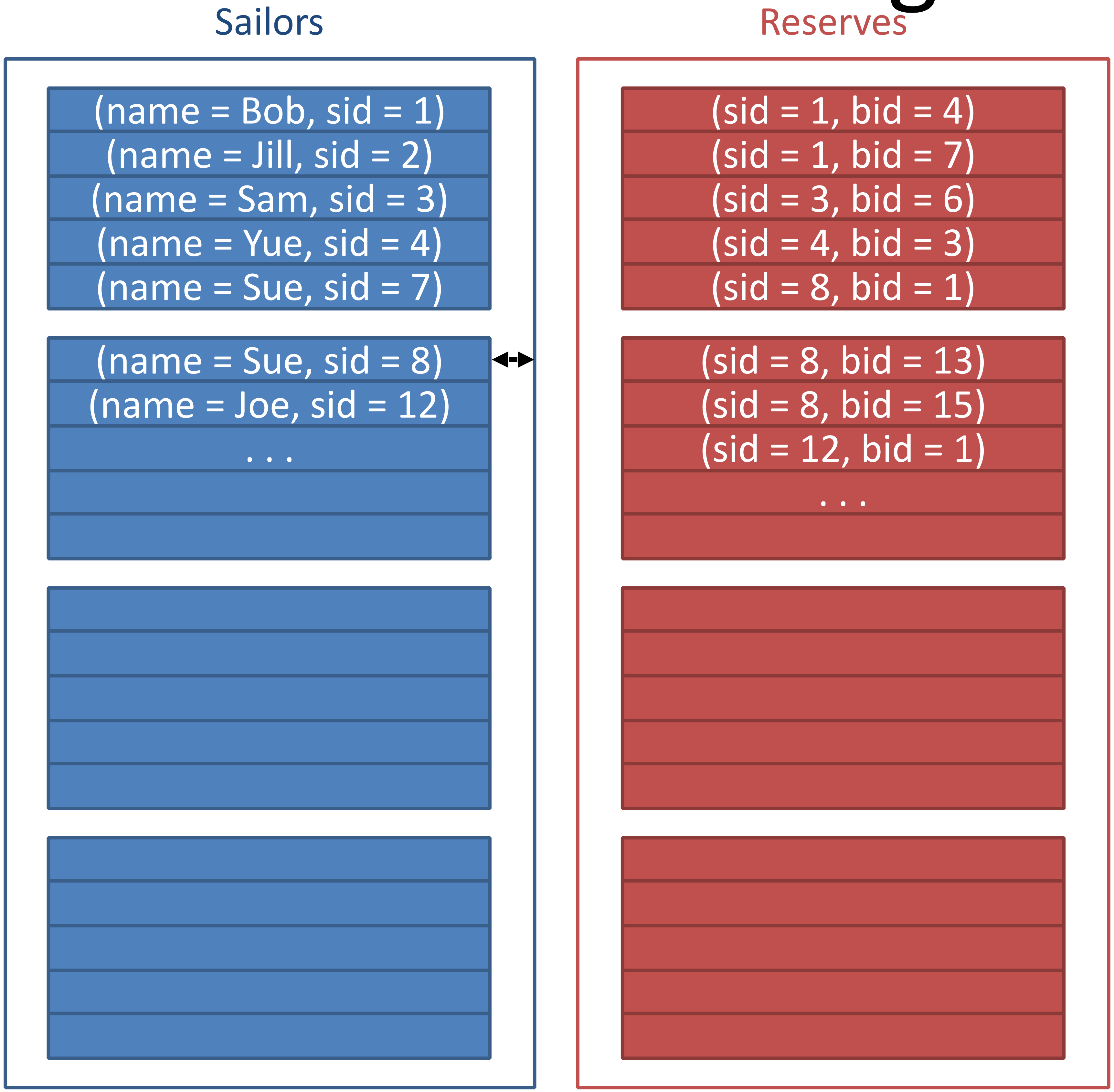
**Steps:**

1. Sort S and R.
2. “Zip” or merge.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)
. . .

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

**Key idea:**

Sort S and R **on join column**, then merge them!

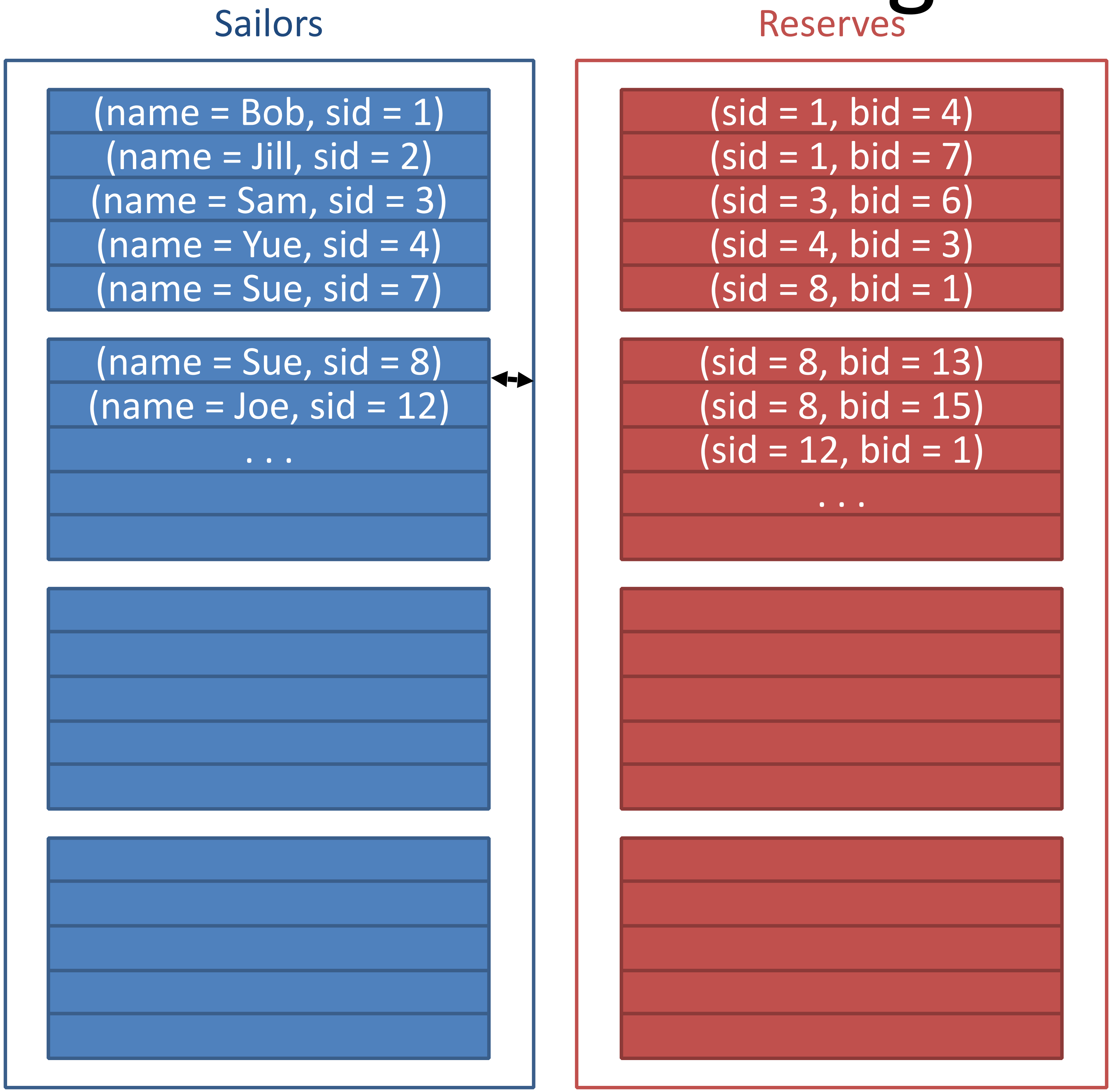
**Steps:**

1. Sort S and R.
2. "Zip" or merge.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)
. . .

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

**Key idea:**

Sort S and R **on join column**, then merge them!

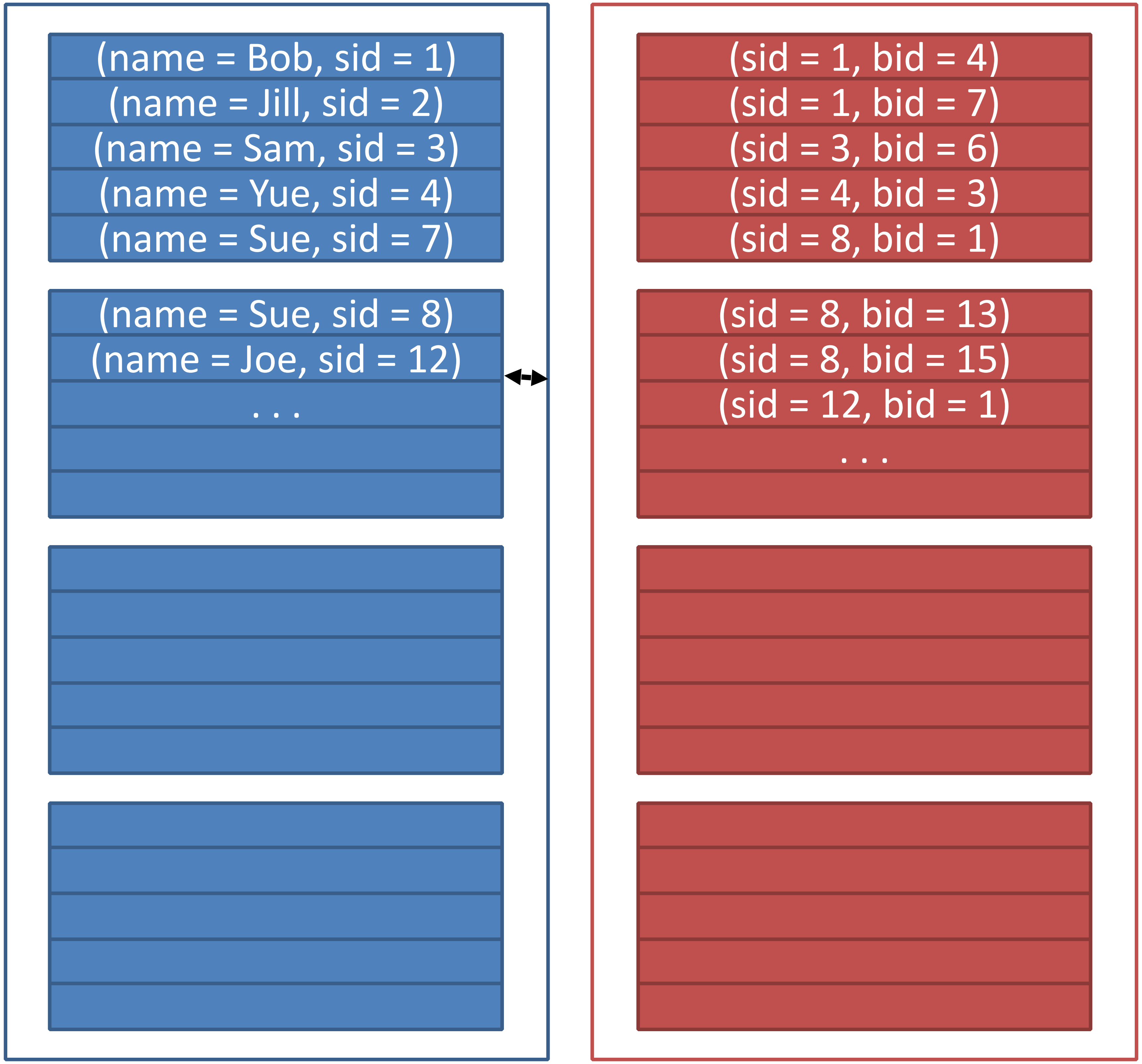
**Steps:**

1. Sort S and R.
2. "Zip" or merge.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)
. . .

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

**Key idea:**

Sort S and R **on join column**, then merge them!

**Steps:**

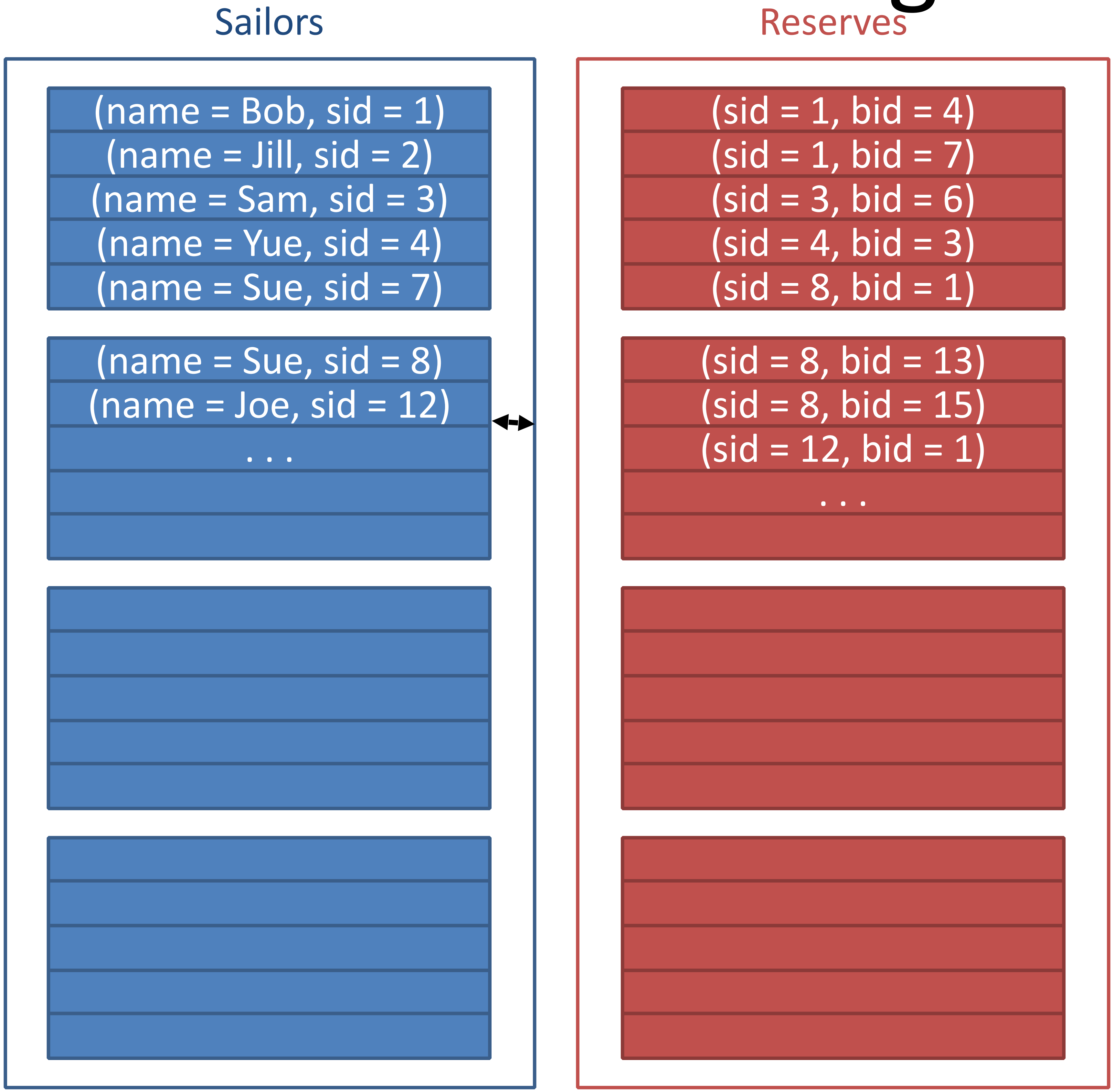
1. Sort S and R.
2. "Zip" or merge.

**Output:**

(name = Bob, sid = 1, bid = 4)
(name = Bob, sid = 1, bid = 7)
(name = Sam, sid = 3, bid = 6)
. . .

**Notation: [S] == “# pages in S” ; |S| == “# tuples in S”**

# Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

**Key idea:**

Sort S and R **on join column**, then merge them!

**Steps:**

1. Sort S and R.
2. “Zip” or merge.

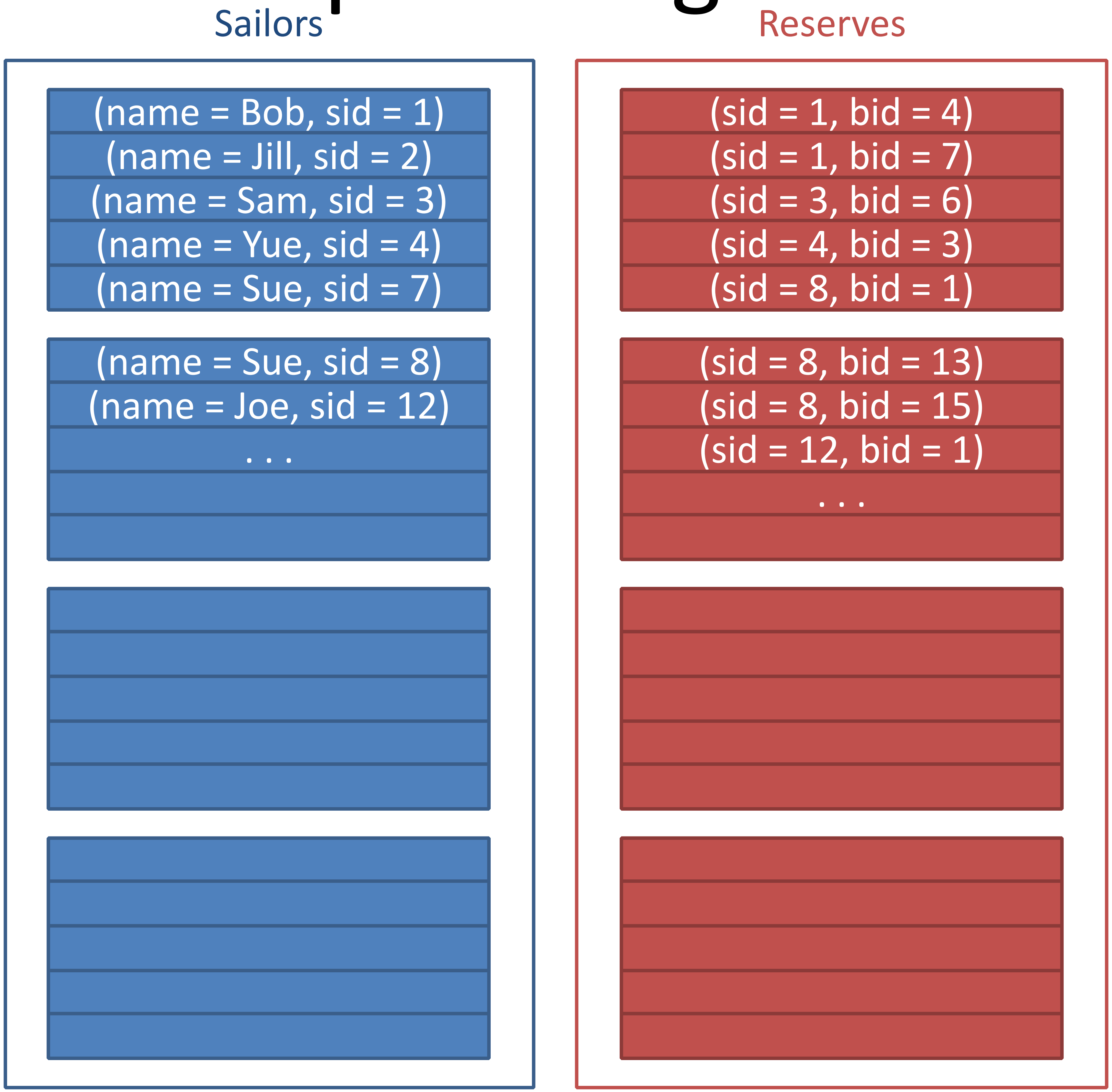
**I/Os:**

~5([S] + [R])

Sorting: 4([S]+[R])

Merging: [S]+[R]

# Optimizing Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Sam, sid = 3)
(name = Yue, sid = 4)
(name = Sue, sid = 7)

(name = Sue, sid = 8)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 3, bid = 6)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 12, bid = 1)
. . .

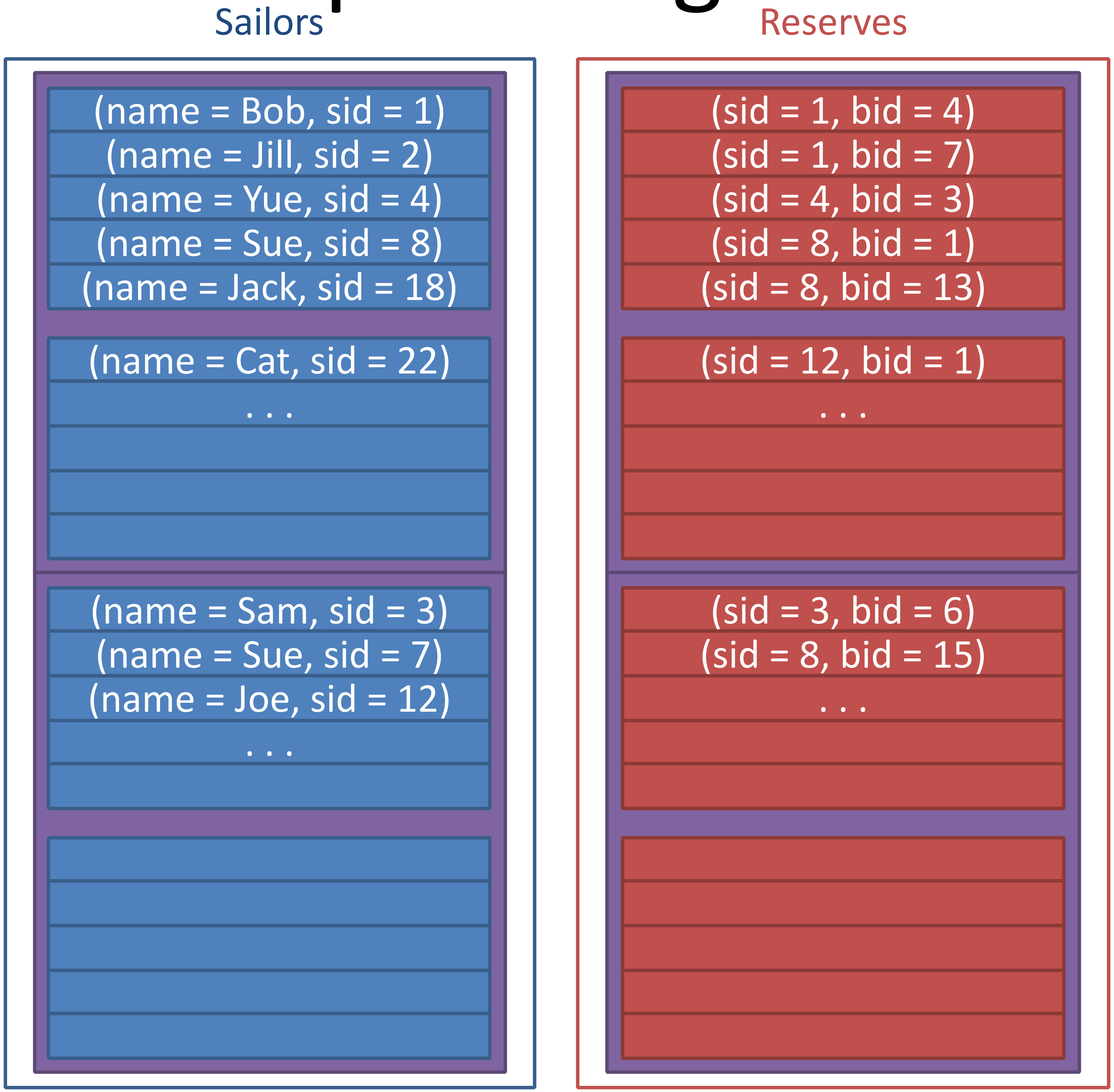
**Key idea:**

Internal Sort on both. Perform merge on all runs!

**Steps:**

1. Internal sort S and R. (Pass 0)
2. Merge all runs.

# Optimizing Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Yue, sid = 4)
(name = Sue, sid = 8)
(name = Jack, sid = 18)

(name = Cat, sid = 22)
. . .

(name = Sam, sid = 3)
(name = Sue, sid = 7)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 4, bid = 3)
(sid = 8, bid = 1)
(sid = 8, bid = 13)

(sid = 12, bid = 1)
. . .

(sid = 3, bid = 6)
(sid = 8, bid = 15)
. . .

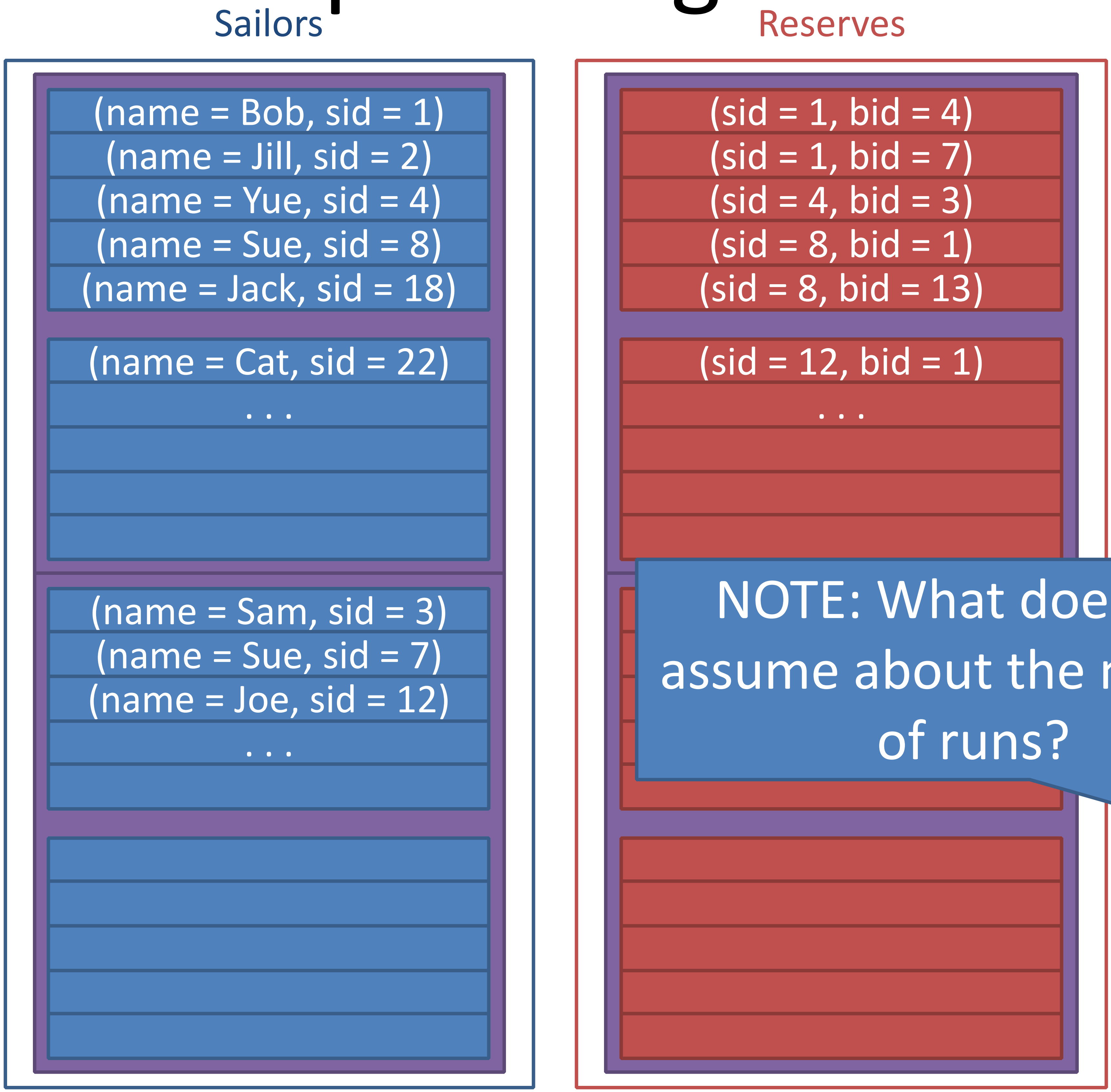
**Key idea:**

Internal Sort on both. Perform merge on all runs!

**Steps:**

1. Internal sort S and R. (Pass 0)
2. Merge all runs.

# Optimizing Sort-Merge Join



Sailors

(name = Bob, sid = 1)
(name = Jill, sid = 2)
(name = Yue, sid = 4)
(name = Sue, sid = 8)
(name = Jack, sid = 18)

(name = Cat, sid = 22)
. . .

(name = Sam, sid = 3)
(name = Sue, sid = 7)
(name = Joe, sid = 12)
. . .

Reserves

(sid = 1, bid = 4)
(sid = 1, bid = 7)
(sid = 4, bid = 3)
(sid = 8, bid = 1)
(sid = 8, bid = 13)

(sid = 12, bid = 1)
. . .

NOTE: What does this assume about the number of runs?

**Key idea:**

Internal Sort on both. Perform merge on all runs!

**Steps:**

1. Internal sort S and R. (Pass 0)

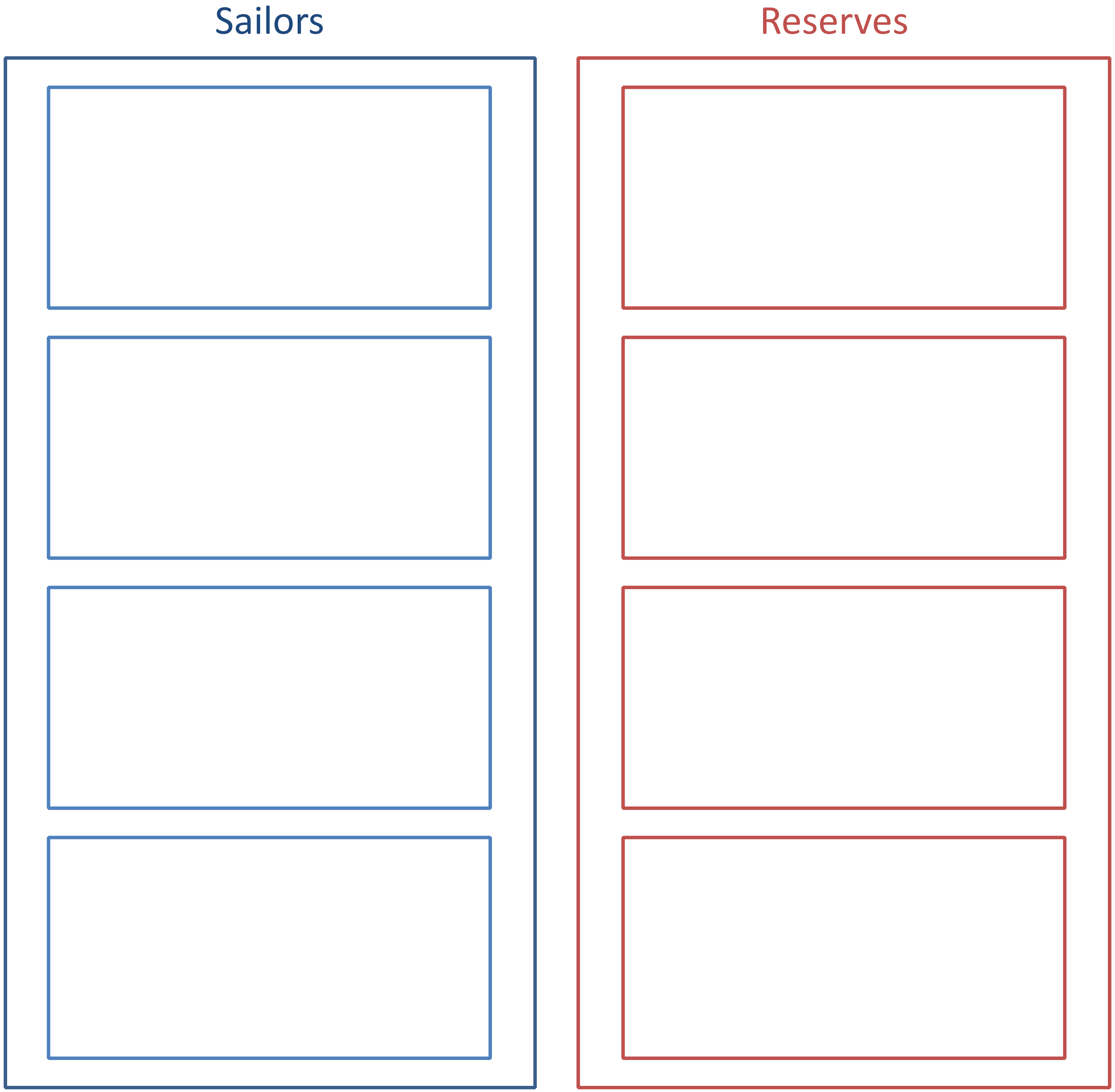
ge all runs.

~3([S] + [R])

Pass 0: 2([S]+[R])

Merging: [S]+[R]

# Hash-Join



Sailors

Reserves

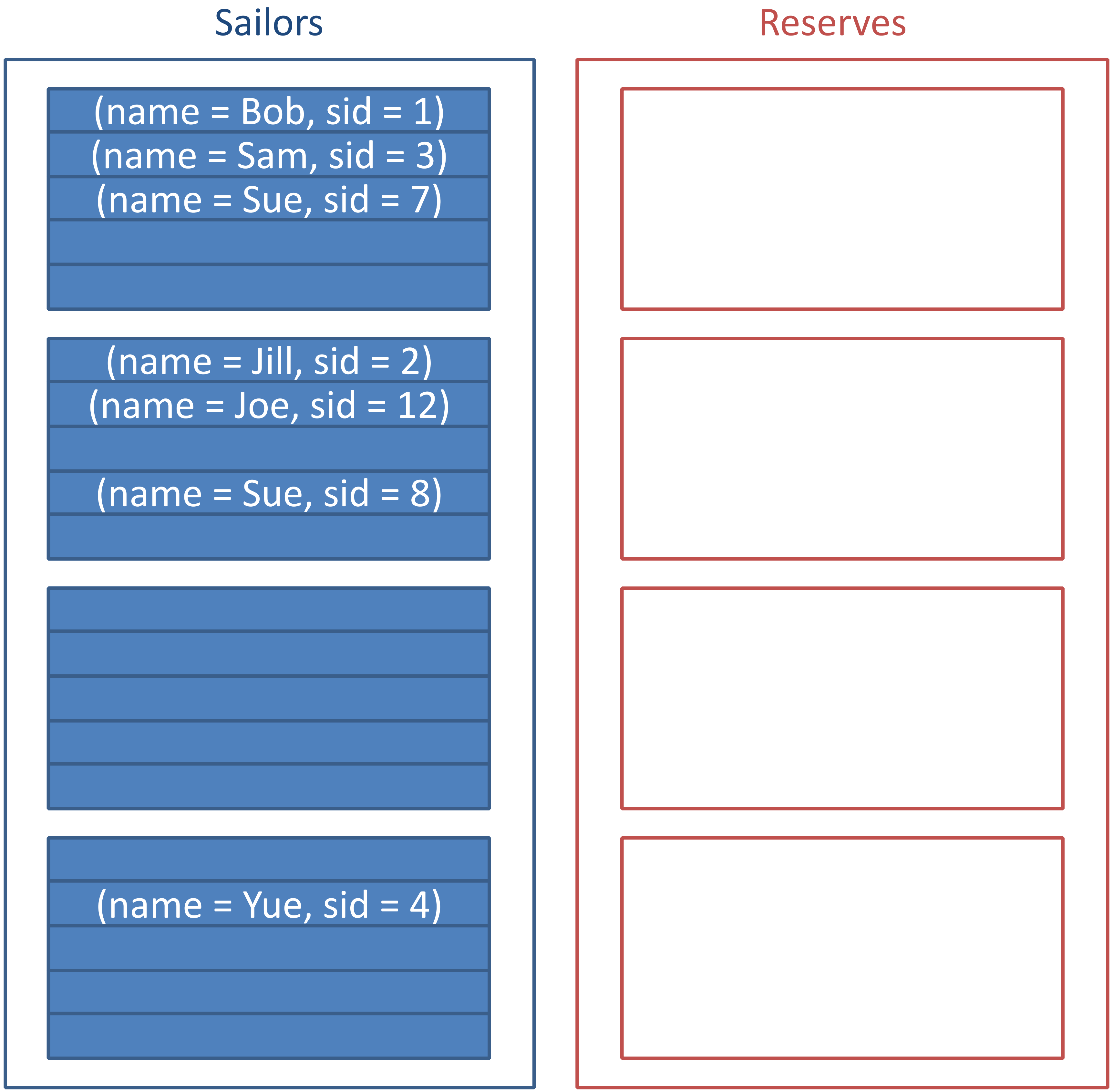
**Key idea:**

Partition S and R using same hash fn, then collect same partitions

**Steps:**

1. Partition S and R
2. Re-Hash, collect

# Hash-Join



Sailors

(name = Bob, sid = 1)
(name = Sam, sid = 3)
(name = Sue, sid = 7)

(name = Jill, sid = 2)
(name = Joe, sid = 12)

(name = Sue, sid = 8)

(name = Yue, sid = 4)

Reserves

**Key idea:**

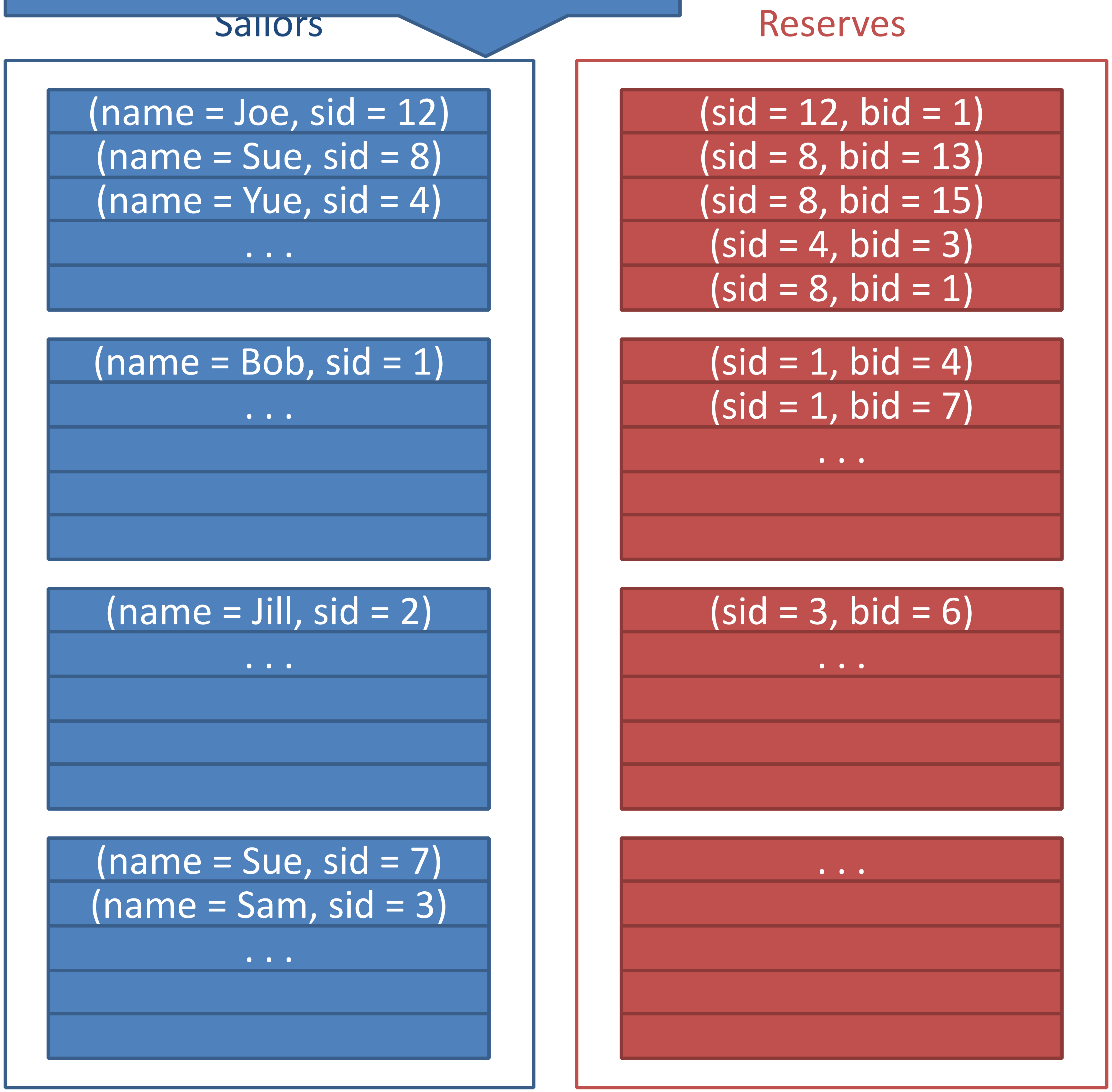
Partition S and R using same hash fn, then collect same partitions

**Steps:**

1. Partition S and R
2. Re-Hash, collect

Hash function: **sid mod 4**

# Hash-Join



Sailors

(name = Joe, sid = 12)
(name = Sue, sid = 8)
(name = Yue, sid = 4)
. . .

(name = Bob, sid = 1)
. . .

(name = Jill, sid = 2)
. . .

(name = Sue, sid = 7)
(name = Sam, sid = 3)
. . .

Reserves

(sid = 12, bid = 1)
(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 1, bid = 4)
(sid = 1, bid = 7)
. . .

(sid = 3, bid = 6)
. . .

. . .

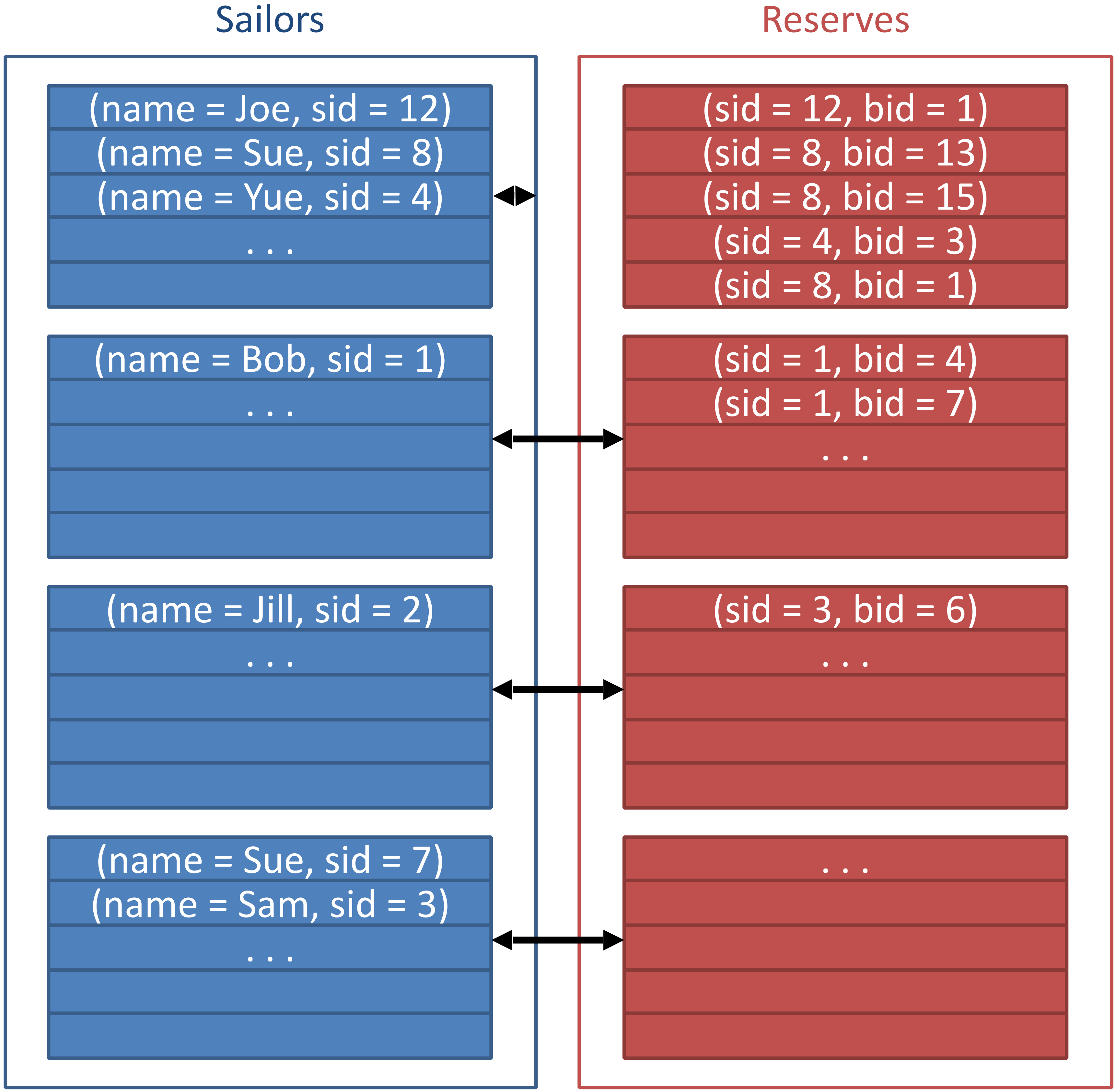
**Key idea:**

Partition S and R using same hash fn, then collect same partitions

**Steps:**

1. Partition S and R
2. Re-Hash, collect

# Hash-Join



Sailors

(name = Joe, sid = 12)
(name = Sue, sid = 8)
(name = Yue, sid = 4)
. . .

(name = Bob, sid = 1)
. . .

(name = Jill, sid = 2)
. . .

(name = Sue, sid = 7)
(name = Sam, sid = 3)
. . .

Reserves

(sid = 12, bid = 1)
(sid = 8, bid = 13)
(sid = 8, bid = 15)
(sid = 4, bid = 3)
(sid = 8, bid = 1)

(sid = 1, bid = 4)
(sid = 1, bid = 7)
. . .

(sid = 3, bid = 6)
. . .

. . .

**Key idea:**

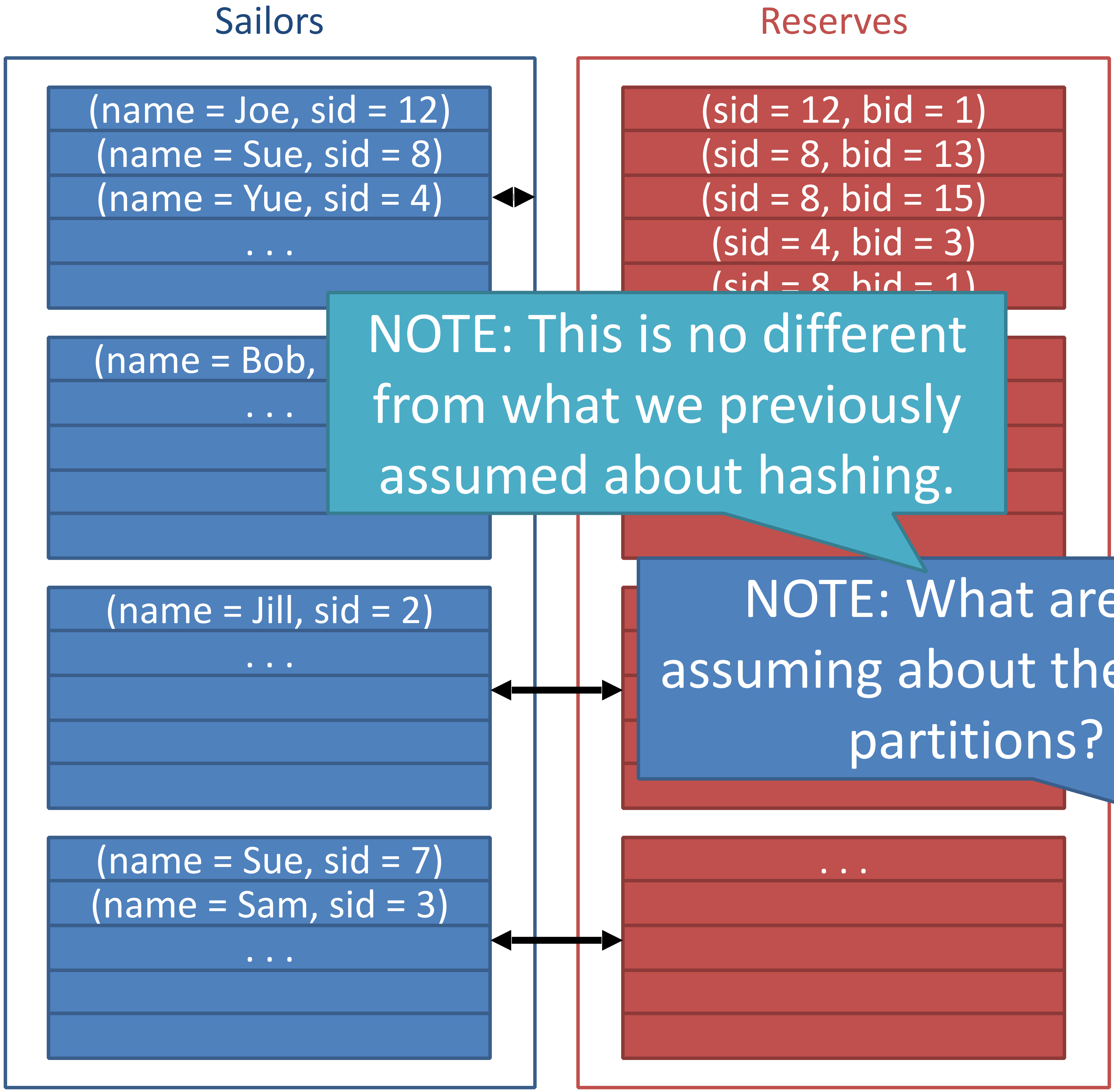
Partition S and R using same hash fn, then collect same partitions

**Steps:**

1. Partition S and R
2. Re-Hash, collect

**Notation: [S] == "# pages in S" ; |S| == "# tuples in S"**

# Hash-Join



NOTE: This is no different from what we previously assumed about hashing.

NOTE: What are we assuming about the size of partitions?

**Key idea:**

Partition S and R using same hash fn, then collect same partitions

**Steps:**

1. Partition S and R

Hash, collect

3([S] + [R])

Partition: 2([S]+[R])

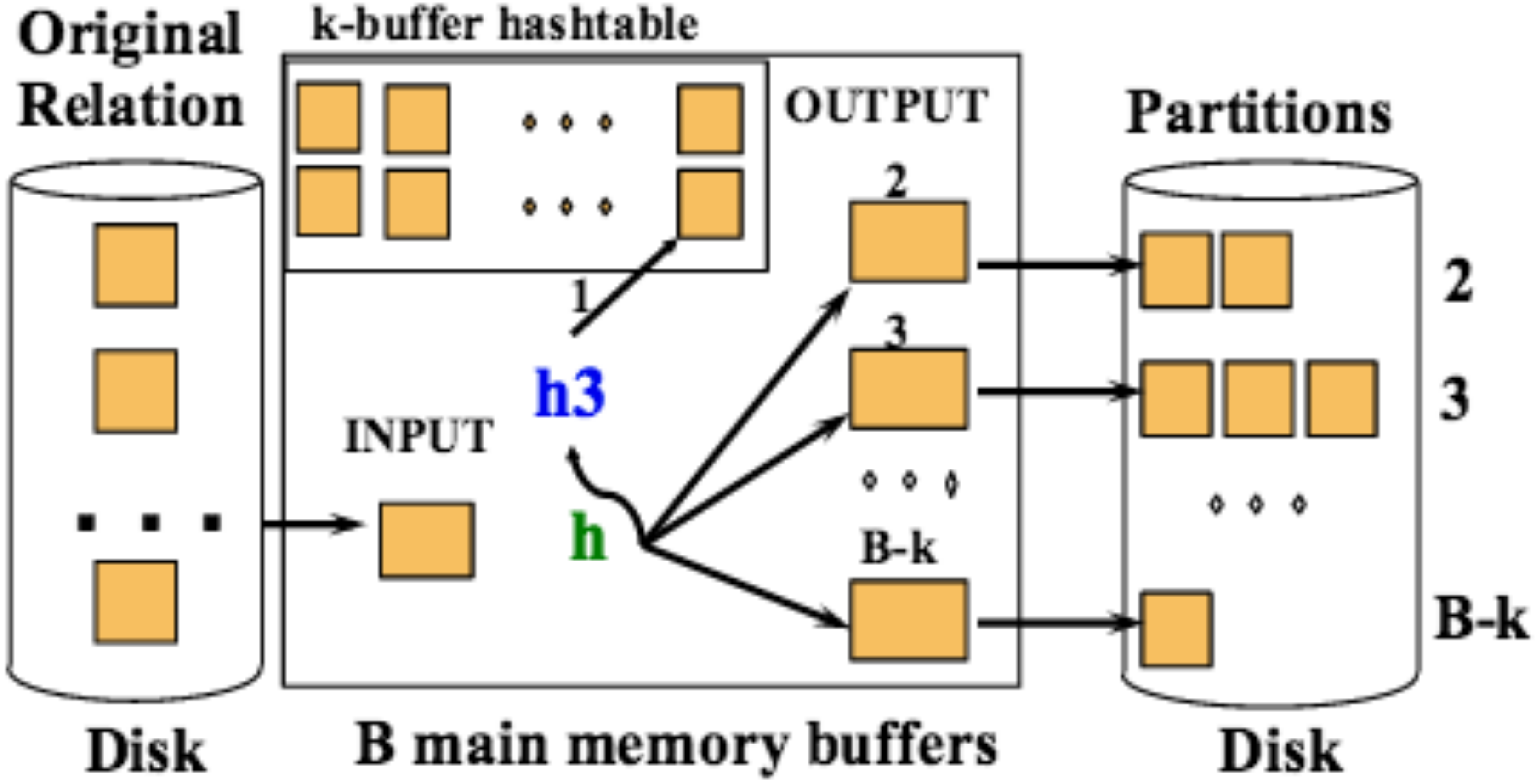
Re-Hash: [S]+[R]

# Hybrid Hashing (Hash Join Made Better)

- The hybrid hash join takes advantage of more available memory. During the partitioning phase, the hybrid hash join uses the available memory for two purposes:
  - To hold the current output buffer page for each of the k partitions
  - To hold an entire partition in-memory, known as "partition 0"
- Because partition 0 is never written to or read from disk, the hybrid hash join typically performs fewer I/O operations than regular hash join. Note that this algorithm is memory-sensitive, because there are two competing demands for memory (the hash table for partition 0, and the output buffers for the remaining partitions). Choosing too large a hash table might cause the algorithm to recurse because one of the non-zero partitions is too large to fit into memory.

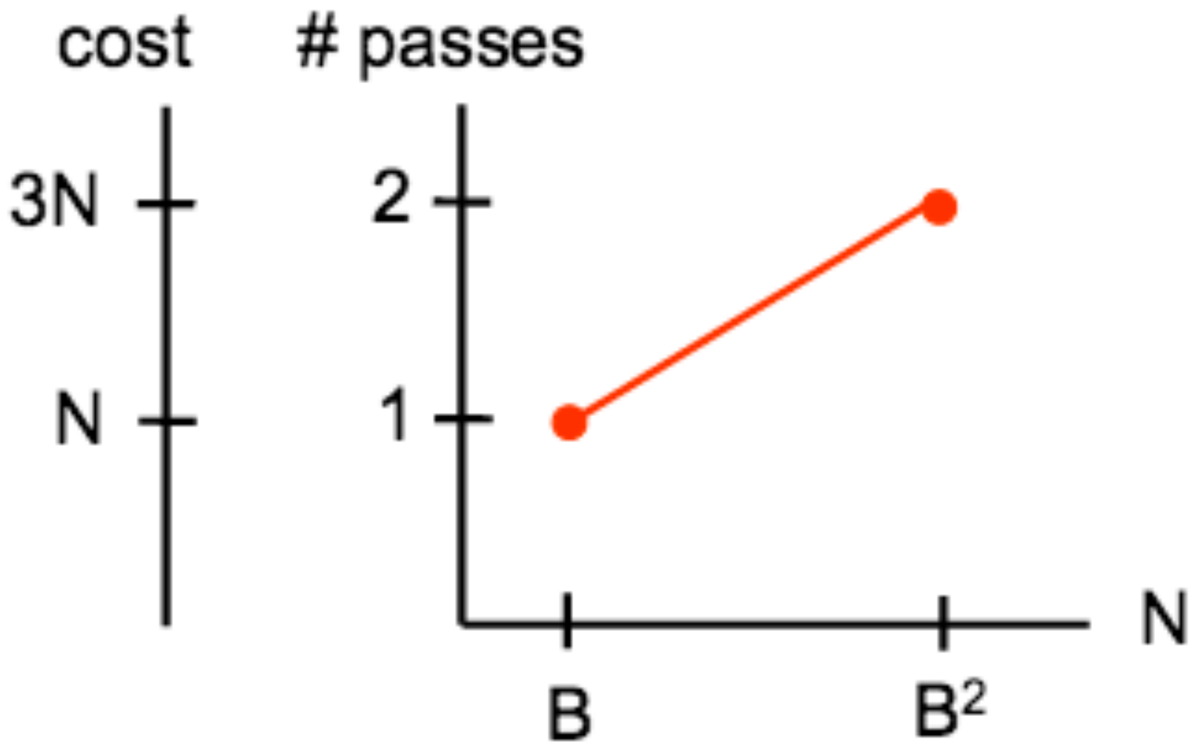
# Hybrid Hashing

Idea: keep one of the hash buckets in memory!



# Cost savings: hybrid hashing

- Now:



# Join Cheatsheet

**Notation: [S] == "# pages in S" ;**
**|S| == "# tuples in S"**

- Chunk nested loop join
  - Take **k pages** of S and match with each page of R.
  - Total Cost: [S] + ([S] / k)*[R]
- Sort merge join
  - Sort S and R **on join column**, then merge them!
  - Total Cost: ~5([S] + [R])
- Hash join
  - Partition S and R using same hash fn, then collect same partitions
  - Total Cost: ~3([S] + [R])
    - Assuming len(partition) ≤ B pages

# Joins that we didn’t go over!

- Index Nested Loop Join
  - More relevant when we start talking about B+ Trees, Indicies, and Text Search
  - Look at lecture slides for clear pictures and understanding

# When is a chunk-nested loops join the best?

# When is a chunk-nested loops join the best?

- Not using an equality predicate
- Join is just a cross product

# When is a sort-merge join the best?

# When is a sort-merge join the best?

- Skewed input data
- Small memory size
- Want sorted output/have sorted input

# When is a hash-join the best?

# When is a hash-join the best?

- One partition large, the other small (can keep in memory) —> Hybrid Hashing

Do questions #1 and #2
(c & d)

IMPORTANT