CS 564: Database Management Systems

Lecture 36: Distributed Analytical Database

Xiangyao Yu

4/22/2024

Announcement

Past final exams posted on canvas

Final exam

- May 6th, 10:05am-12:05pm
- Room 1: VAN VLECK B130 (**lastname ≤ ‘Mao’**)
- Room 2: NOLAND 132 (**lastname > ‘Mao’**)
- Two cheat sheets allowed; can reuse the midterm one
- Cumulative with roughly 70% content after midterm
- Contact instructor by **May 1st** if need McBurney accommodation

Course evaluation: https://heliocampusac.wisc.edu/

Outline

Distributed architecture

Data partitioning

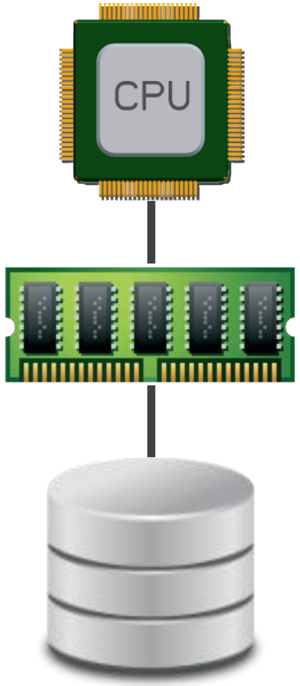
Query execution

- Merge and Split operators

Join execution

- Co-partitioned
- Partition-based
- Broadcast-based

Centralized vs. Distributed Database



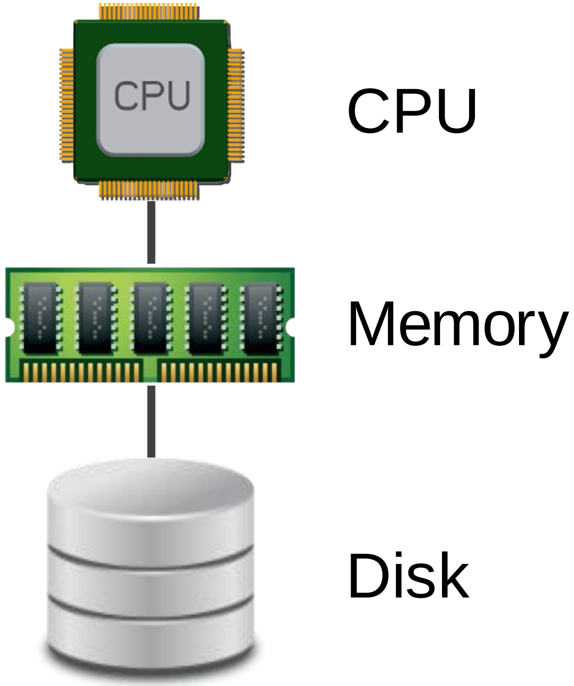
CPU

Memory

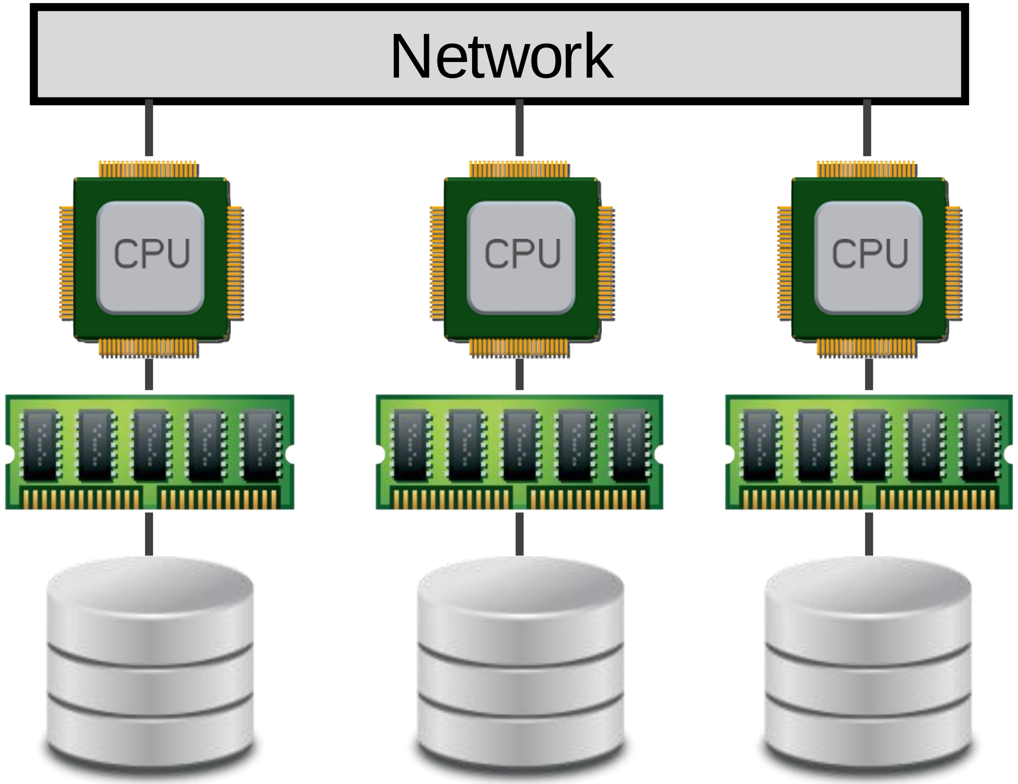
Disk

Centralized database

Centralized vs. Distributed Database



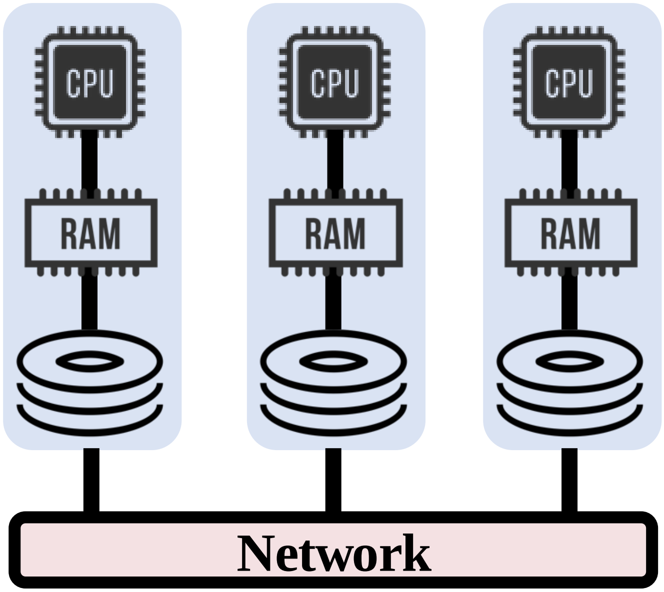
Centralized database



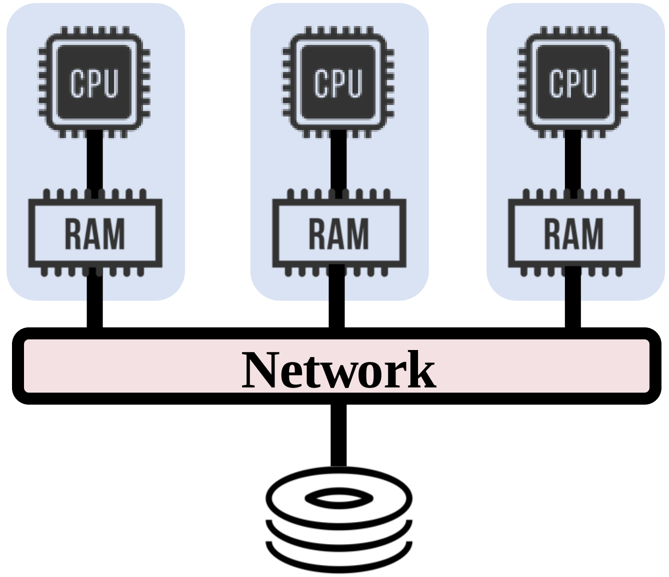
Distributed database

– Multiple nodes connected using network
– Communication cost can be significant

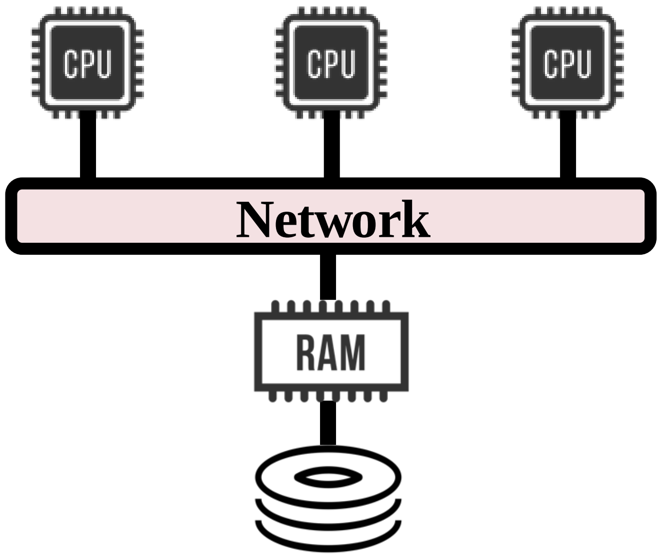
Distributed Architecture



Shared Nothing

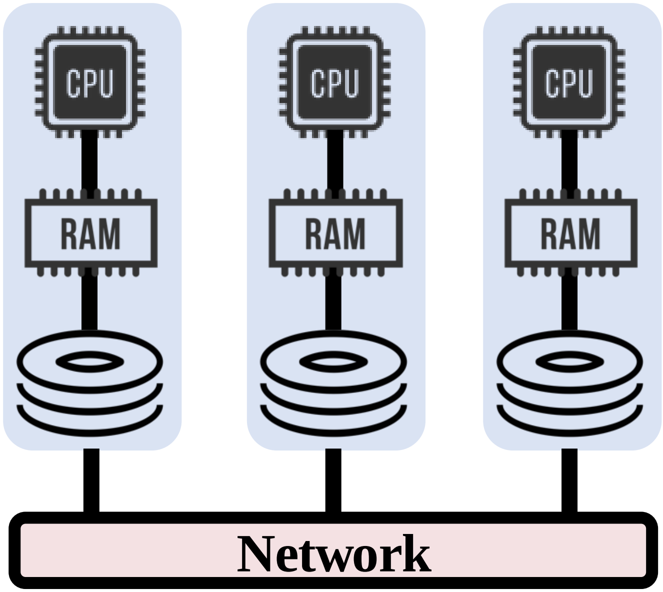


Shared Disk



Shared Memory

Distributed Architecture



Shared Nothing

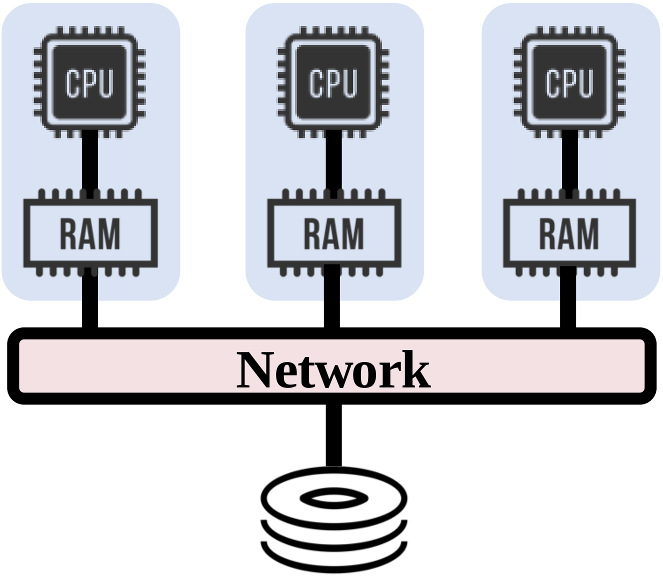
Each memory and disk is owned by some processor that acts as a server for that data

- Scales to **thousands of servers and beyond**

Important optimization goal: minimize network data transfer

This lecture focuses on shared-nothing

Distributed Architecture



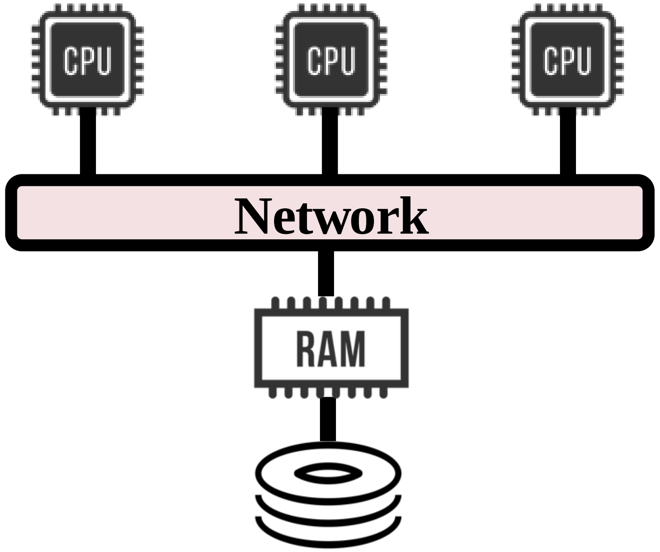
Shared Disk

Each processor has a private memory but has direct access to all disks

- Scale to **tens of servers**

Example: Network attached storage (NAS) and storage area network (SAN)

Distributed Architecture



Shared Memory

All processors share direct access to a common global memory and to all disks

- Does not scale beyond **a single server**

Example: multicore processors

Outline

Distributed architecture

Data partitioning

Query execution

- Merge and Split operators

Join execution

- Co-partitioned
- Partition-based
- Broadcast-based

Parallel SQL Database

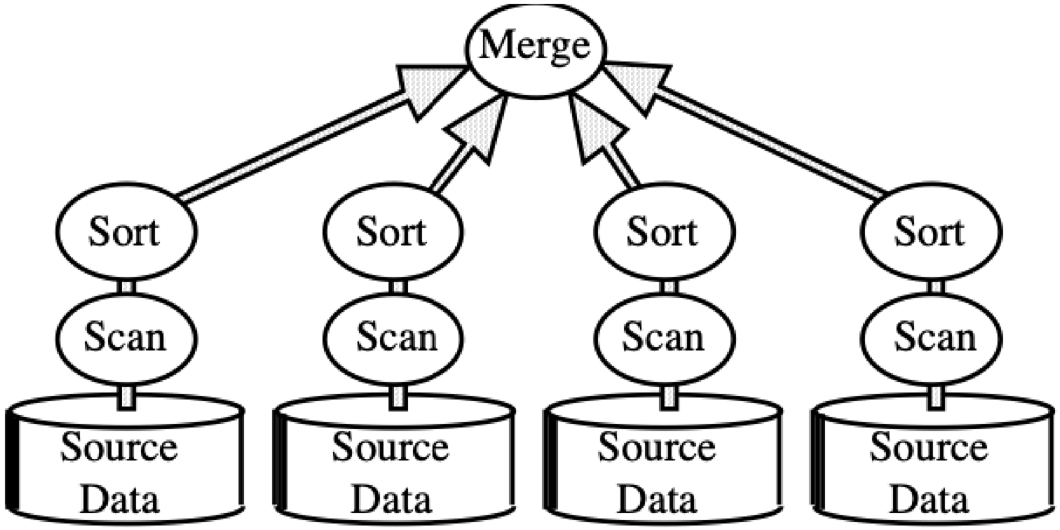
Most database programs are written in relational language SQL

- **Goal: make SQL work on distributed system without rewriting**
- Benefits of a high-level programming interface

Parallel SQL Database

Most database programs are written in relational language SQL

- **Goal: make SQL work on distributed system without rewriting**
- Benefits of a high-level programming interface



Partitioned Parallelism

Partitioned Parallelism

Partition data across nodes

- Partition a table using the partition key (e.g., a single or multiple columns)
- Each record is mapped to one node

A single operator is executed collectively across multiple nodes

Partitioned Parallelism

Partition data across nodes

- Partition a table using the partition key (e.g., a single or multiple columns)
- Each record is mapped to one node

A single operator is executed collectively across multiple nodes

key≤300 300<key≤700 700<key

Range partitioning

Partitioned Parallelism

Partition data across nodes

– Partition a table using the partition key (e.g., a single or multiple columns)

– Each record is mapped to one node

A single operator is executed collectively across multiple nodes

key≤300 300<key≤700 700<key

Range partitioning

hash(key)=0 hash(key)=1 hash(key)=2

Hash partitioning

Outline

Distributed architecture

Data partitioning

Query execution

- Merge and Split operators

Join execution

- Co-partitioned
- Partition-based
- Broadcast-based

Distributed Relational Operators

Split a relation into smaller relations

Merge multiple small relations into one bigger relation

Distributed Relational Operators

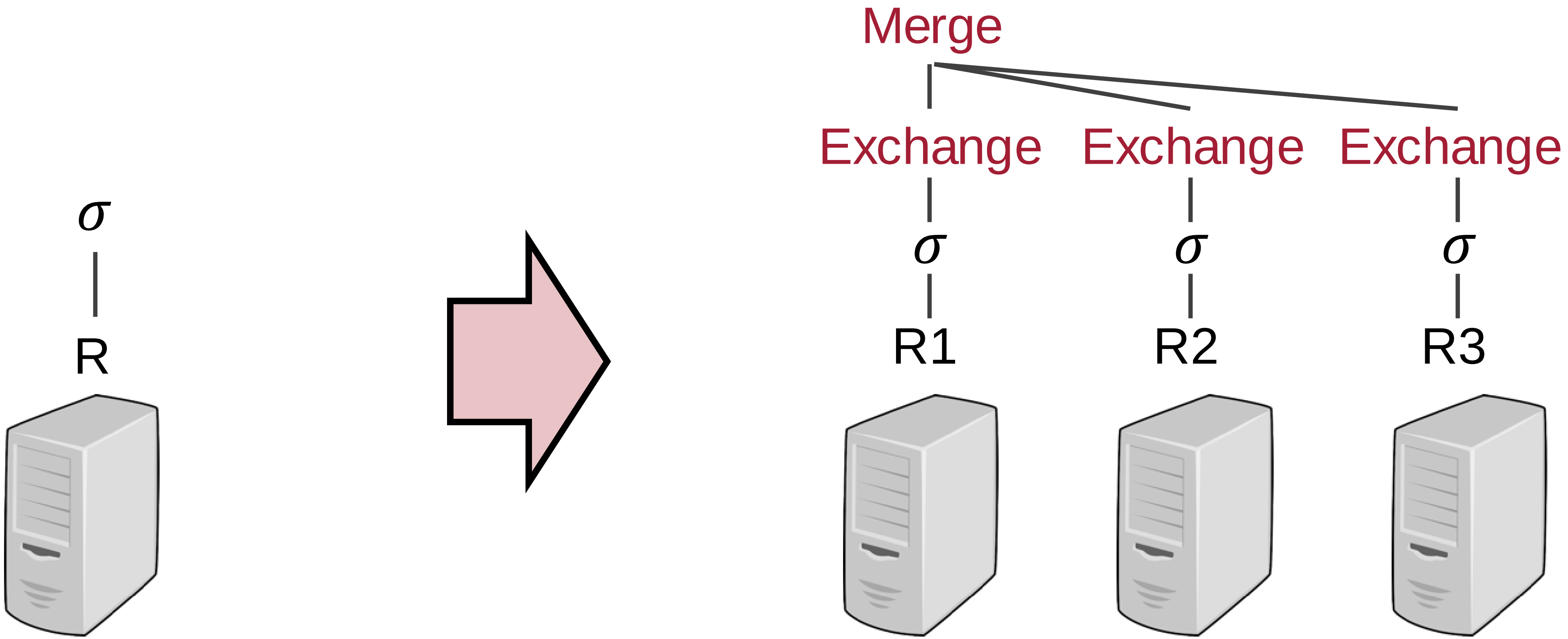
Split a relation into smaller relations

Merge multiple small relations into one bigger relation

Exchange: move relations between distributed nodes

- **Broadcast**: Send the source relation to all the nodes
- **Shuffle**: Split the relation and send each partition to the corresponding node

Distributed Selection



Step 1: Each partition is filtered locally

Step 2: Filtered partitions are exchanged and merged in one node

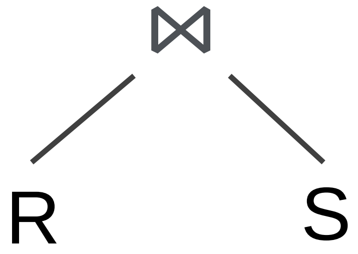
Outline

- Distributed architecture
- Data partitioning
- Query execution
  - Merge and Split operators
- **Join execution**
  - Co-partitioned
  - Partition-based
  - Broadcast-based

Distributed Join—Co-Partitioned

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



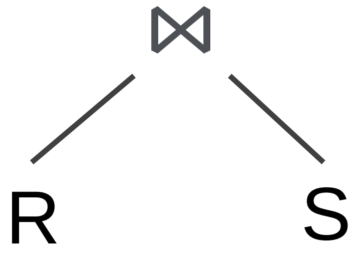
Distributed Join—Co-Partitioned

R (A, B), S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```

Assume **R** and **S** are co-partitioned

– Partitioned on the join key using the **same partitioning function**



R1	S1	R2	S2	R3	S3
{B≤100}		*{100<B<200}*		*{200<B}*	

For example, rows in **R1** cannot possibly join with rows in **S2**

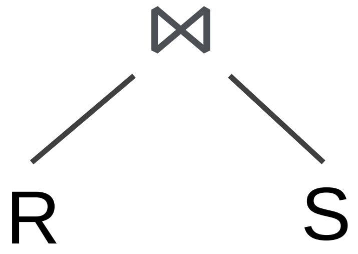
Distributed Join—Co-Partitioned

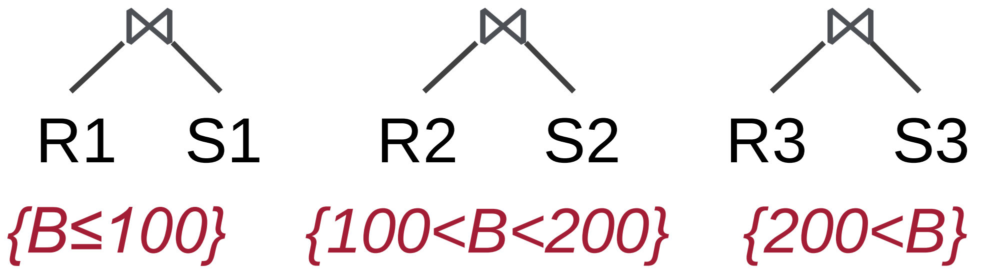
R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```

Assume **R** and **S** are co-partitioned

- Partitioned on the join key using the **same partitioning function**



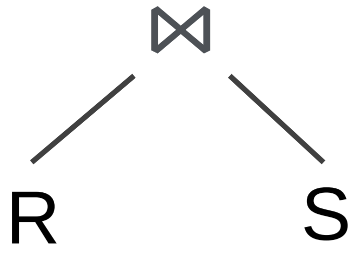


Join local partitions of R and S (i.e., R1⋈S1, R2⋈S2, R3⋈S3)

Distributed Join—Co-Partitioned

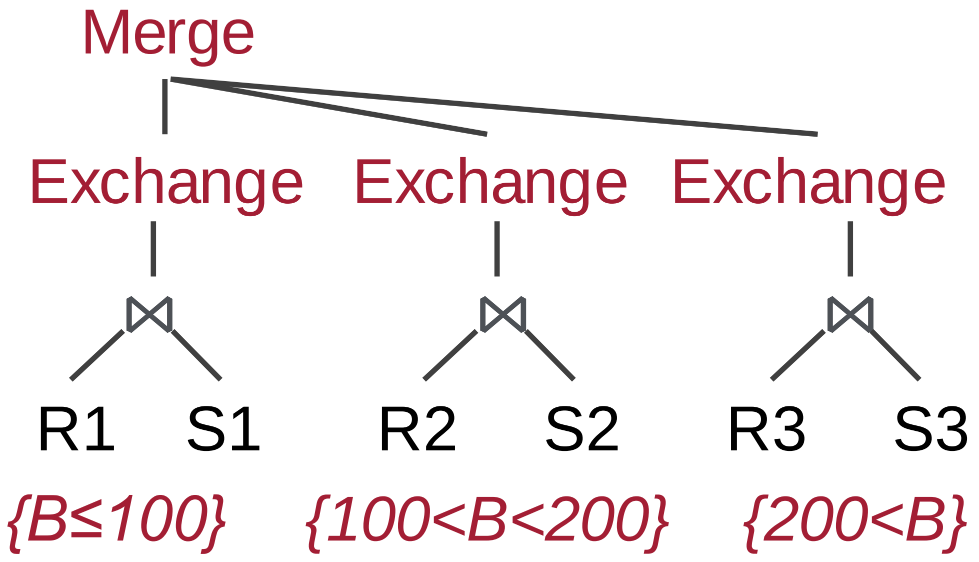
R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



Assume **R** and **S** are co-partitioned

- Partitioned on the join key using the **same partitioning function**



Join local partitions of R and S (i.e., R1⋈S1, R2⋈S2, R3⋈S3)

Send join outputs to a single node and merge

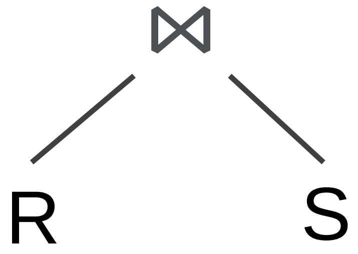
Distributed Join—Partition-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```

Relation R is range-partitioned on column **B**

Relation S is range-partitioned on column **C**



{B≤100}	*{100<B≤200}*	*{200<B}*
R1 S1	R2 S2	R3 S3
{C≤20}	*{20<C≤40}*	*{40<C}*

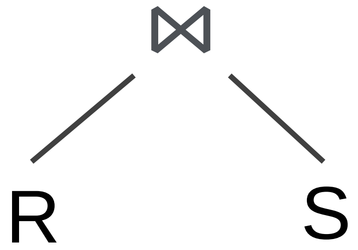
Distributed Join—Partition-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```

Relation R is range-partitioned on column **B**

Relation S is range-partitioned on column **C**



$\{B \le 100\}$ *$\{100 < B \le 200\}$*   *$\{200 < B\}$*

R1 S1   R2 S2   R3 S3

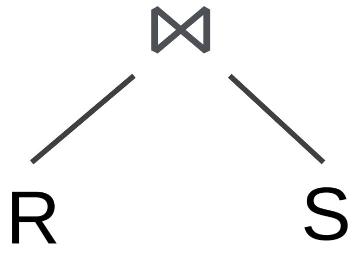
$\{C \le 20\}$ *$\{20 < C \le 40\}$*   *$\{40 < C\}$*

Re-partition relation S on column B

Distributed Join—Partition-Based

R (A, B), S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



Split

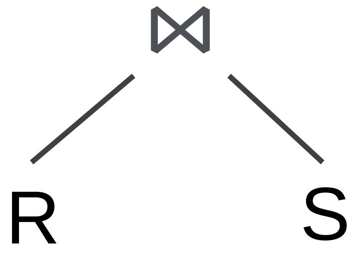
S1	S2	S3
{C≤20}	*{20<C≤40}*	*{40<C}*

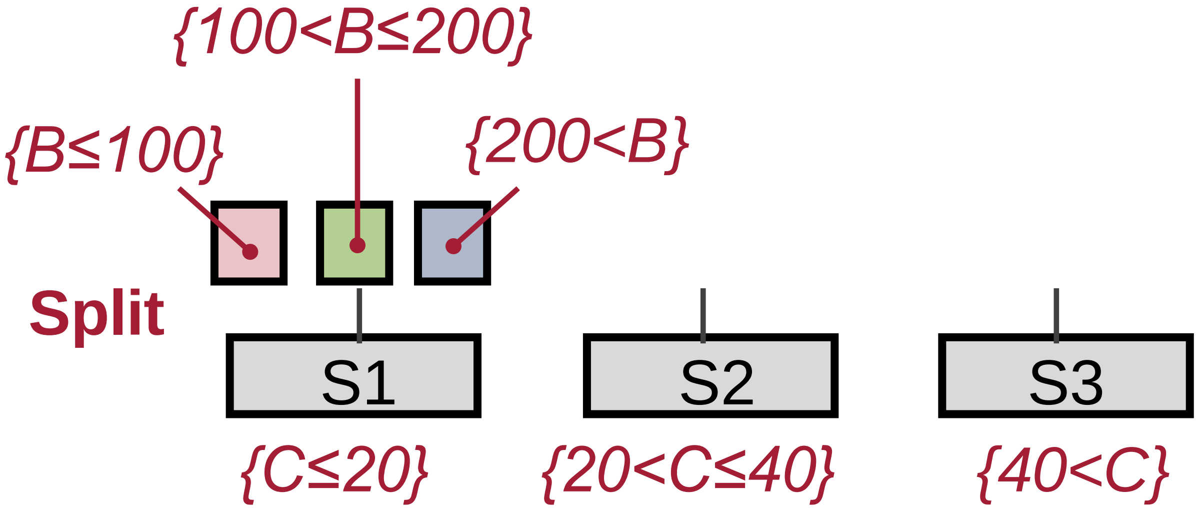
Re-partition relation S on column B

Distributed Join—Partition-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



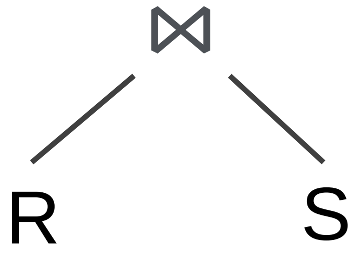


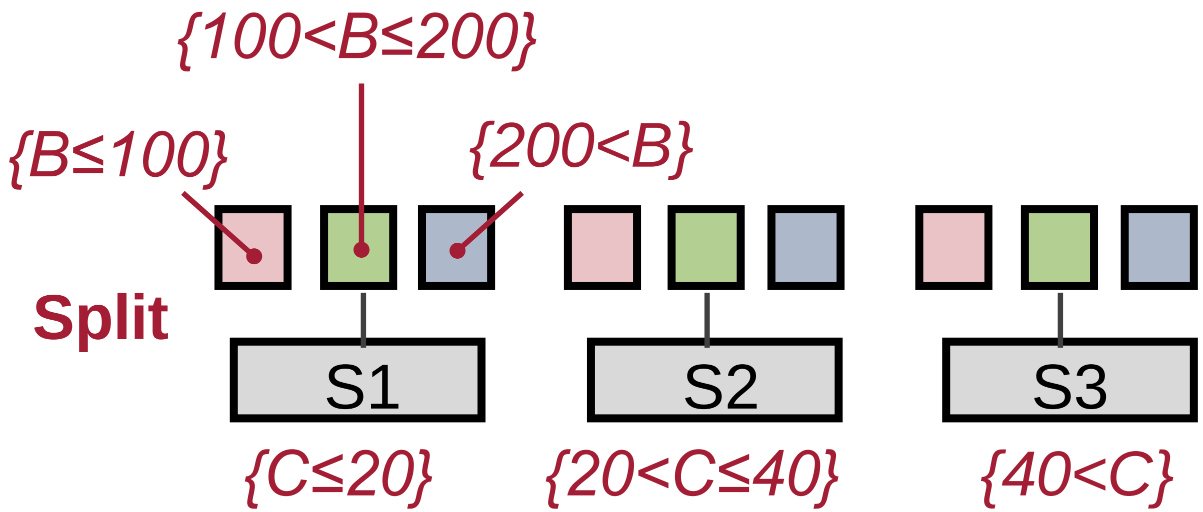
Re-partition relation S on column B

Distributed Join—Partition-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



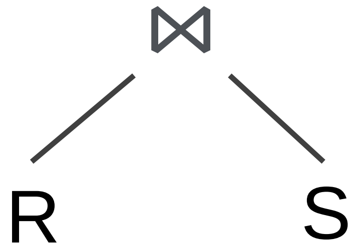


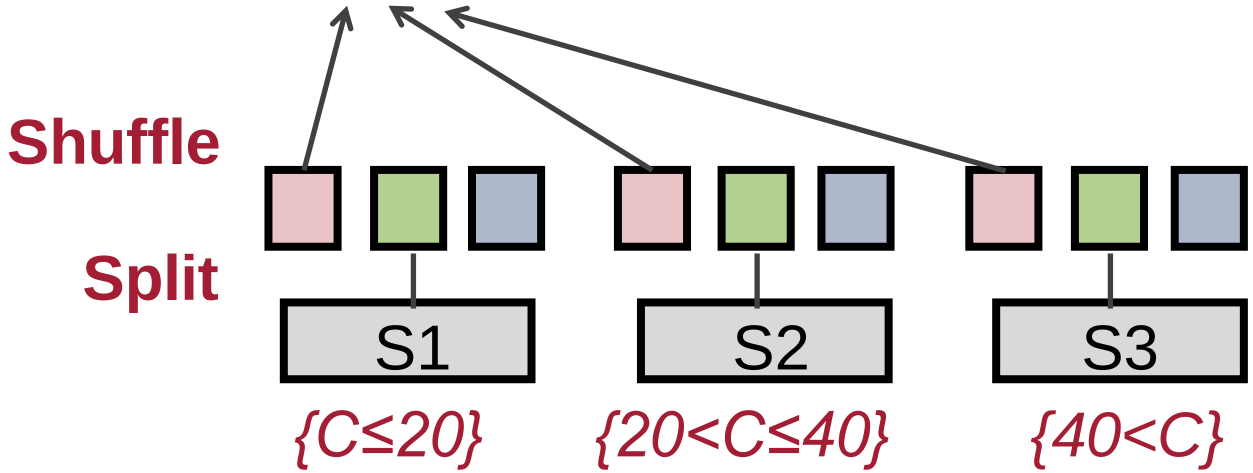
Re-partition relation S on column B

Distributed Join—Partition-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



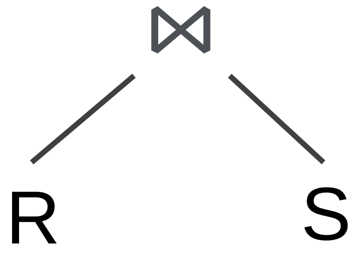


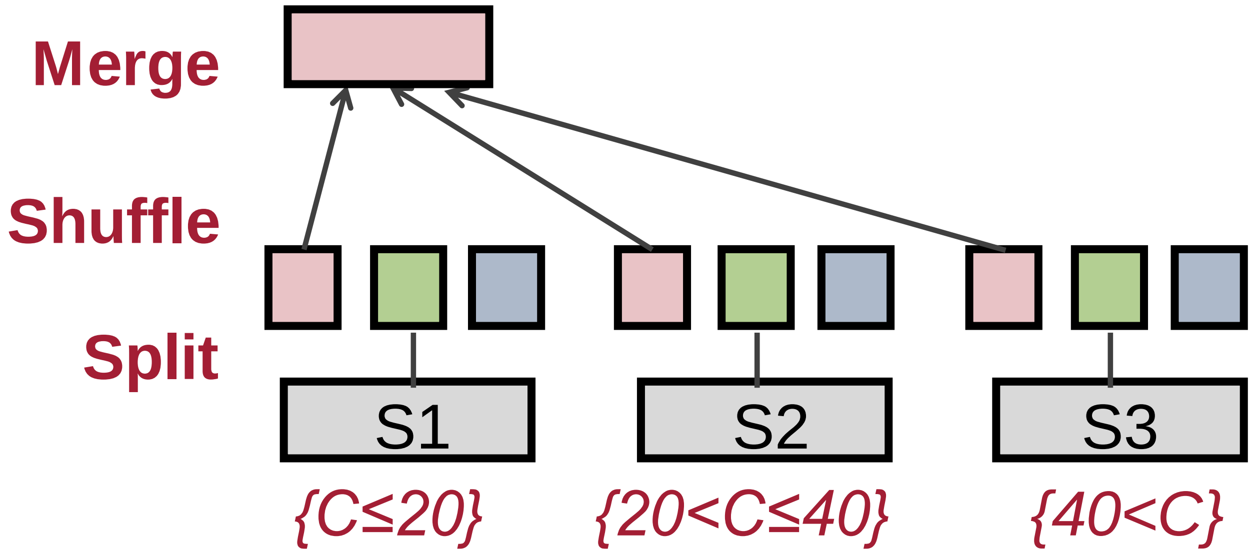
Re-partition relation S on column B

Distributed Join—Partition-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



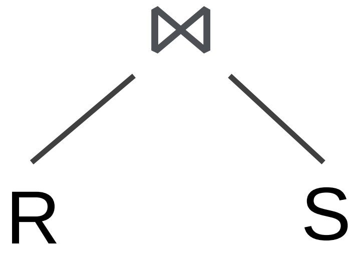


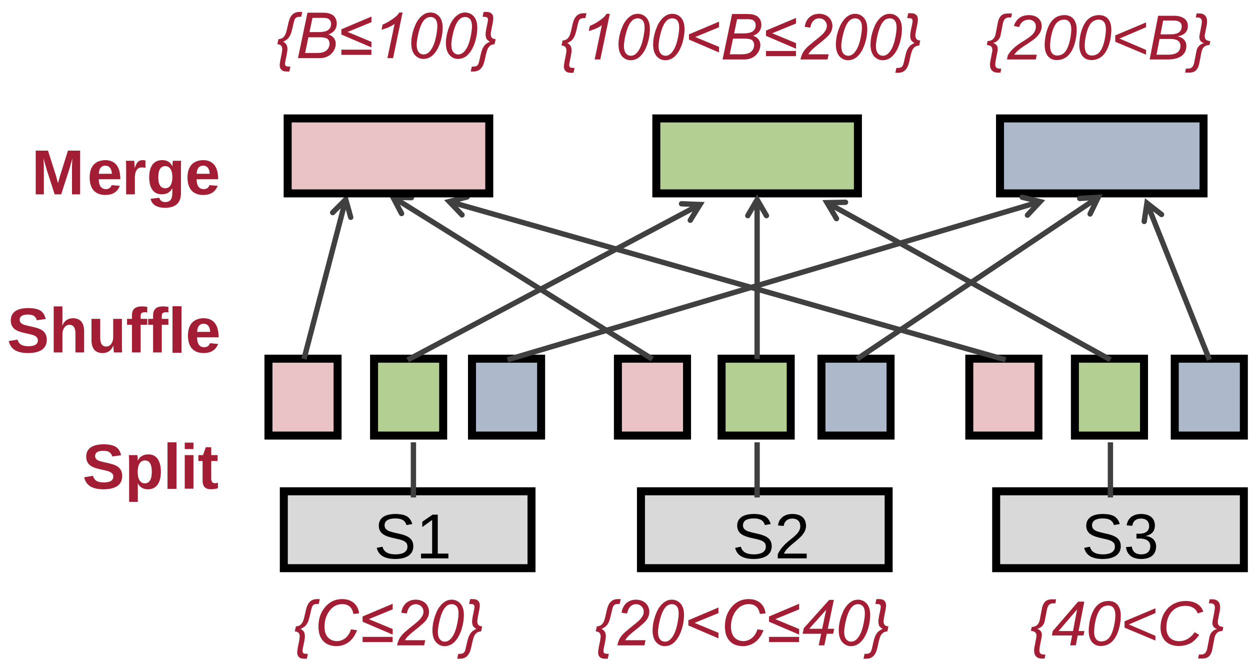
Re-partition relation S on column B

Distributed Join—Partition-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



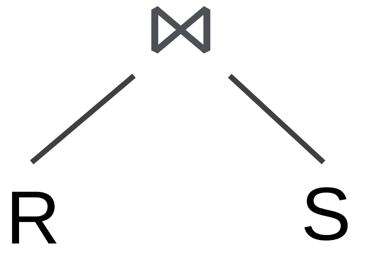


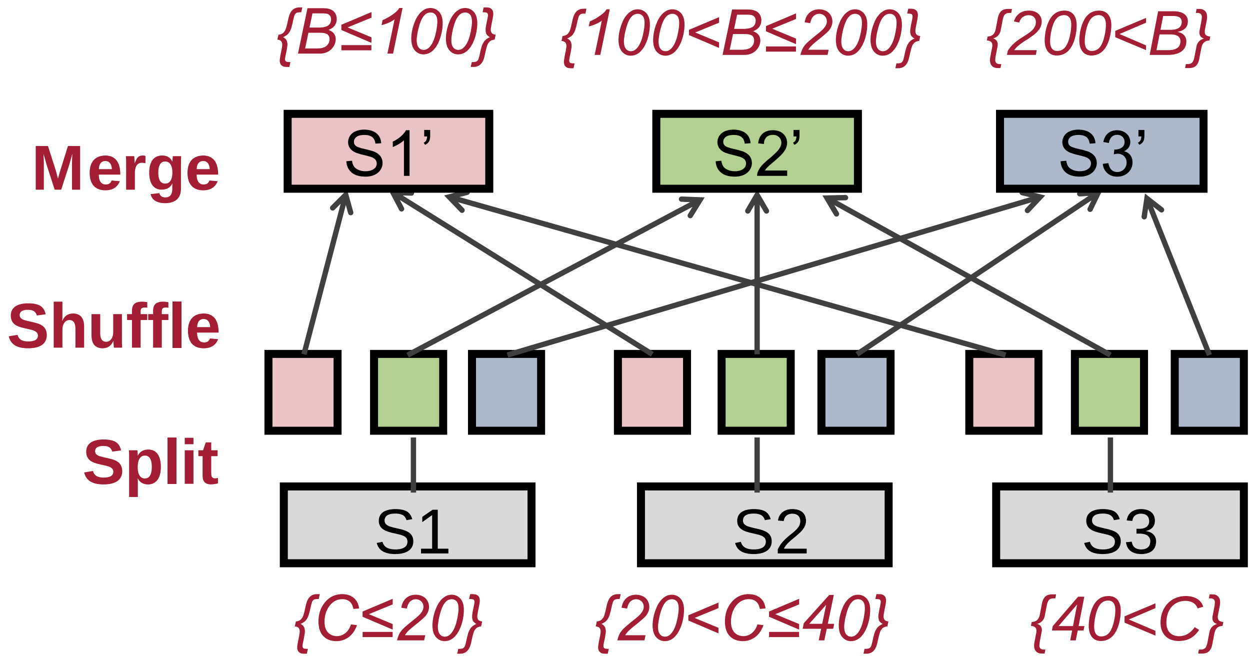
Re-partition relation S on column B

Distributed Join—Partition-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



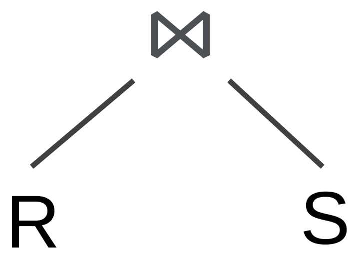


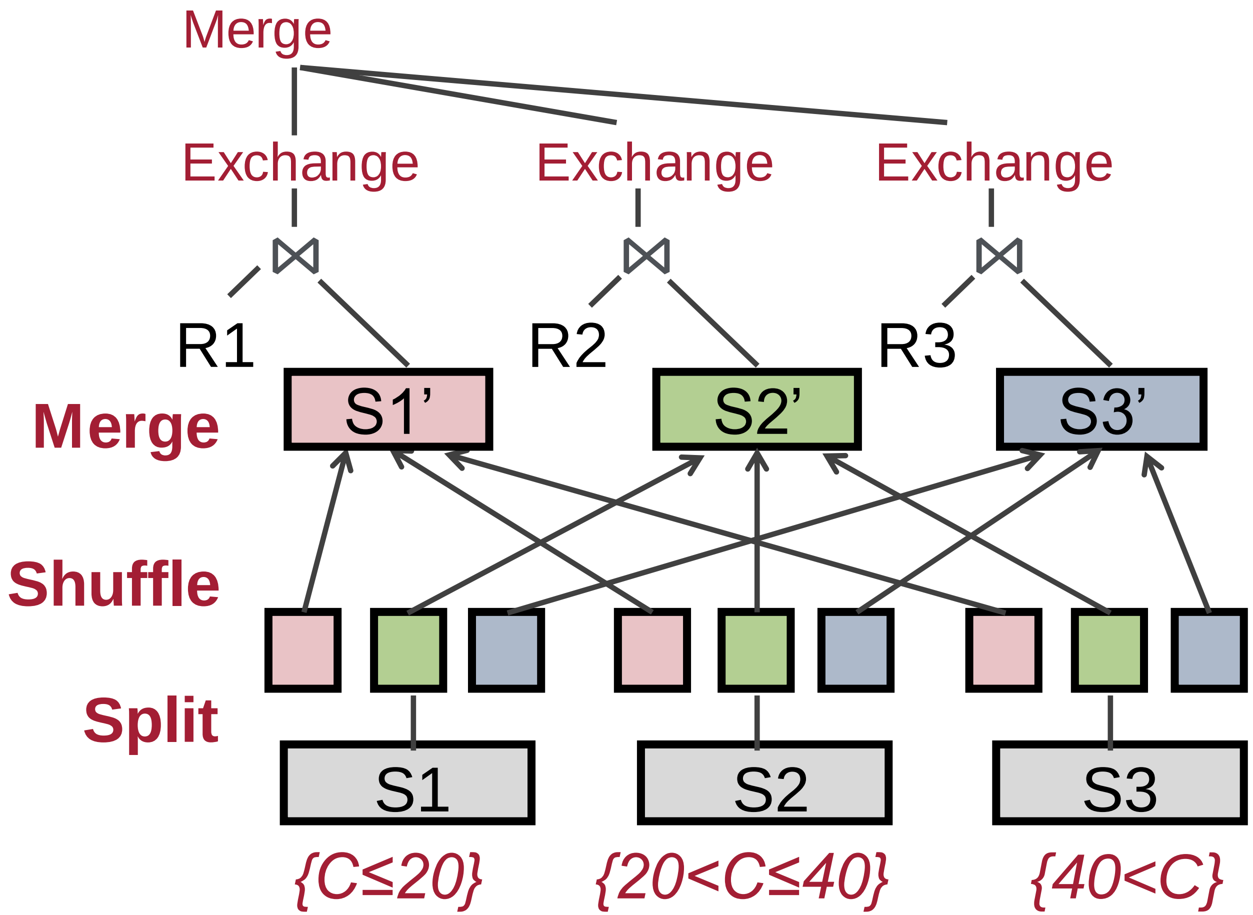
Now both R and S are co-partitioned on B

Distributed Join—Partition-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```





Now both R and S are co-partitioned on B

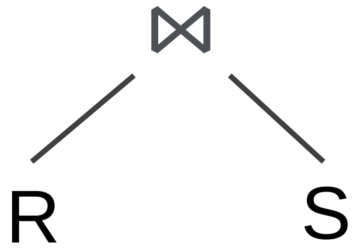
Use the co-partitioned join algorithm to finish the query

Distributed Join—Partition-Based

R (A, B), S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```

What if neither **R** nor **S** is partitioned on the join key?



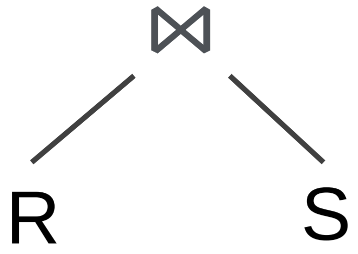
$\{H(A)=0\}$	$\{H(A)=1\}$	$\{H(A)=2\}$
R1 S1	R2 S2	R3 S3
$\{C \leq 20\}$	$\{20 < C \leq 40\}$	$\{40 < C\}$

Distributed Join—Partition-Based

R (A, B), S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```

What if neither **R** nor **S** is partitioned on the join key?



{H(A)=0} *{H(A)=1}* *{H(A)=2}*

R1 S1 R2 S2 R3 S3

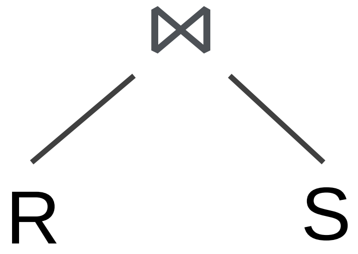
{C≤20} *{20<C≤40}* *{40<C}*

Re-partition both R and S on column B **using the same partition function!**

Distributed Join—Broadcast-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



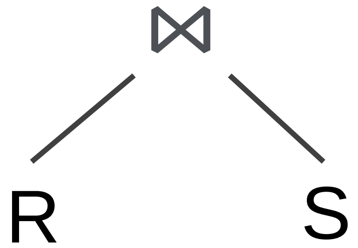
{H(A)=0}	*{H(A)=1}*	*{H(A)=2}*
R1 S1	R2 S2	R3 S3
{C≤20}	*{20<C≤40}*	*{40<C}*

If R and S are not co-partitioned, another solution is to **broadcast one relation to all nodes**

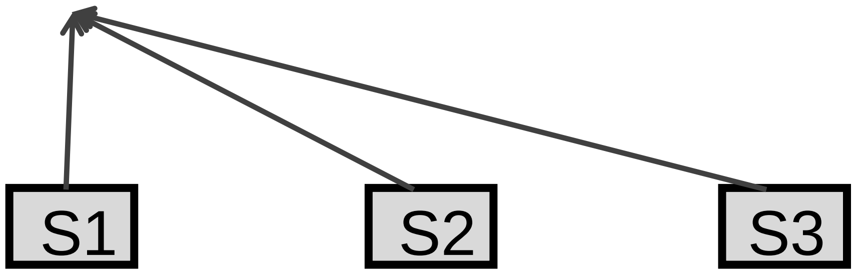
Distributed Join—Broadcast-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



Broadcast

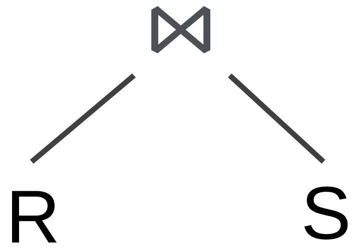


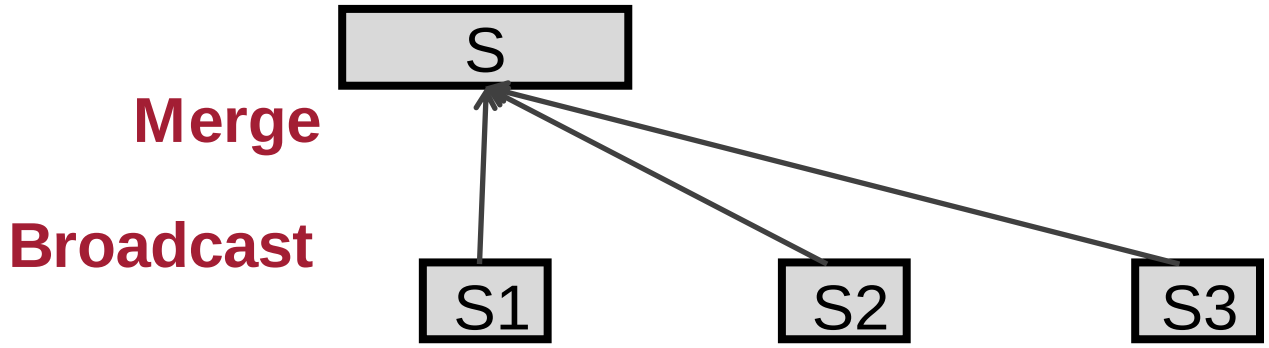
If R and S are not co-partitioned, another solution is to **broadcast one relation to all nodes**

Distributed Join—Broadcast-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



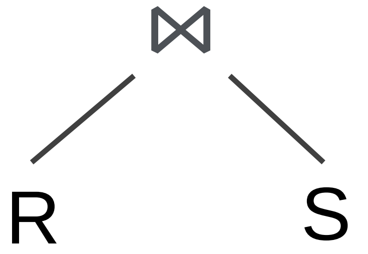


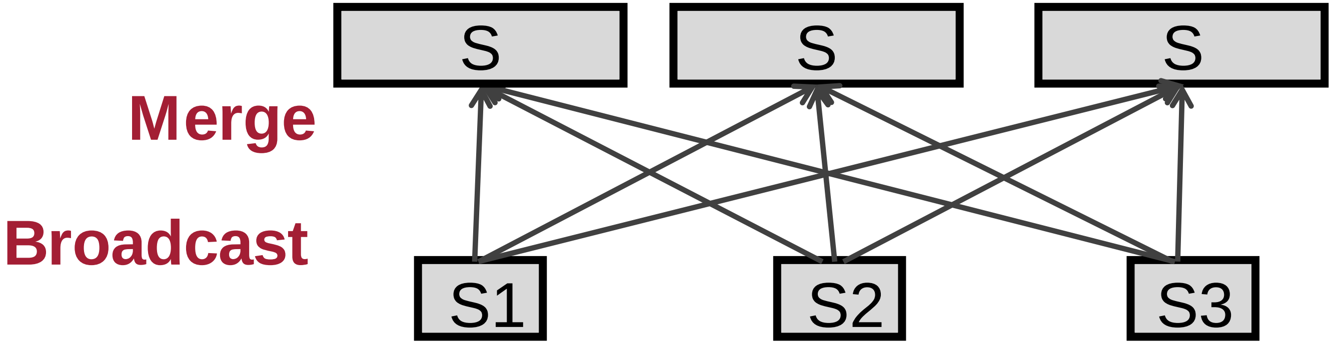
If R and S are not co-partitioned, another solution is to **broadcast one relation to all nodes**

Distributed Join—Broadcast-Based

R (A, B),   S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```



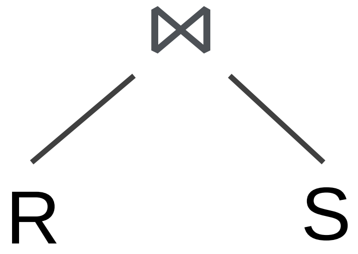


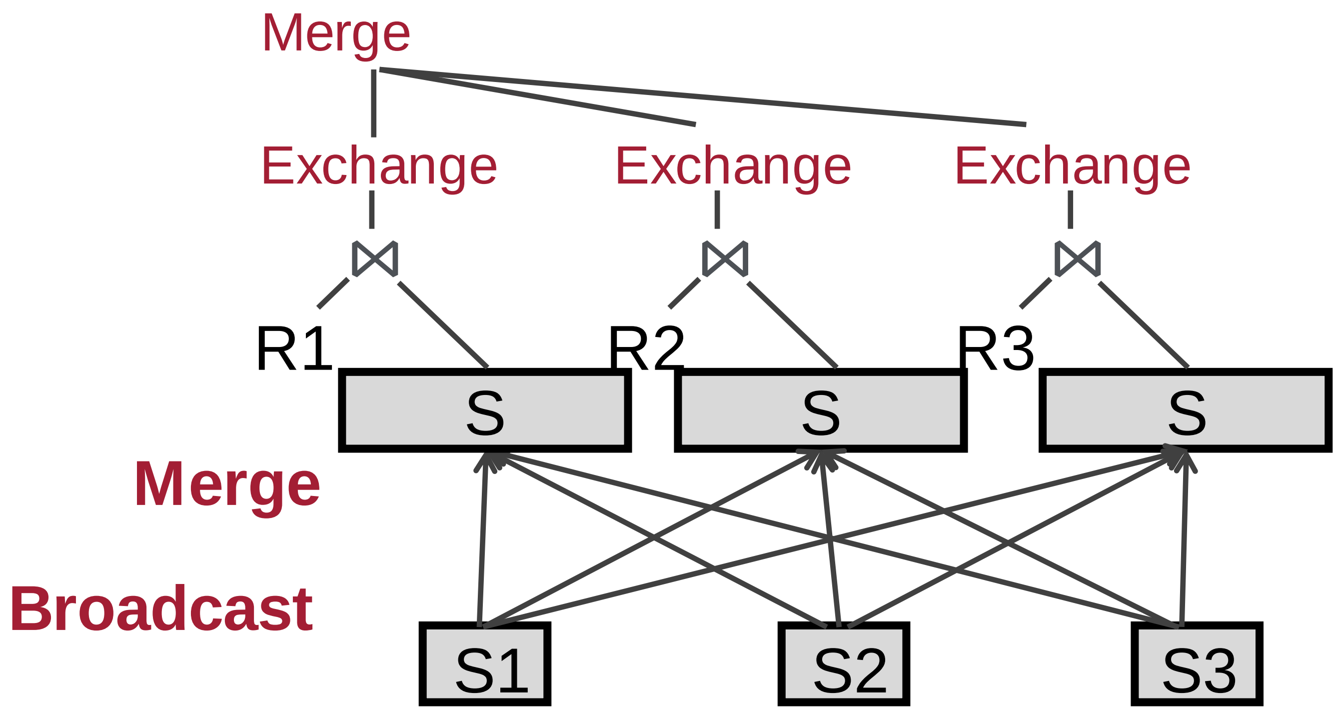
If R and S are not co-partitioned, another solution is to **broadcast one relation to all nodes**

Distributed Join—Broadcast-Based

R (A, B), S (B, C)

```
SELECT *
FROM R, S
WHERE R.B = S.B
```





If R and S are not co-partitioned, another solution is to **broadcast one relation to all nodes**

Distributed Join Methods

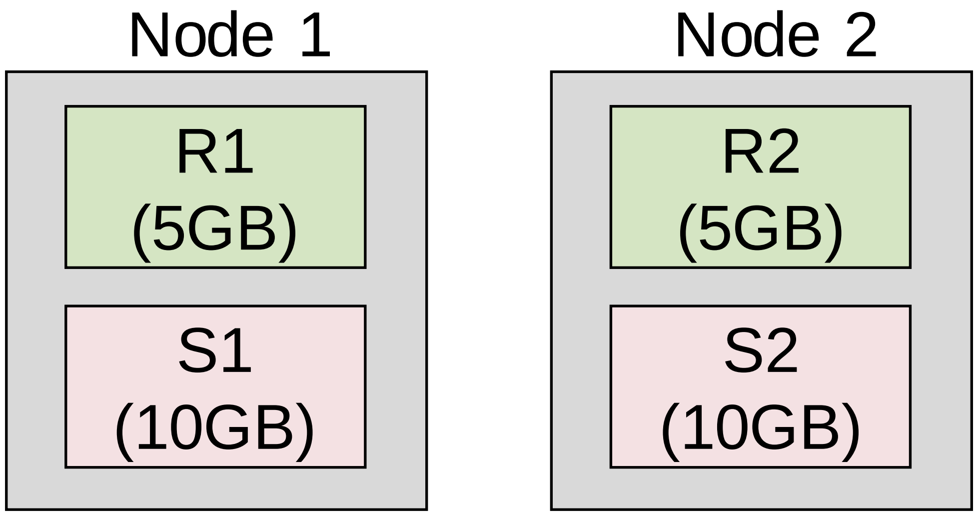
Co-partitioned: Works if the two relations are already co-partitioned on the join key (using the same partitioning function)

Partition-based: Partition one or two relations such that both relations are co-partitioned

Broadcast-based: Broadcast the smaller relation to all nodes

Exercise

We execute R⋈S on two nodes with the data layout shown below. Assume S is partitioned on the join key, but R is not. What is the network IO cost for partition-based join and broadcast-based join, respectively? (we assume uniform distribution; we assume the join result size is negligible)



Outline

Distributed architecture

Data partitioning

Query execution

- Merge and Split operators

Join execution

- Co-partitioned
- Partition-based
- Broadcast-based