

Hive Bucketing in Apache Spark

Tejas Patil

Facebook

Agenda

- Why bucketing ?
- Why is shuffle bad ?
- How to avoid shuffle ?
- When to use bucketing ?
- Spark's bucketing support
- Bucketing semantics of Spark vs Hive
- Hive bucketing support in Spark
- SQL Planner improvements

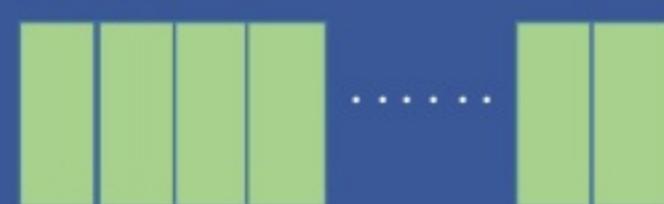
Why bucketing ?

Table A



JOIN

Table B





Broadcast hash join

- Ship smaller table to all nodes
- Stream the other table



Broadcast hash join

- Ship smaller table to all nodes
- Stream the other table

Shuffle hash join

- Shuffle both tables,
- Hash smaller one, stream the bigger one



Broadcast hash join

- Ship smaller table to all nodes
- Stream the other table

Shuffle hash join

- Shuffle both tables,
- Hash smaller one, stream the bigger one

Sort merge join

- Shuffle both tables,
- Sort both tables, buffer one, stream the bigger one



Broadcast hash join

- Ship smaller table to all nodes
- Stream the other table

Shuffle hash join

- Shuffle both tables,
- Hash smaller one, stream the bigger one

Sort merge join

- Shuffle both tables,
- Sort both tables, buffer one, stream the bigger one

Sort Merge Join

Table A

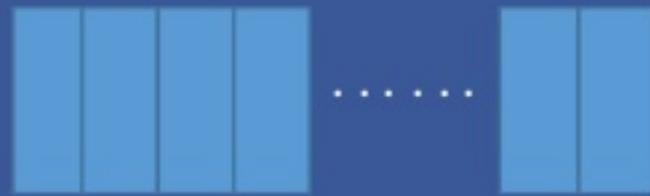
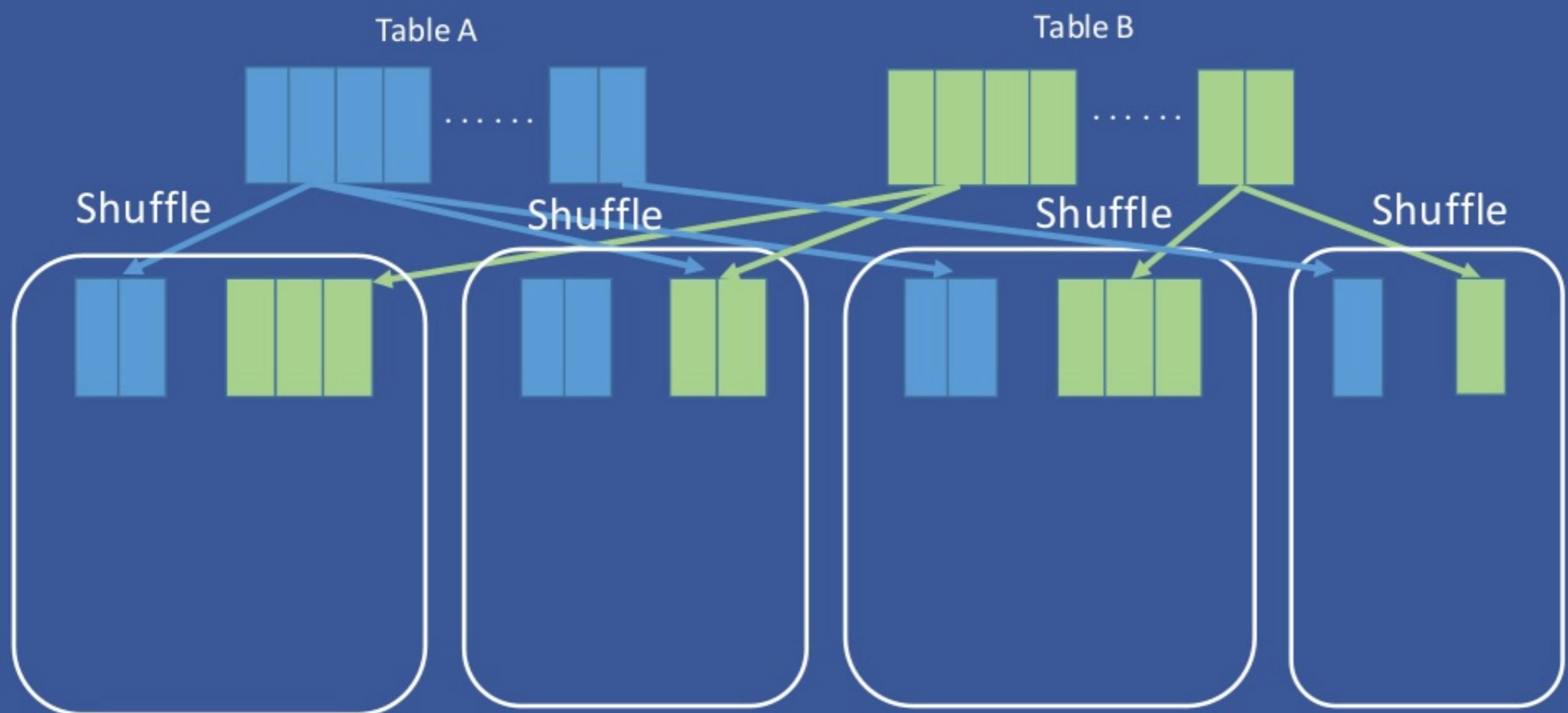


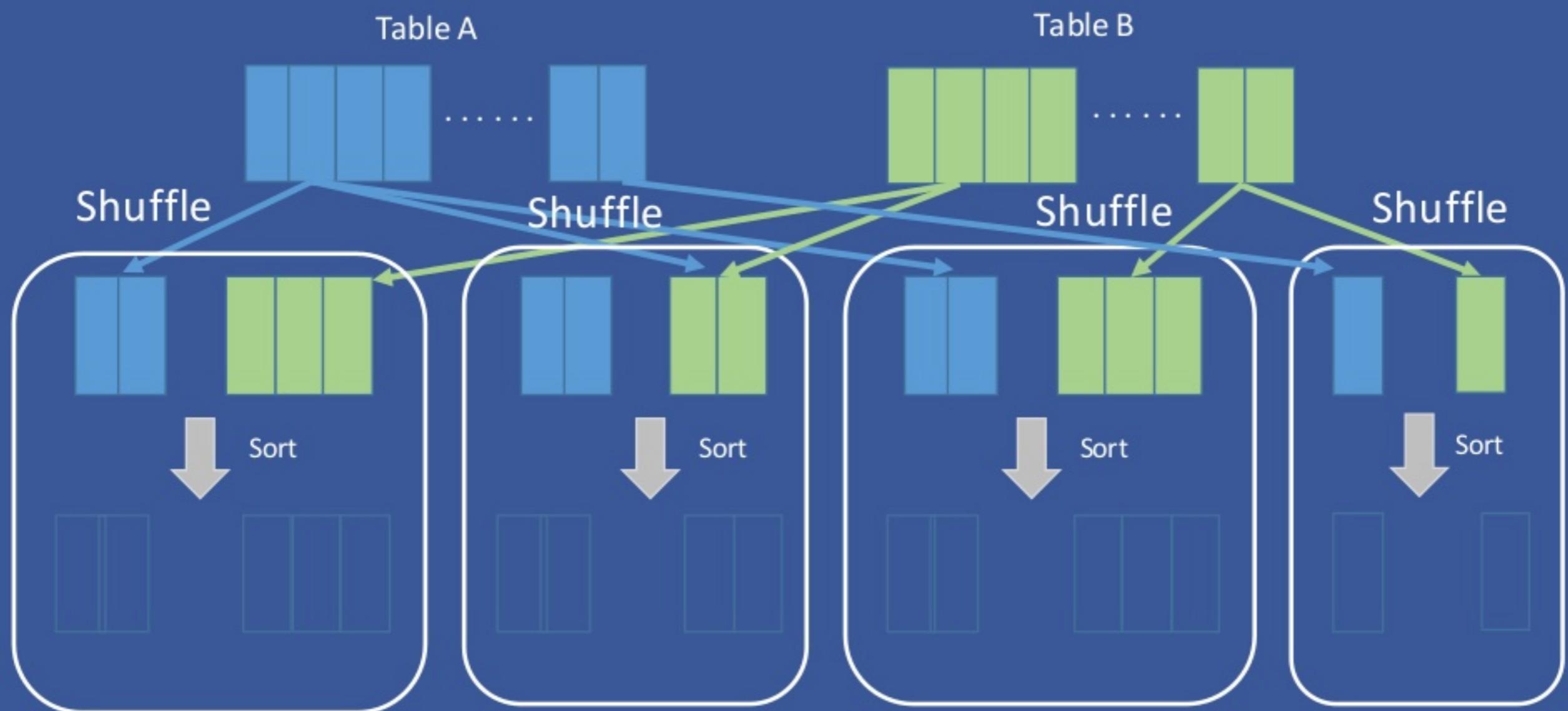
Table B



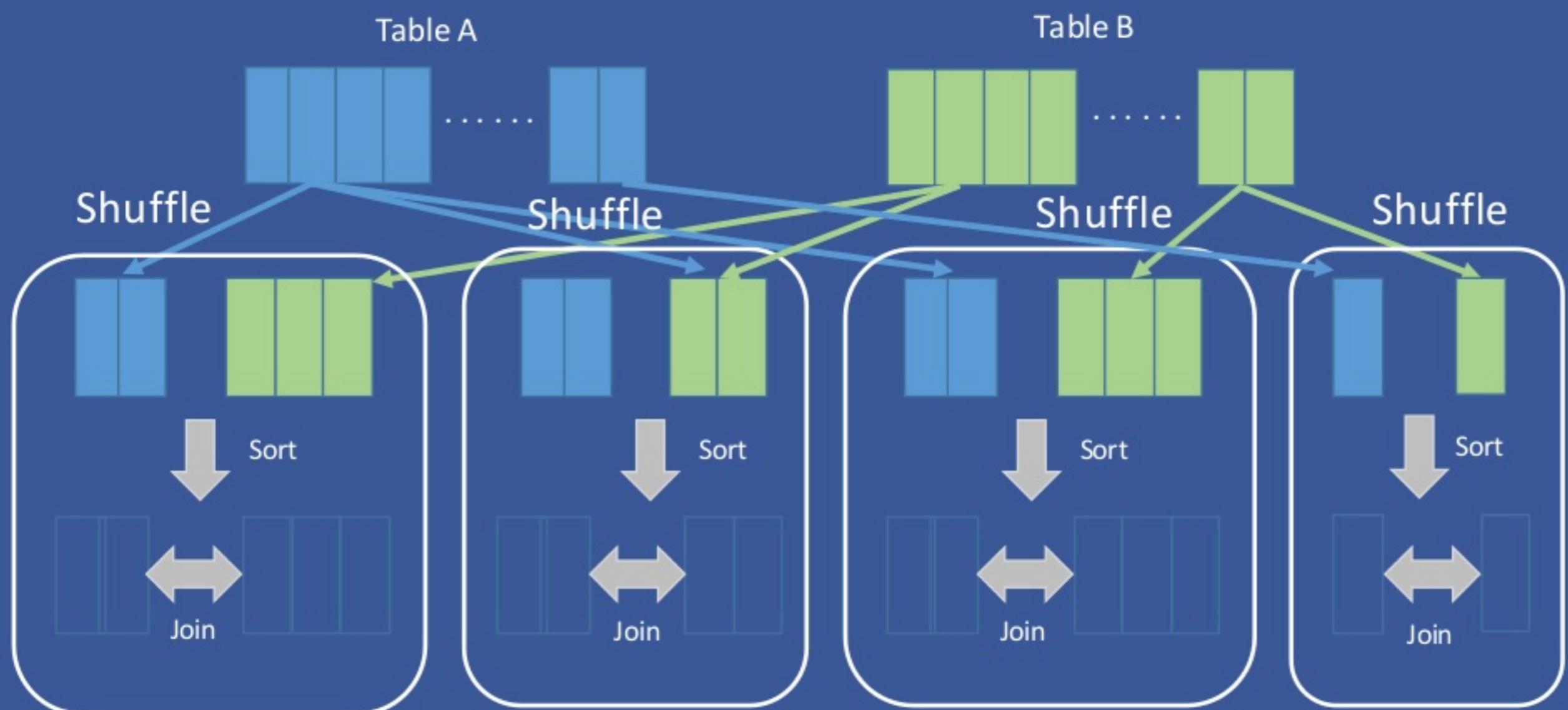
Sort Merge Join



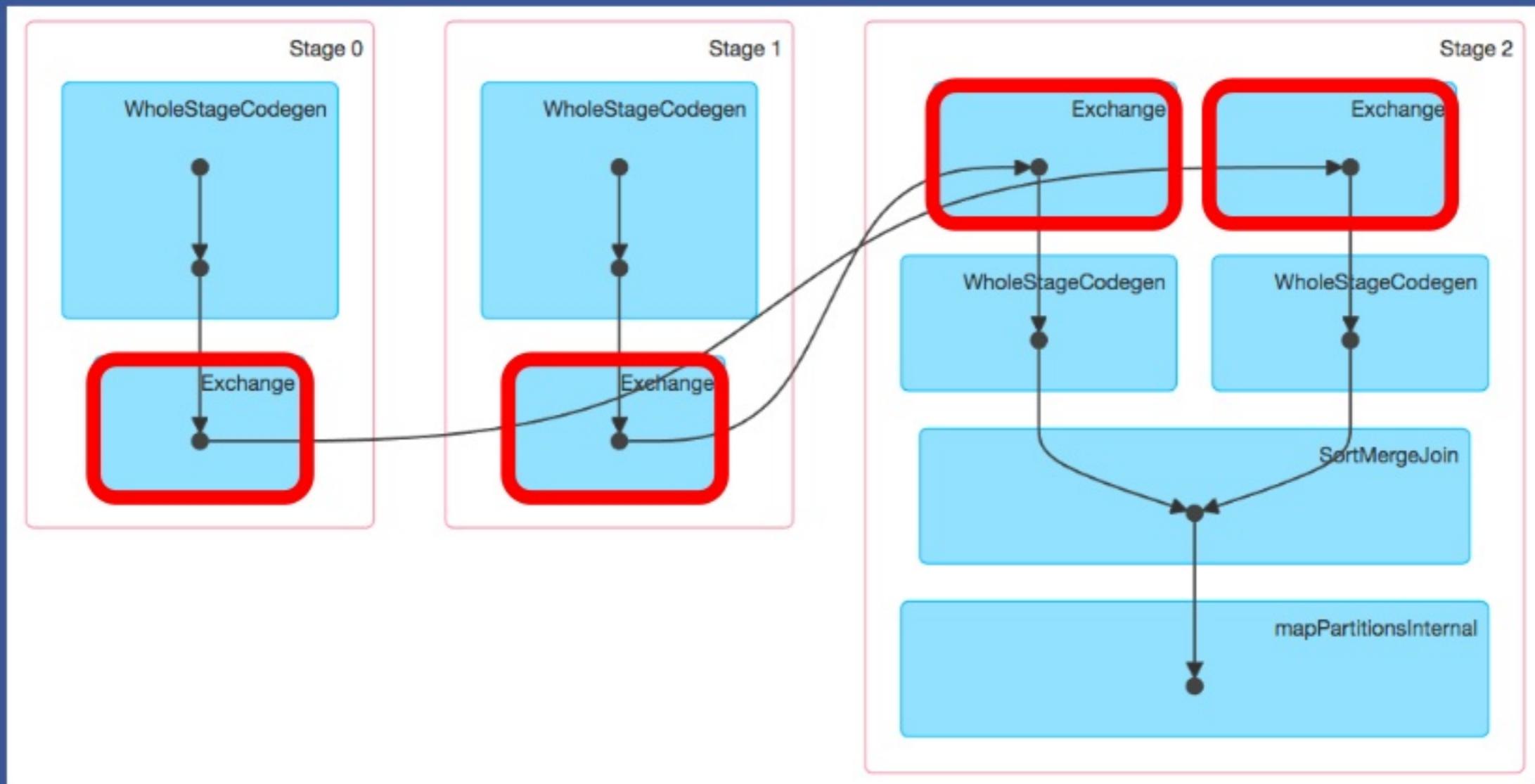
Sort Merge Join



Sort Merge Join

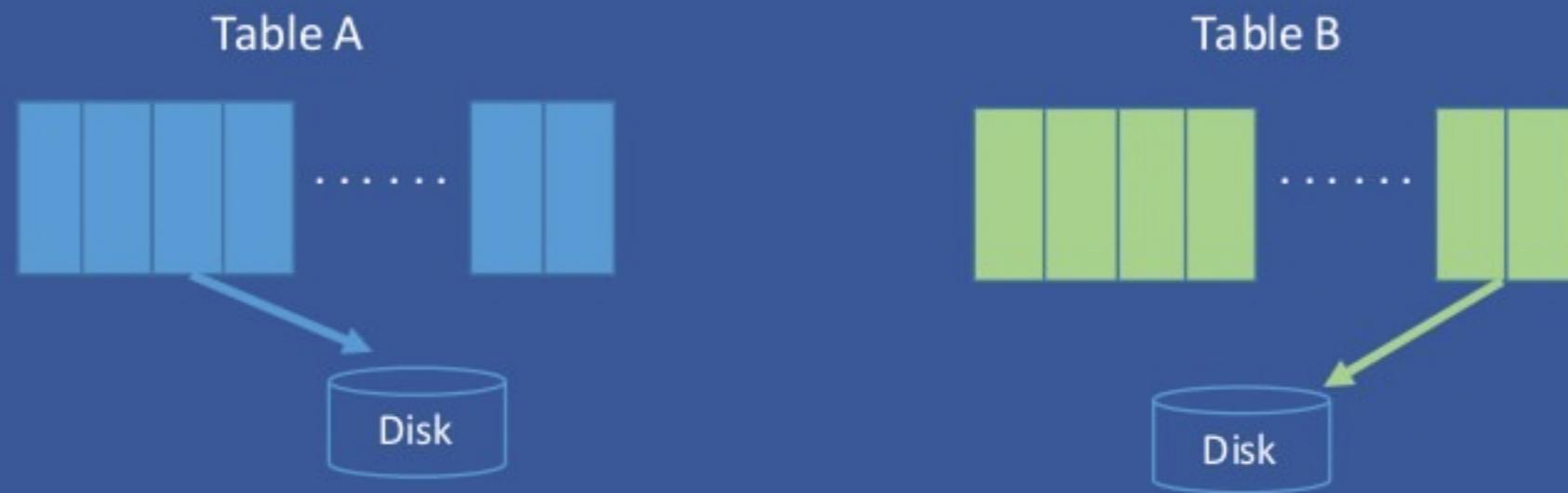


Sort Merge Join



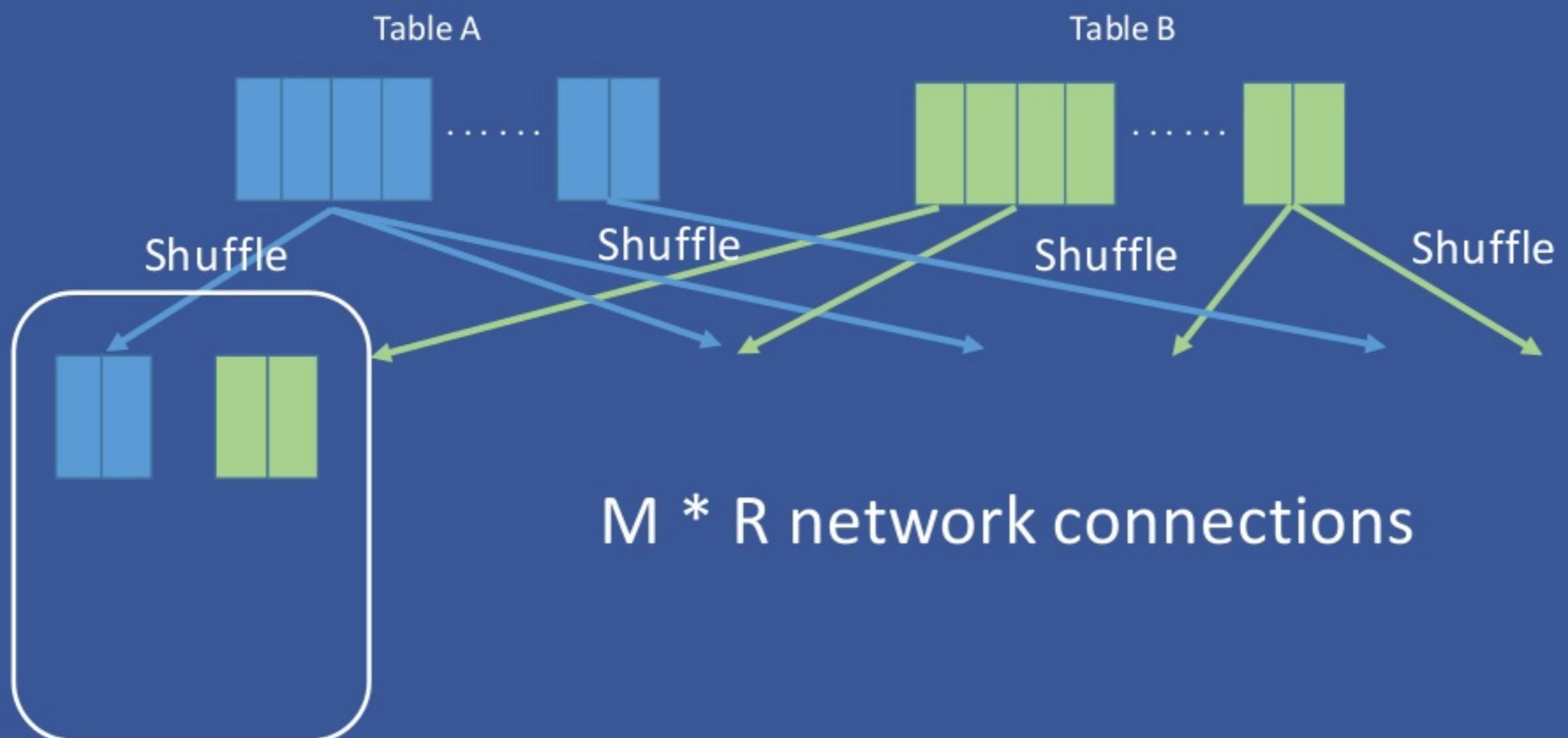
Why is shuffle bad ?

Why is shuffle bad ?

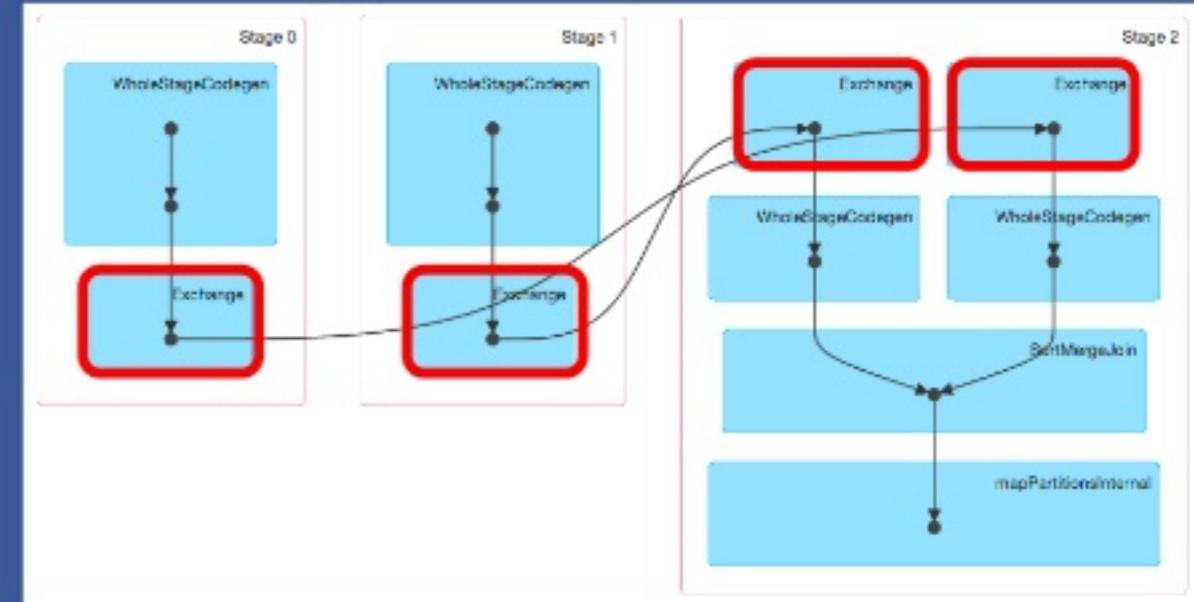
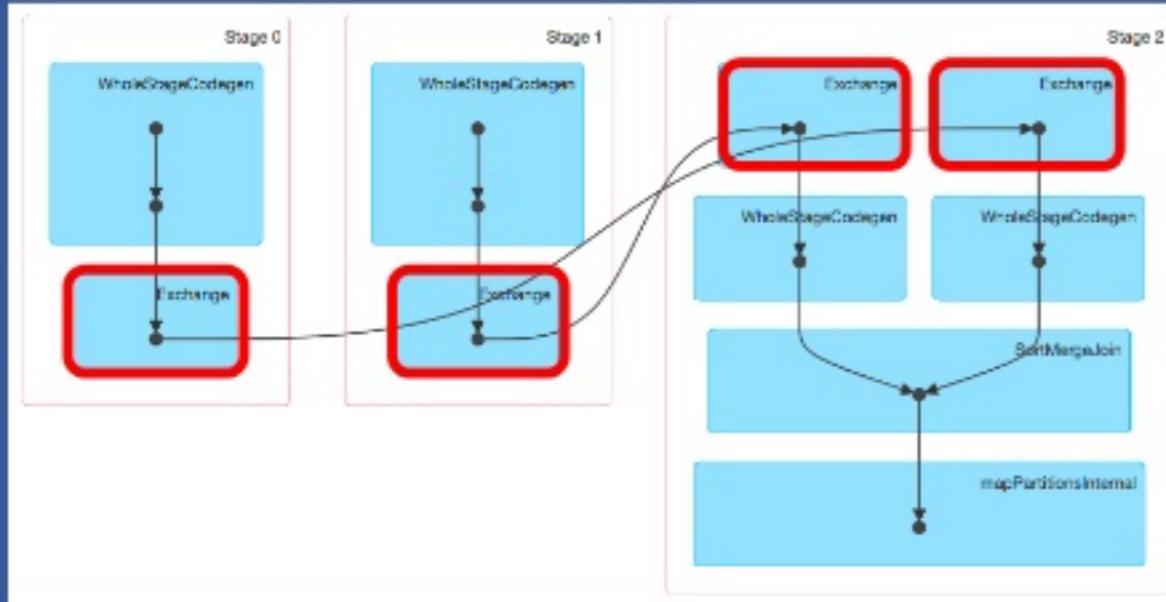
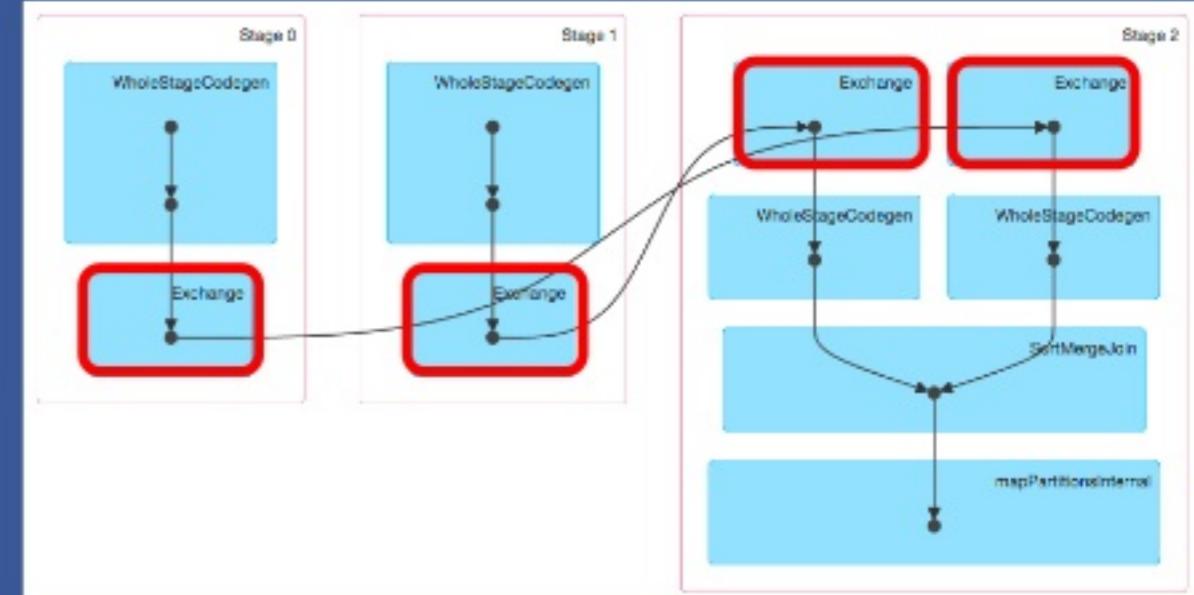
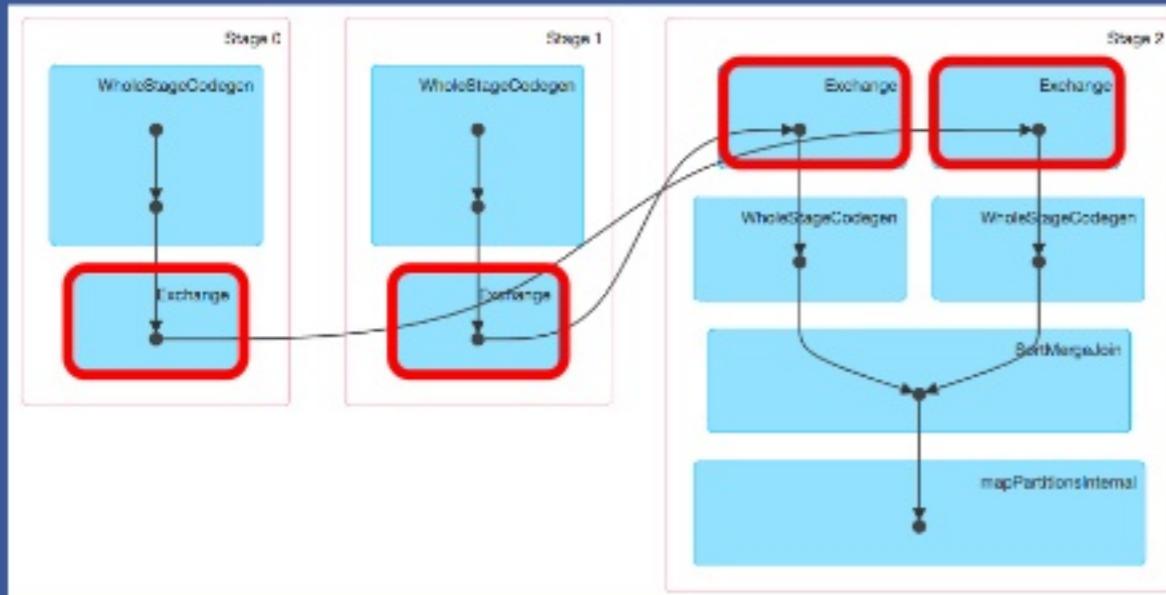


Disk IO for shuffle outputs

Why is shuffle bad ?



How to avoid shuffle ?

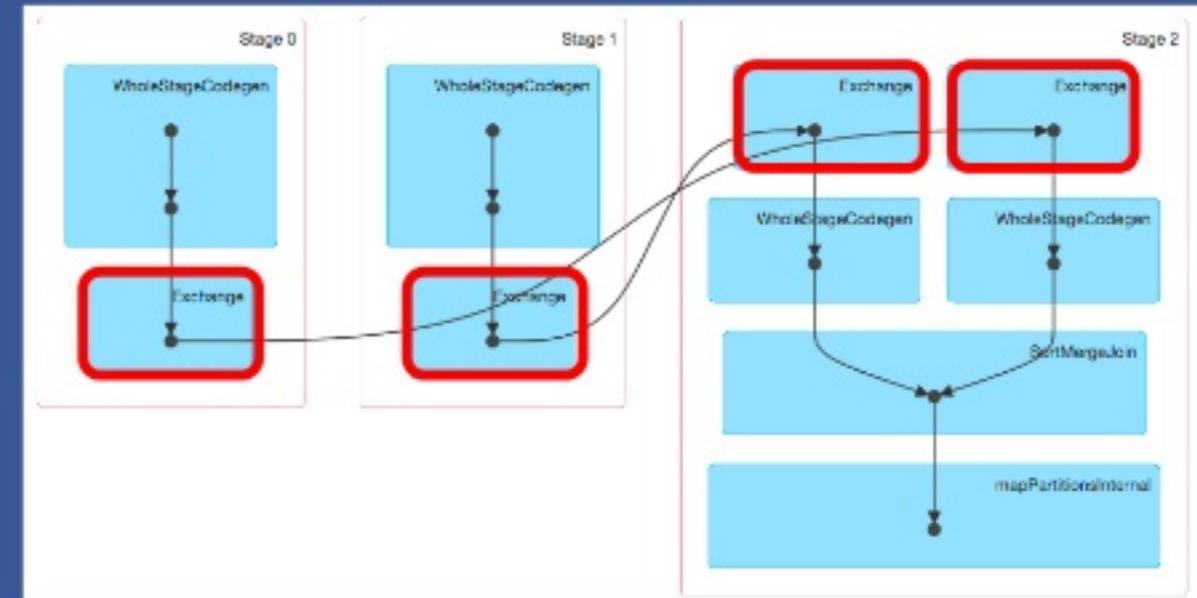
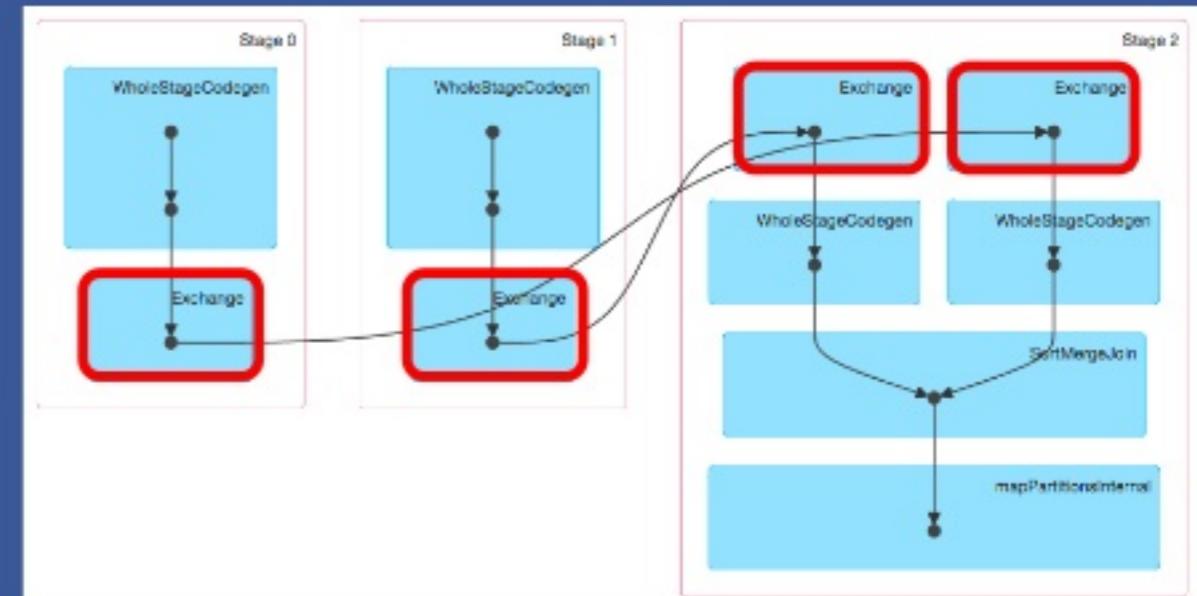


student s JOIN attendance a
ON s.id = a.student_id

student s JOIN results r
ON s.id = r.student_id

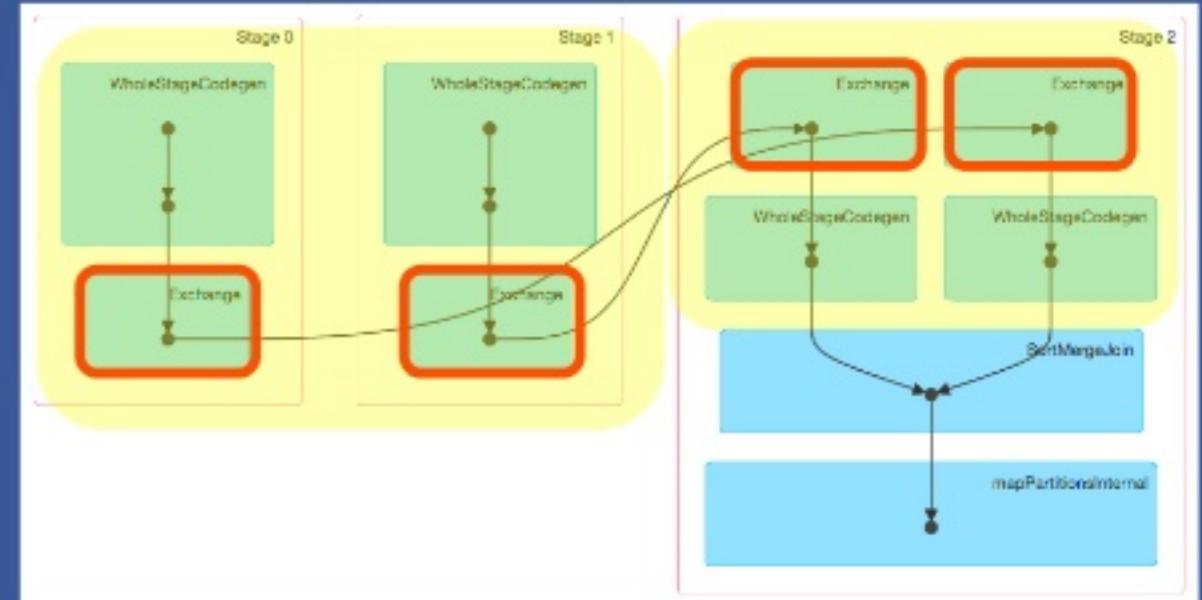
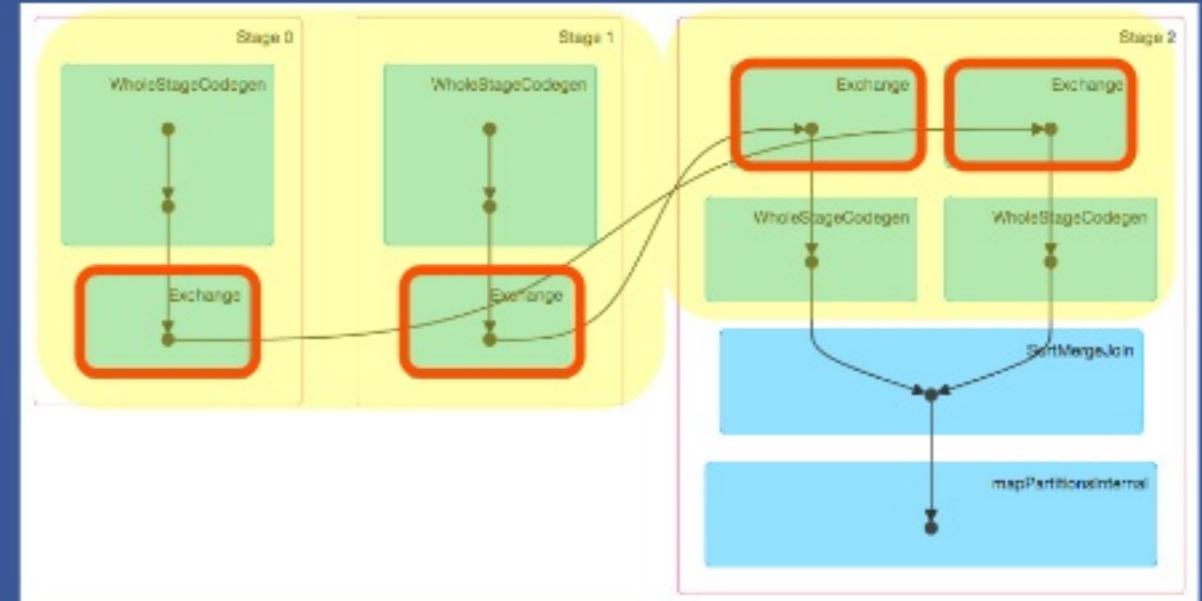
.....
.....

student s JOIN course_registration c
ON s.id = c.student_id



student s JOIN attendance a
ON s.id = a.student_id

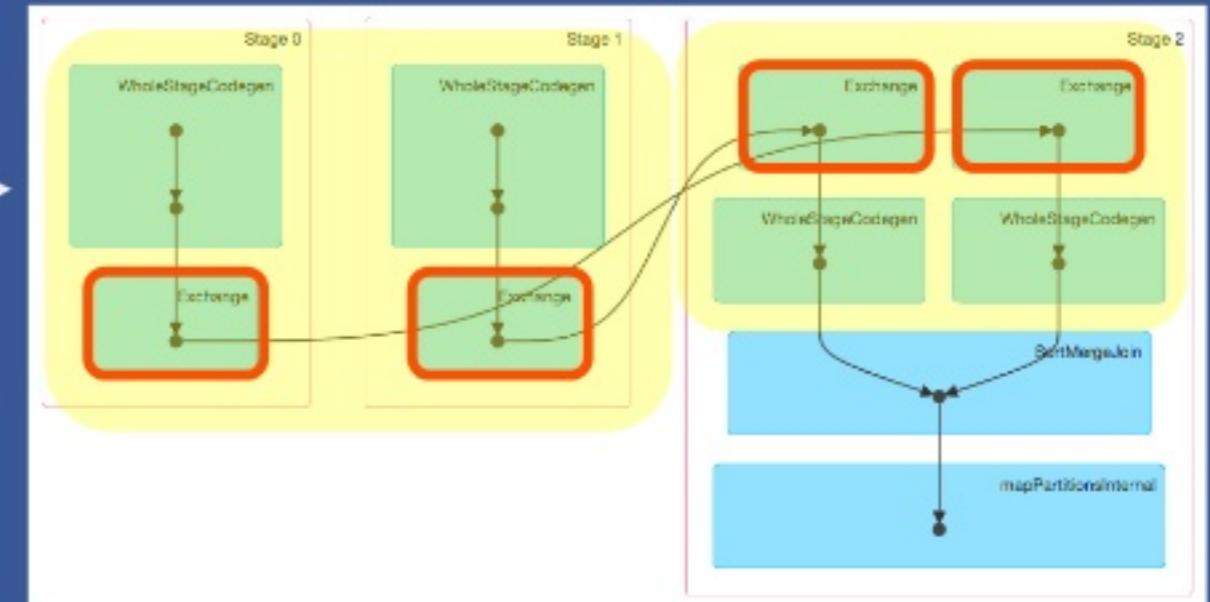
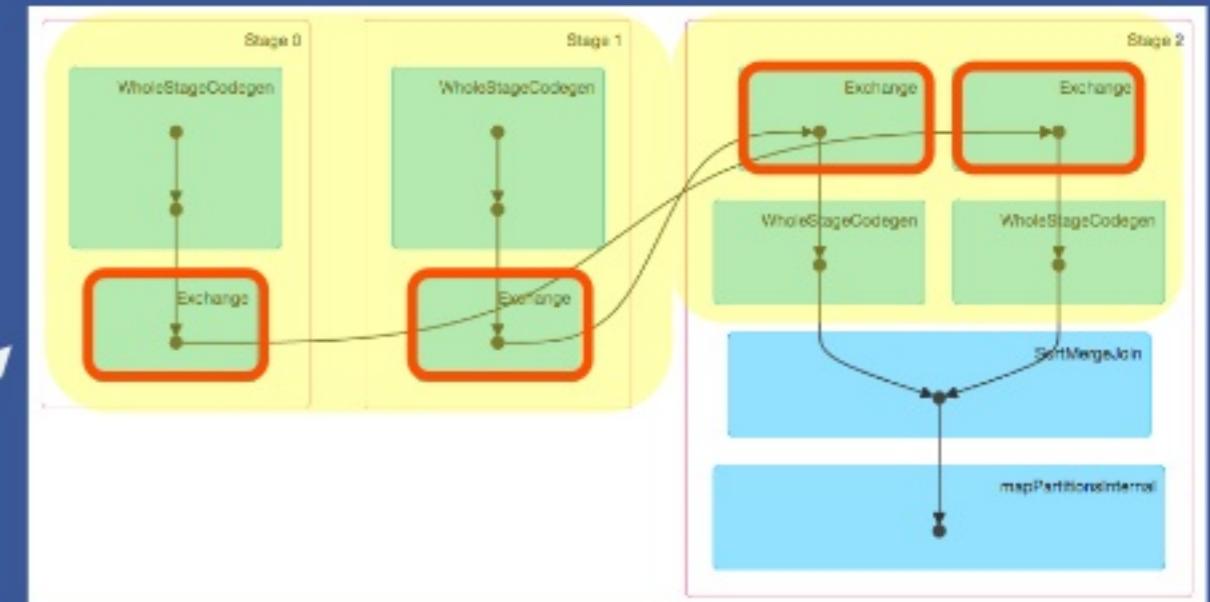
student s JOIN results r
ON s.id = r.student_id



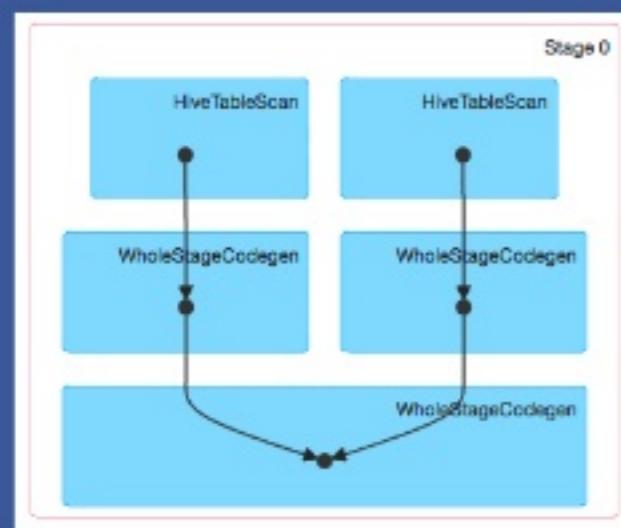
student s JOIN attendance a
ON s.id = a.student_id

student s JOIN results r
ON s.id = r.student_id

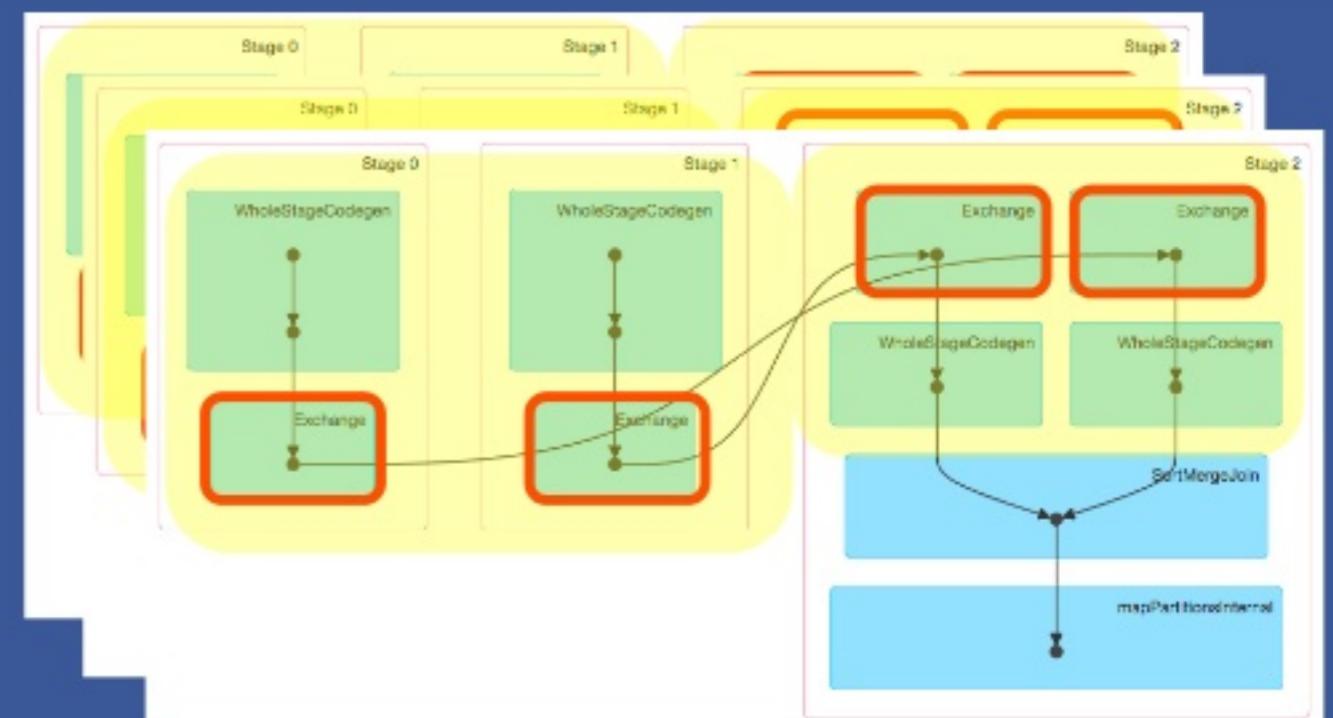
Pre-compute at
table creation



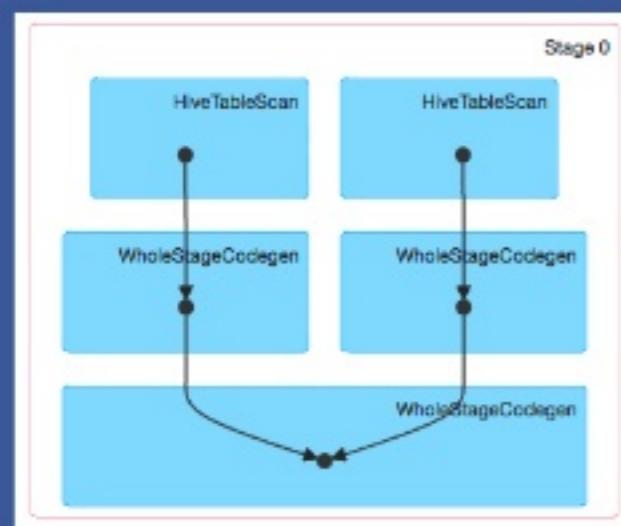
Bucketing =
pre-(shuffle + sort) inputs
on join keys



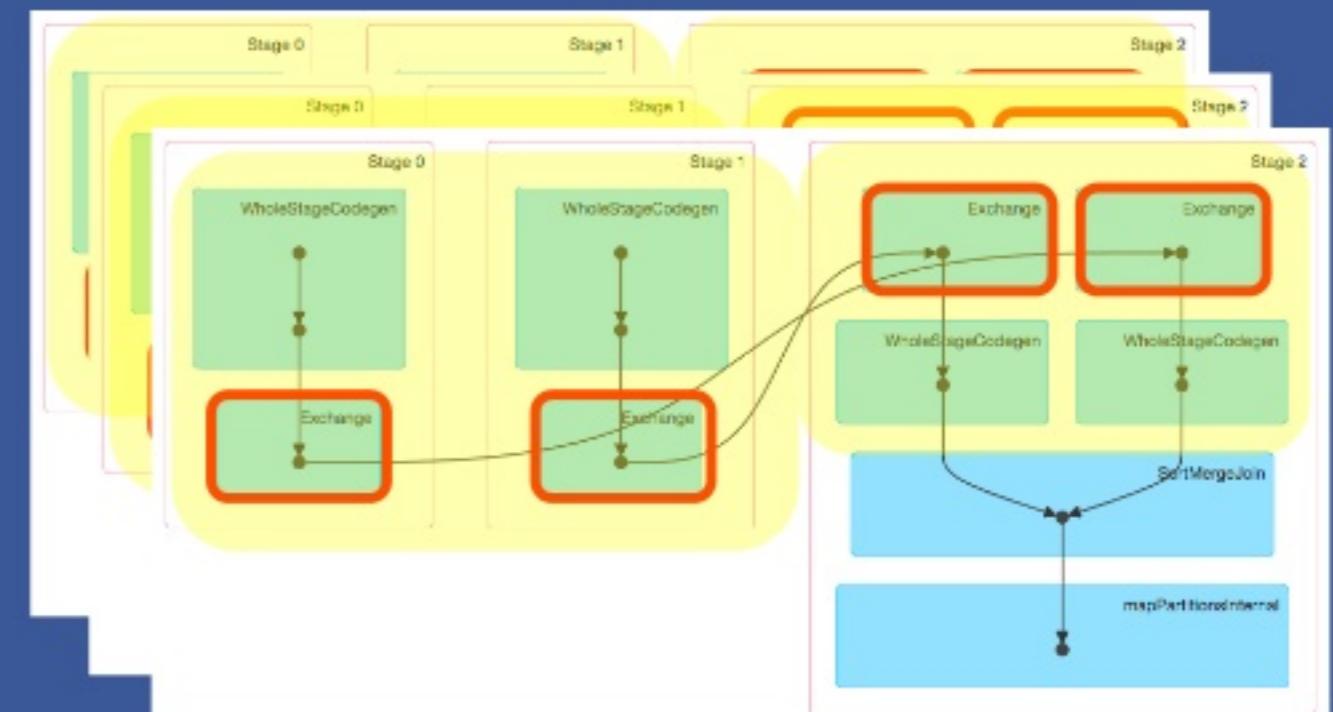
Without
bucketing



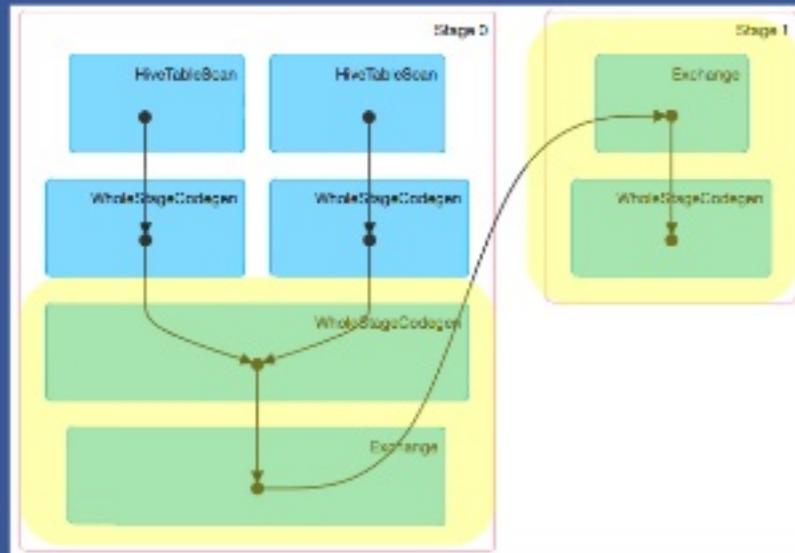
Job(s) populating input table



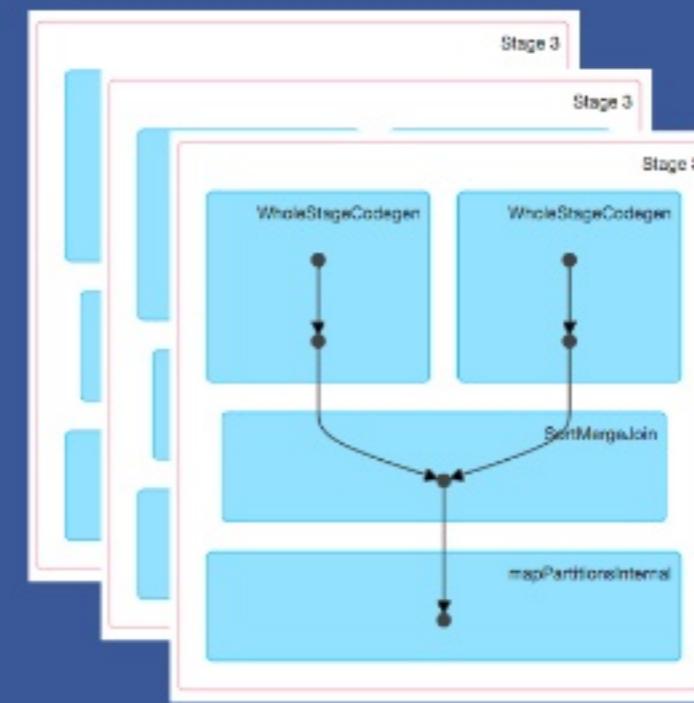
Without
bucketing



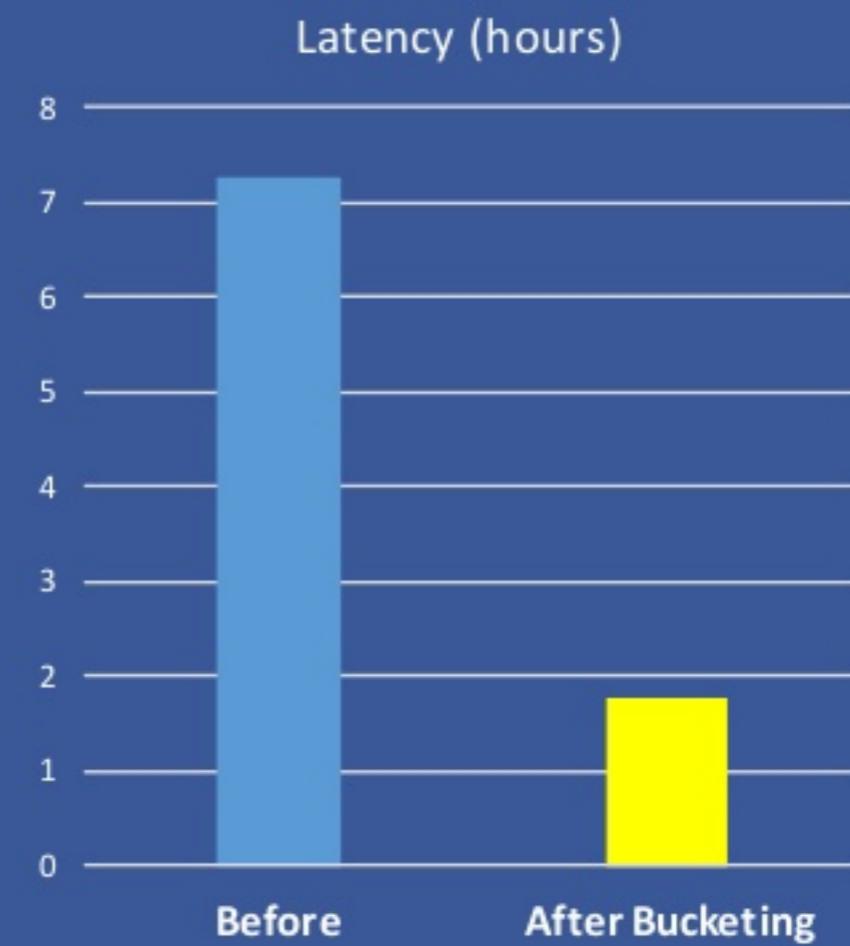
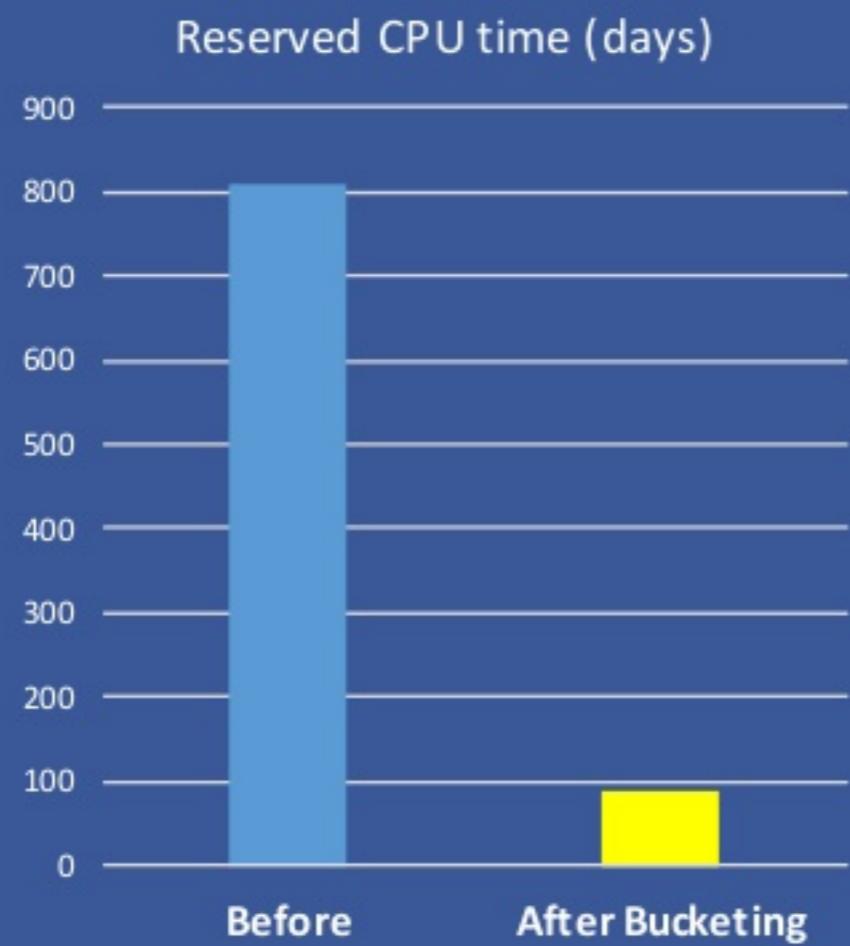
Job(s) populating input table



With
bucketing



Performance comparison



When to use bucketing ?

- Tables used frequently in JOINs with same key
 - Student -> student_id
 - Employee -> employee_id
 - Users -> user_id

When to use bucketing ?

- Tables used frequently in JOINs with same key
 - Student -> student_id
 - Employee -> employee_id
 - Users -> user_id
- => Dimension tables

When to use bucketing ?

- Tables used frequently in JOINs with same key
 - Student -> student_id
 - Employee -> employee_id
 - Users -> user_id
- Loading daily cumulative tables
 - Both base and delta tables could be bucketed on a common column

When to use bucketing ?

- Tables used frequently in JOINs with same key
 - Student -> student_id
 - Employee -> employee_id
 - Users -> user_id
- Loading daily cumulative tables
 - Both base and delta tables could be bucketed on a common column
- Indexing capability

Spark's bucketing support

 Spark / SPARK-12538
bucketed table support

[Edit](#) [Comment](#) [Agile Board](#) [More](#) [Close Issue](#) [Reopen Issue](#)

[Print](#) [Export](#)

Details

Type:	 New Feature	Status:	RESOLVED
Priority:	 Major	Resolution:	Fixed
Affects Version/s:	None	Fix Version/s:	2.0.0
Component/s:	SQL		
Labels:	None		
Target Version/s:	2.0.0		

Description

cc Nong Li , please attach the design doc.

People

Assignee:	 Wencheng Fan
Reporter:	 Wencheng Fan
Votes:	 0 Vote for this issue
Watchers:	 5 Start watching this issue

Dates

Created:	28/Dec/15 05:54
Updated:	15/Jan/16 17:22
Resolved:	15/Jan/16 17:21

Issue Links



Creation of bucketed tables

via Dataframe API

```
df.write  
  .bucketBy(numBuckets, "col1", ...)  
  .sortBy("col1", ...)  
  .saveAsTable("bucketed_table")
```

Creation of bucketed tables

via SQL statement

```
CREATE TABLE bucketed_table(  
    column1 INT,  
    ...  
) USING parquet  
CLUSTERED BY(column1, ...)  
SORTED BY (column1, ...)  
INTO `n` BUCKETS
```

Check bucketing spec

```
scala> sparkContext.sql("DESC FORMATTED student").collect.foreach(println)
[# col_name,data_type,comment]
[student_id,int,null]
[name,int,null]
[# Detailed Table Information,,]
[Database,default,]
[Table,table1,]
[Owner,tejas,]
[Created,Fri May 12 08:06:33 PDT 2017,]
[Type,MANAGED,]
[Num Buckets,64,]
[Bucket Columns,[`student_id`],]
[Sort Columns,[`student_id`],]
[Properties,[serialization.format=1],]
[Serde Library,org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe,]
[InputFormat,org.apache.hadoop.mapred.SequenceFileInputFormat,]
```

Bucketing config

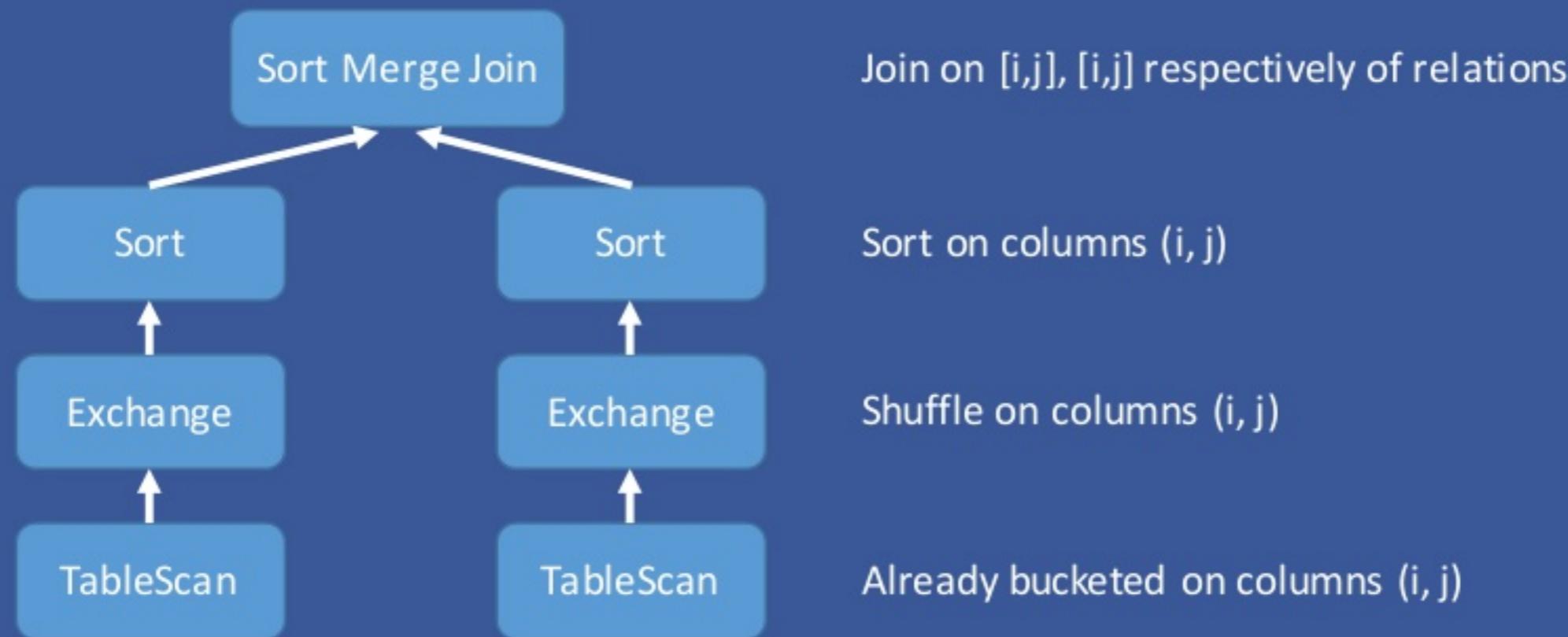
```
SET spark.sql.sources.bucketing.enabled=true
```

[SPARK-15453] Extract bucketing info in
FileSourceScanExec

```
SELECT * FROM tableA JOIN tableB ON tableA.i= tableB.i AND tableA.j= tableB.j
```

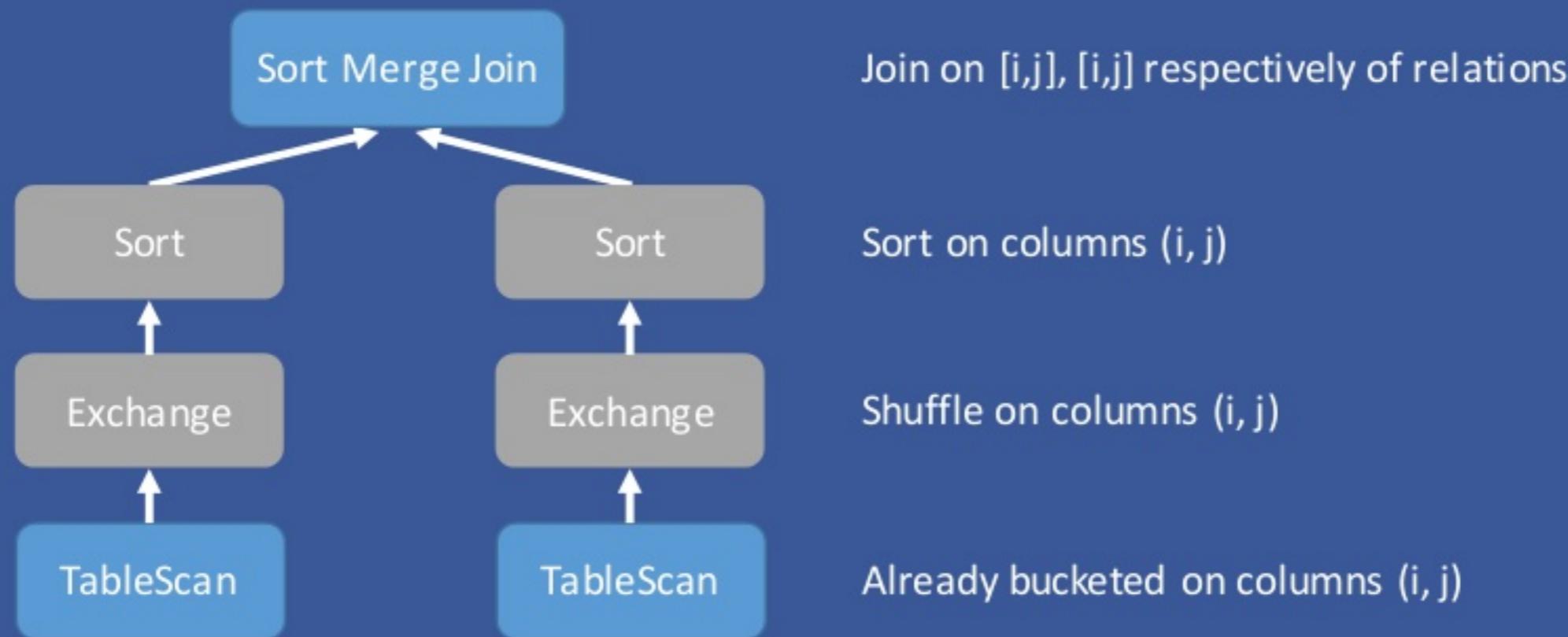
[SPARK-15453] Extract bucketing info in FileSourceScanExec

SELECT * FROM tableA JOIN tableB ON tableA.i= tableB.i AND tableA.j= tableB.j



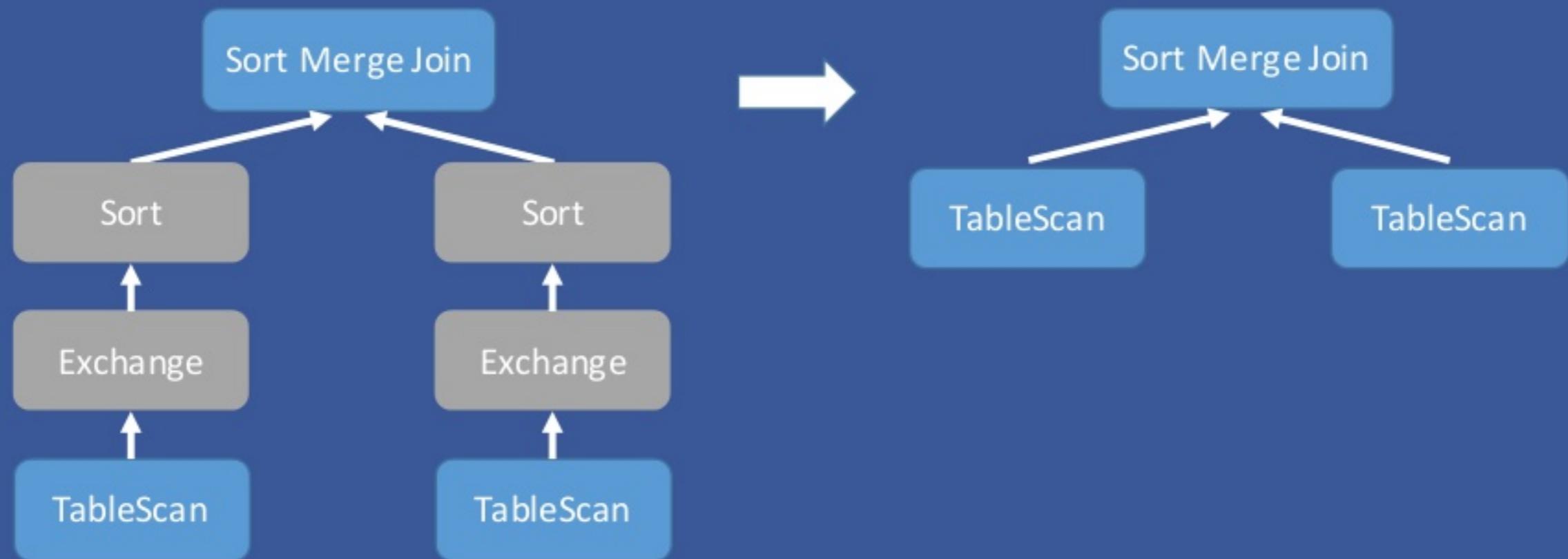
[SPARK-15453] Extract bucketing info in FileSourceScanExec

SELECT * FROM tableA JOIN tableB ON tableA.i= tableB.i AND tableA.j= tableB.j



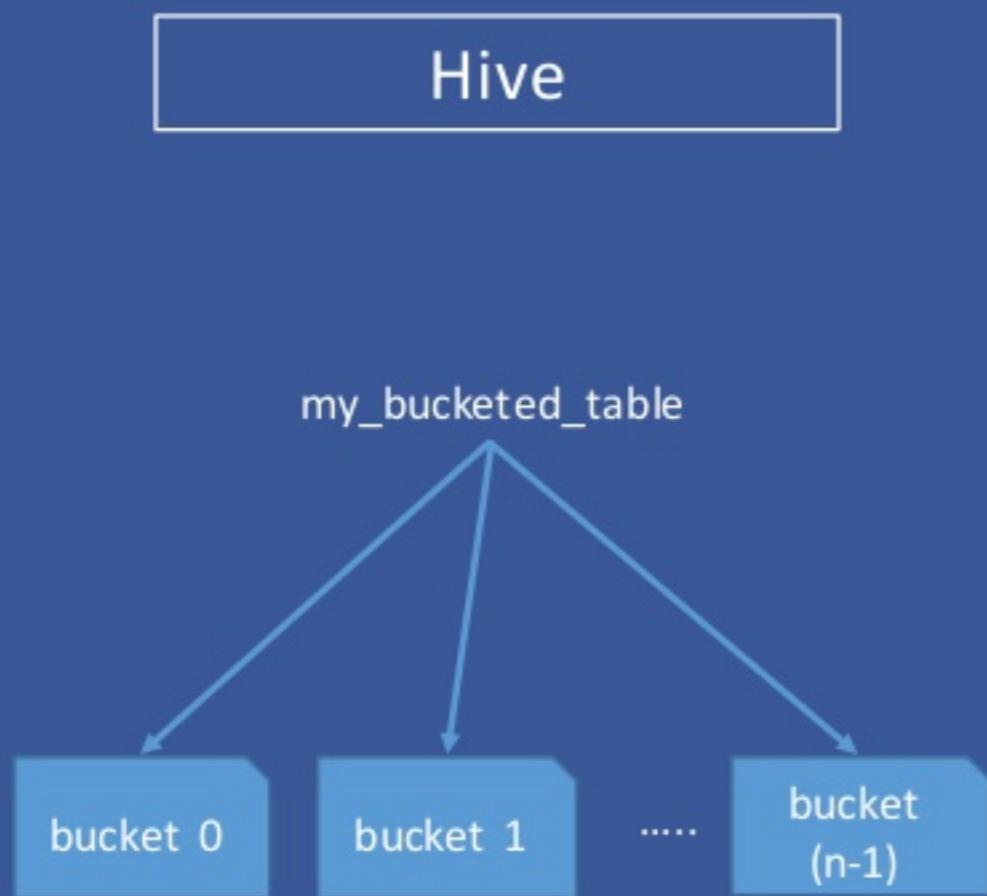
[SPARK-15453] Extract bucketing info in FileSourceScanExec

SELECT * FROM tableA JOIN tableB ON tableA.i= tableB.i AND tableA.j= tableB.j



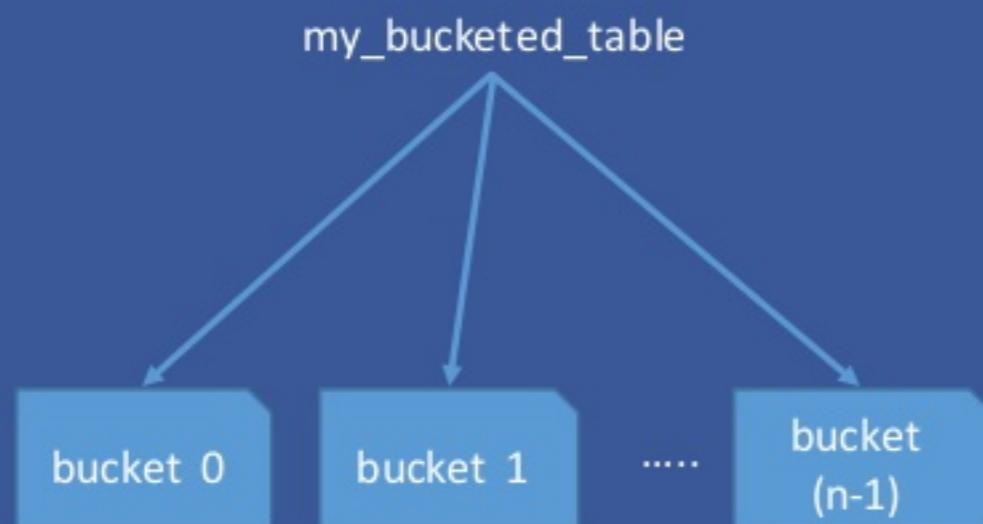
Bucketing semantics of Spark vs Hive

Bucketing semantics of Spark vs Hive

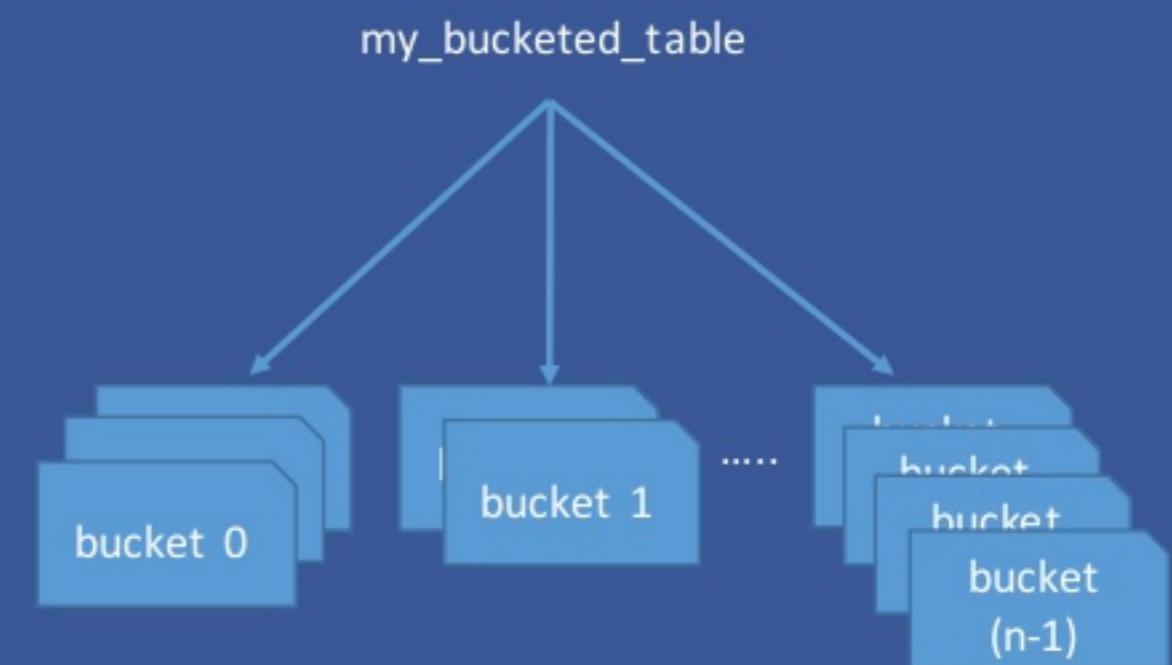


Bucketing semantics of Spark vs Hive

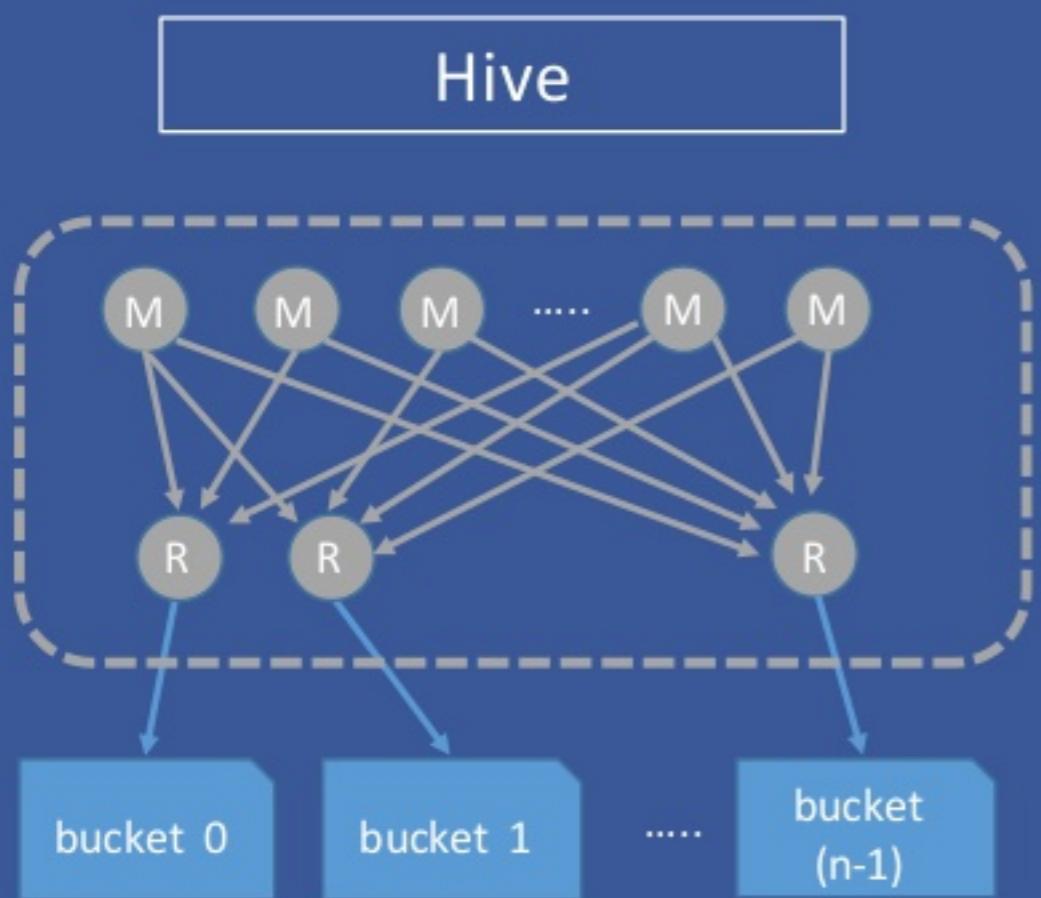
Hive



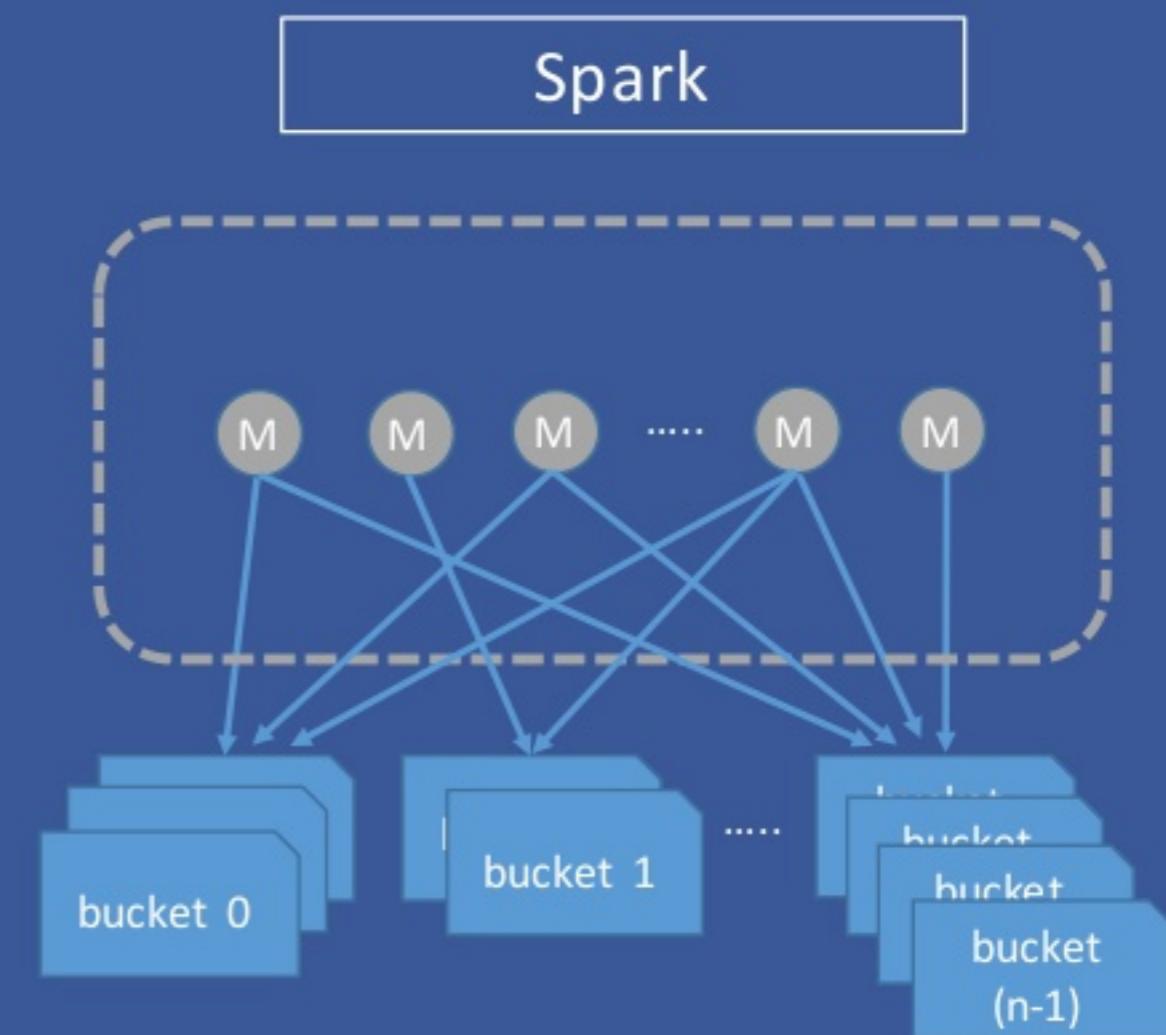
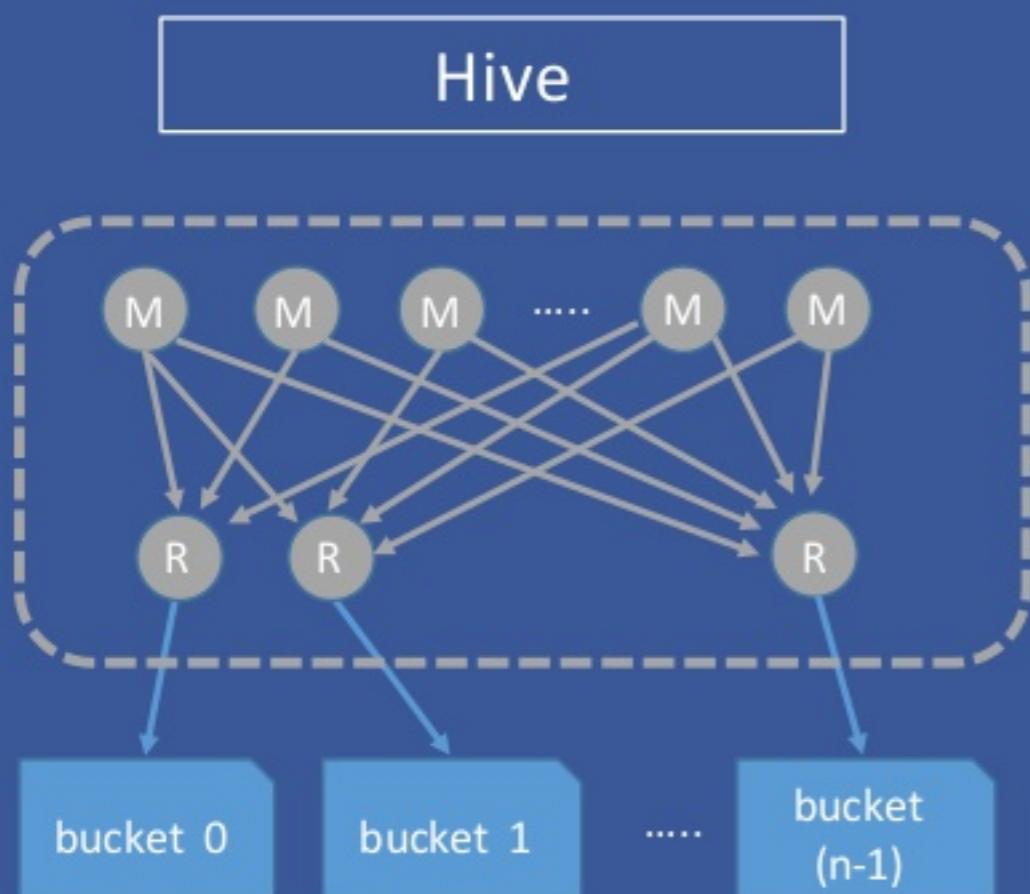
Spark



Bucketing semantics of Spark vs Hive



Bucketing semantics of Spark vs Hive



Bucketing semantics of Spark vs Hive

	Hive	Spark
Model	Optimizes reads, writes are costly	Writes are cheaper, reads are costlier

Bucketing semantics of Spark vs Hive

	Hive	Spark
Model	Optimizes reads, writes are costly	Writes are cheaper, reads are costlier
Unit of bucketing	A single file represents a single bucket	A collection of files together comprise a single bucket

Bucketing semantics of Spark vs Hive

	Hive	Spark
Model	Optimizes reads, writes are costly	Writes are cheaper, reads are costlier
Unit of bucketing	A single file represents a single bucket	A collection of files together comprise a single bucket
Sort ordering	Each bucket is sorted globally. This makes it easy to optimize queries reading this data	Each file is individually sorted but the “bucket” as a whole is not globally sorted

Bucketing semantics of Spark vs Hive

	Hive	Spark
Model	Optimizes reads, writes are costly	Writes are cheaper, reads are costlier
Unit of bucketing	A single file represents a single bucket	A collection of files together comprise a single bucket
Sort ordering	Each bucket is sorted globally. This makes it easy to optimize queries reading this data	Each file is individually sorted but the “bucket” as a whole is not globally sorted
Hashing function	Hive’s inbuilt hash	Murmur3Hash

[SPARK-19256] Hive bucketing support

- Introduce Hive's hashing function [SPARK-17495]
- Enable creating hive bucketed tables [SPARK-17729]
- Support Hive's `Bucketed file format`
- Propagate Hive bucketing information to planner [SPARK-17654]
 - Expressing outputPartitioning and requiredChildDistribution
 - Creating empty bucket files when no data for a bucket
- Allow Sort Merge join over tables with number of buckets multiple of each other
- Support N-way Sort Merge Join

[SPARK-19256] Hive bucketing support

- Introduce Hive's hashing function [SPARK-17495]
- Enable creating hive bucketed tables [SPARK-17729]

Merged in upstream

- Support Hive's 'Bucketed file format'
- Propagate Hive bucketing information to planner [SPARK-17654]
 - Expressing outputPartitioning and requiredChildDistribution
 - Creating empty bucket files when no data for a bucket
- Allow Sort Merge join over tables with number of buckets multiple of each other
- Support N-way Sort Merge Join

[SPARK-19256] Hive bucketing support

FB-prod (6 months)

- Introduce Hive's hashing function [SPARK-17495]
- Enable creating hive bucketed tables [SPARK-17729]
- Support Hive's 'Bucketed file format'
- Propagate Hive bucketing information to planner [SPARK-17654]
 - Expressing outputPartitioning and requiredChildDistribution
 - Creating empty bucket files when no data for a bucket
- Allow Sort Merge join over tables with number of buckets multiple of each other
- Support N-way Sort Merge Join

SQL Planner improvements

[SPARK-19122] Unnecessary shuffle+sort added if join predicates ordering differ from bucketing and sorting order

```
SELECT * FROM a JOIN b ON a.x=b.x AND a.y=b.y
```

[SPARK-19122] Unnecessary shuffle+sort added if join predicates ordering differ from bucketing and sorting order

```
SELECT * FROM a JOIN b ON a.x=b.x AND a.y=b.y
```



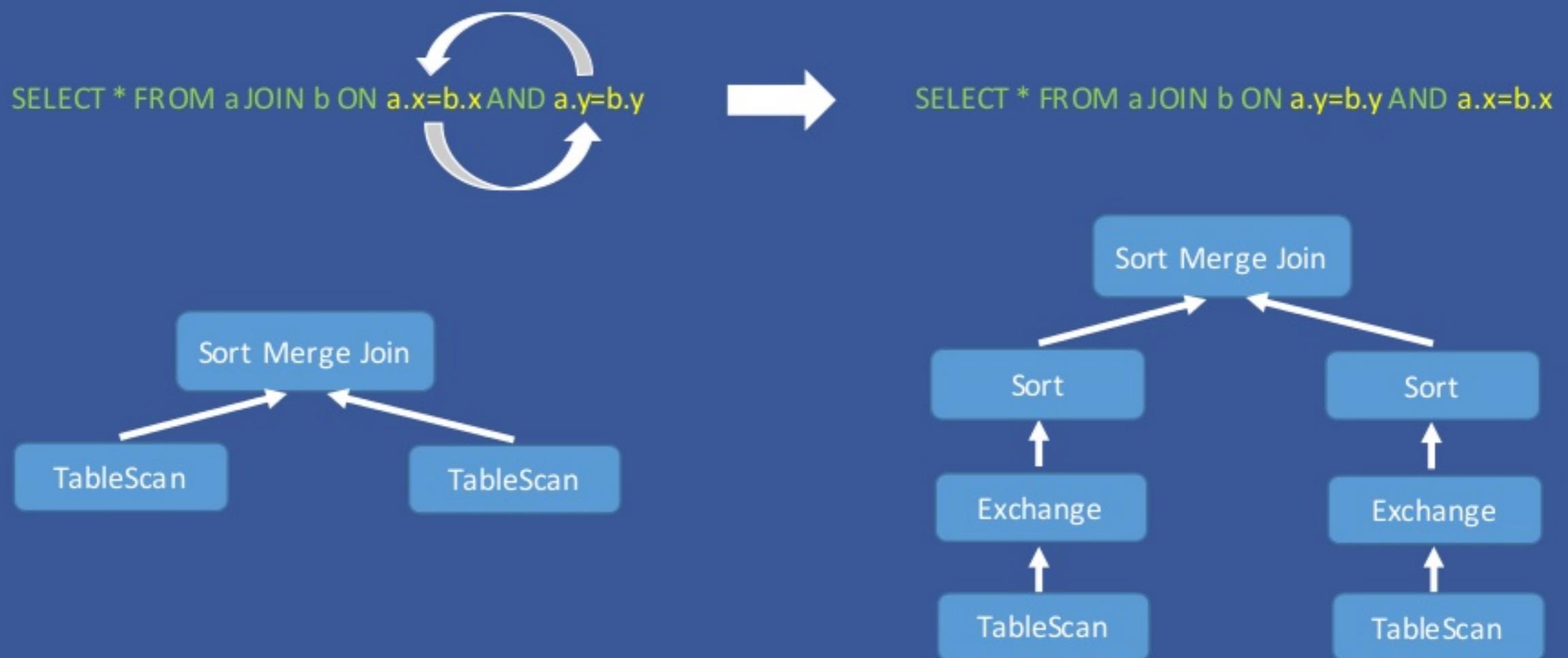
[SPARK-19122] Unnecessary shuffle+sort added if join predicates ordering differ from bucketing and sorting order

`SELECT * FROM a JOIN b ON a.x=b.x AND a.y=b.y`  

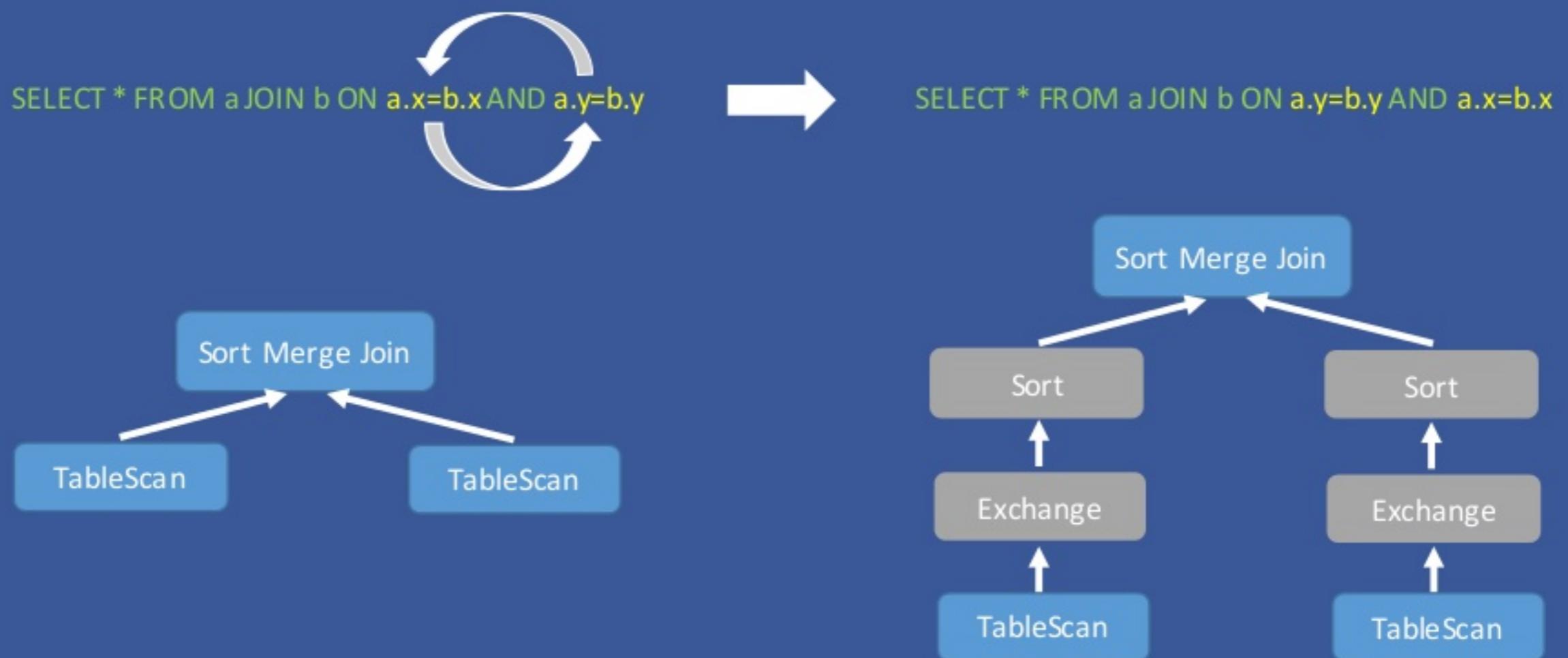
`SELECT * FROM a JOIN b ON a.y=b.y AND a.x=b.x`



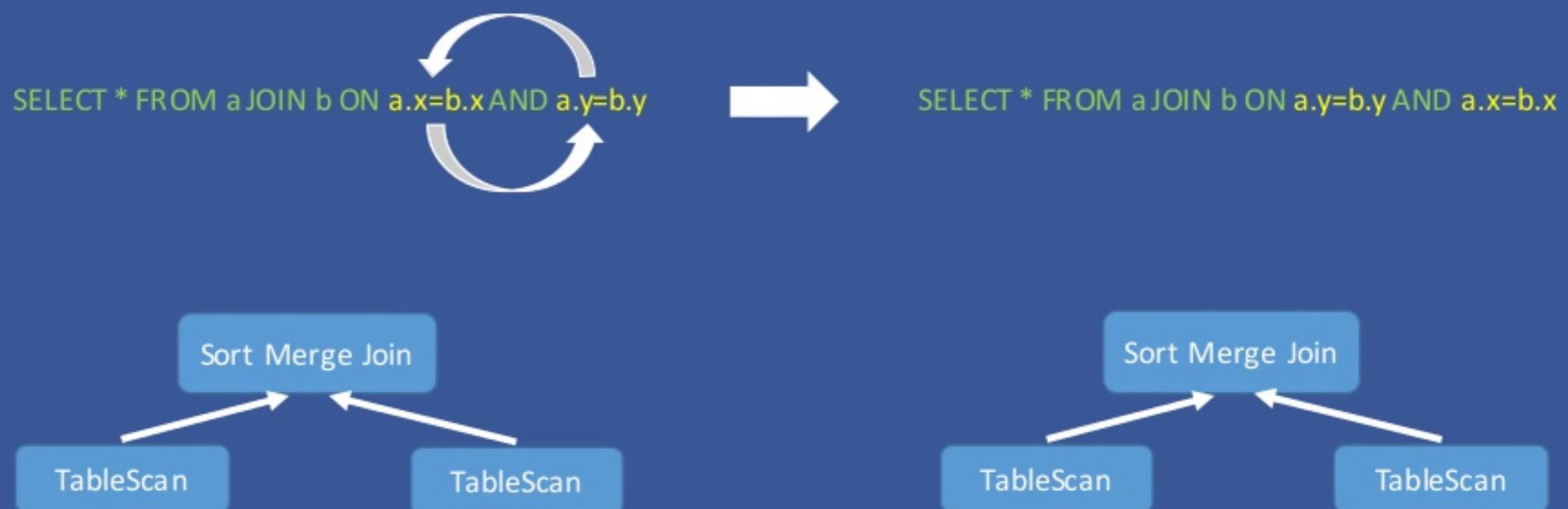
[SPARK-19122] Unnecessary shuffle+sort added if join predicates ordering differ from bucketing and sorting order



[SPARK-19122] Unnecessary shuffle+sort added if join predicates ordering differ from bucketing and sorting order



[SPARK-19122] Unnecessary shuffle+sort added if join predicates ordering differ from bucketing and sorting order

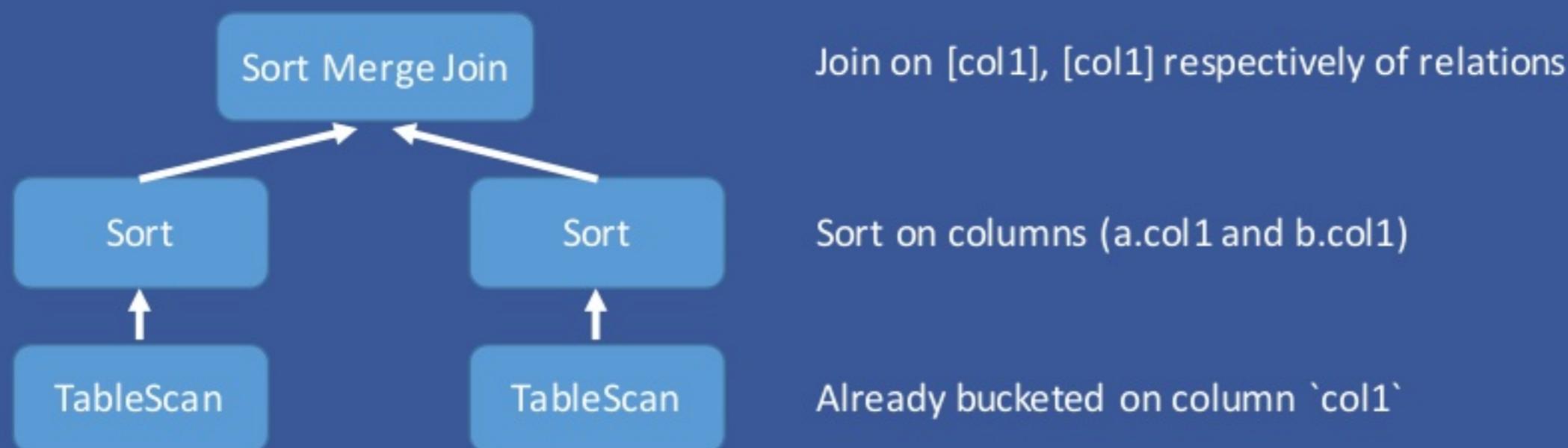


[SPARK-17271] Planner adds un-necessary Sort even if child ordering is semantically same as required ordering

```
SELECT * FROM table1 a JOIN table2 b ON a.col1=b.col1
```

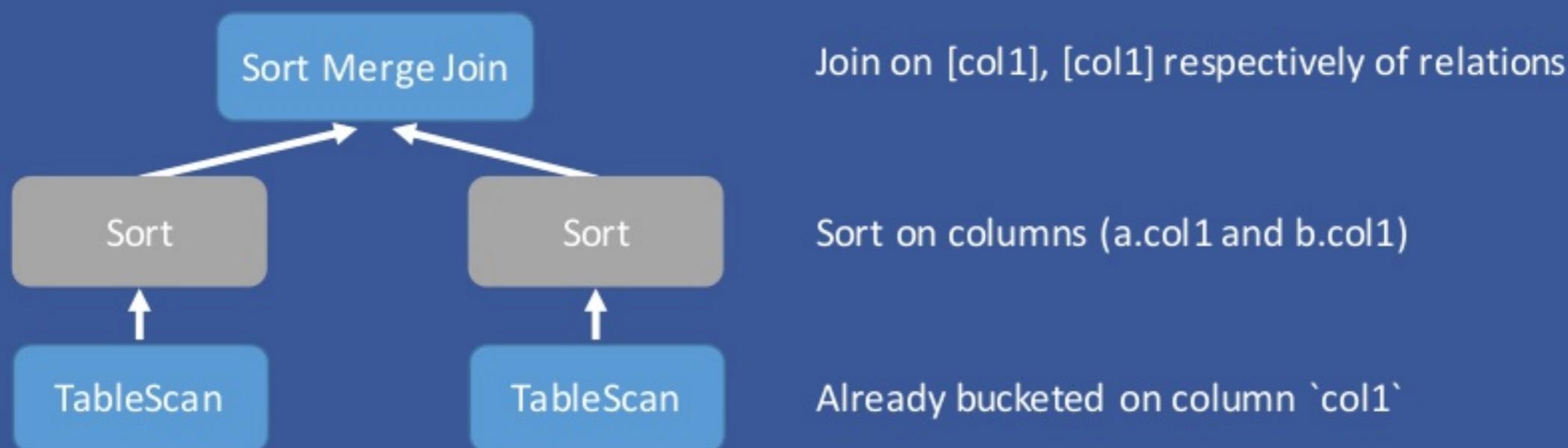
[SPARK-17271] Planner adds un-necessary Sort even if child ordering is semantically same as required ordering

SELECT * FROM table1 a JOIN table2 b ON a.col1=b.col1



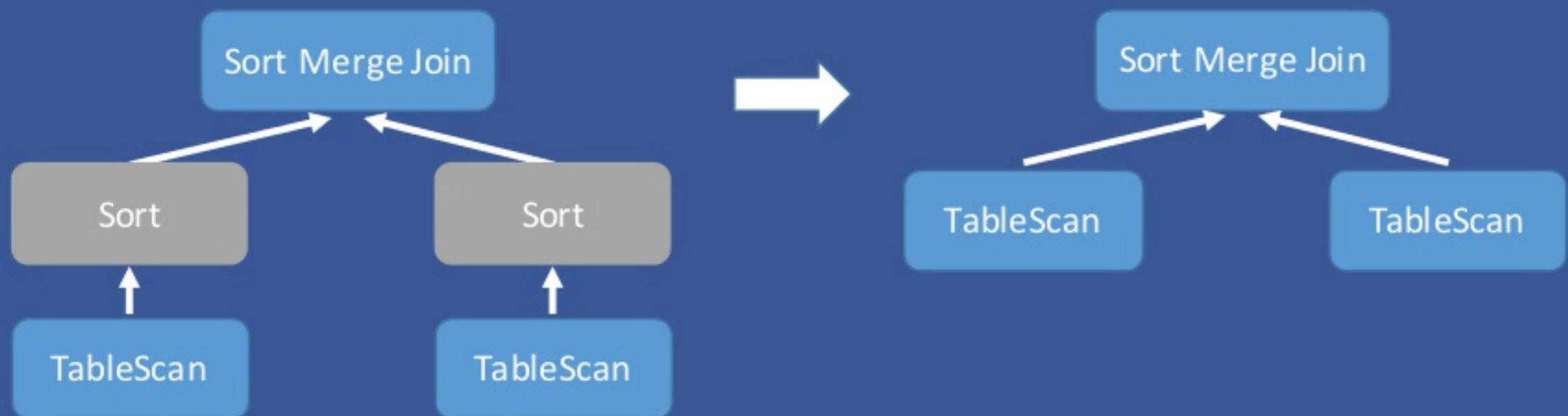
[SPARK-17271] Planner adds un-necessary Sort even if child ordering is semantically same as required ordering

SELECT * FROM table1 a JOIN table2 b ON a.col1=b.col1



[SPARK-17271] Planner adds un-necessary Sort even if child ordering is semantically same as required ordering

SELECT * FROM table1 a JOIN table2 b ON a.col1=b.col1

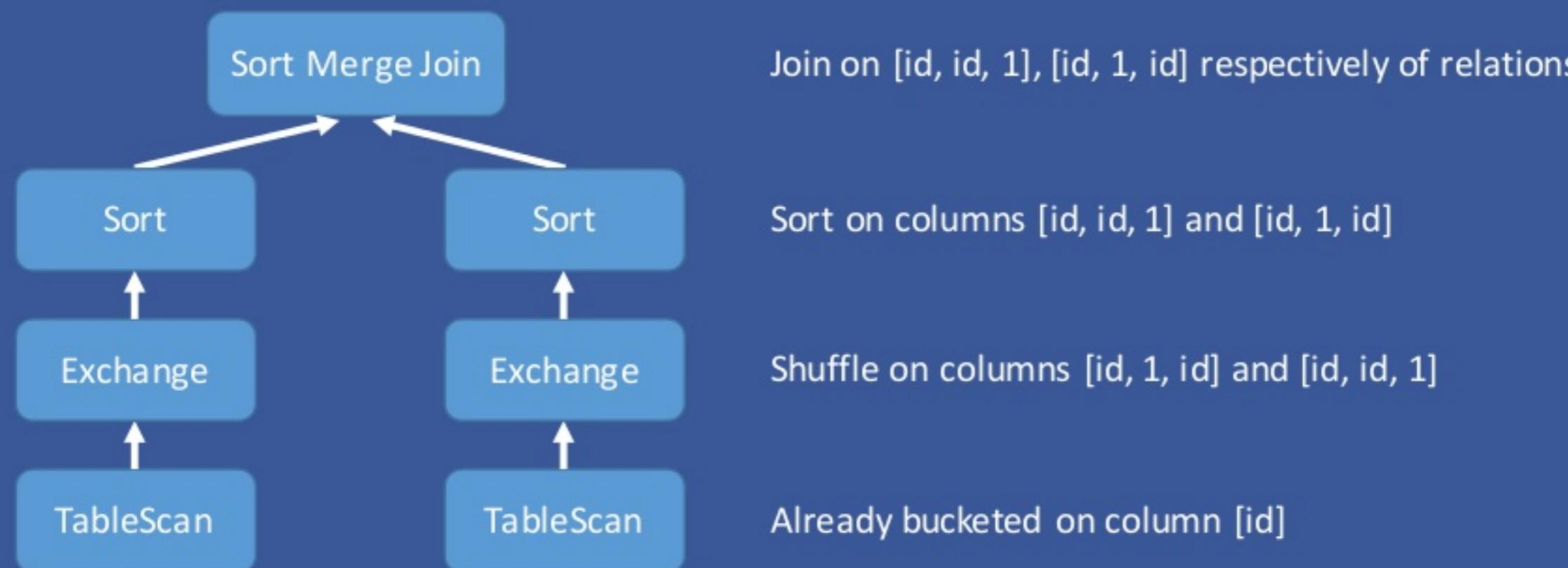


[SPARK-17698] Join predicates should not contain filter clauses

```
SELECT a.id, b.id FROM table1 a FULL OUTER JOIN table2 b ON a.id = b.id AND a.id='1' AND b.id='1'
```

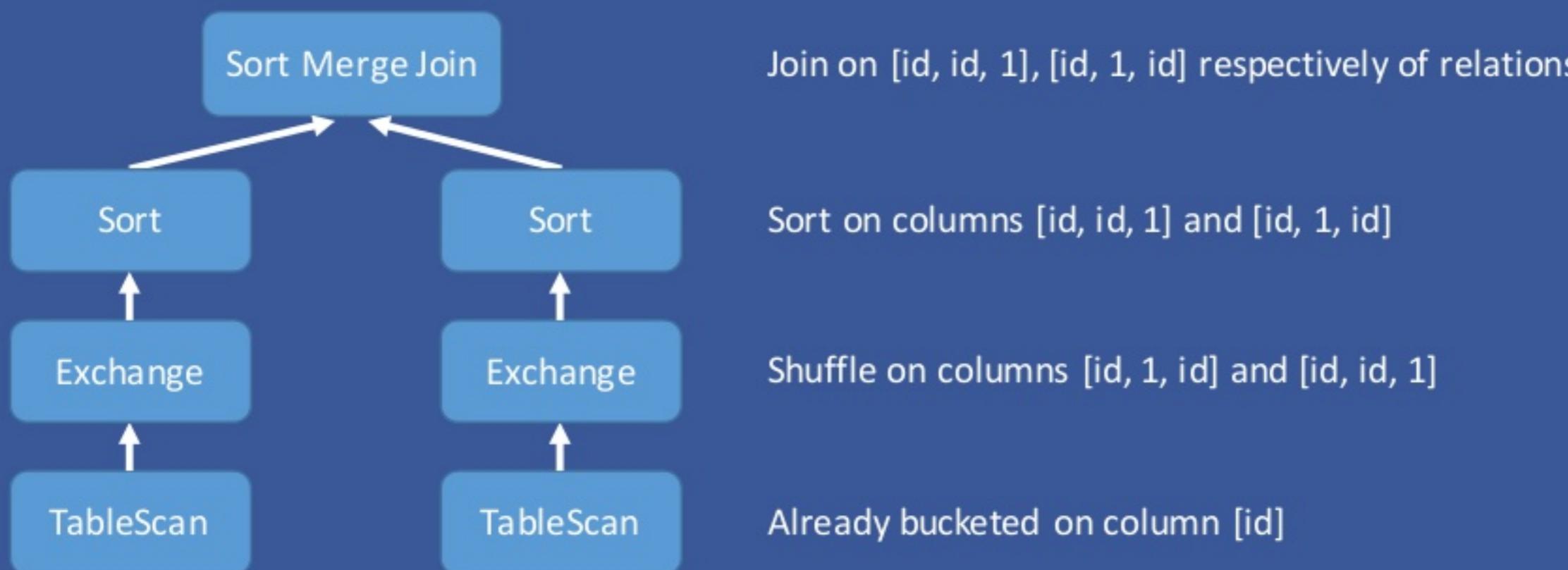
[SPARK-17698] Join predicates should not contain filter clauses

```
SELECT a.id, b.id FROM table1 a FULL OUTER JOIN table2 b ON a.id = b.id AND a.id='1' AND b.id='1'
```



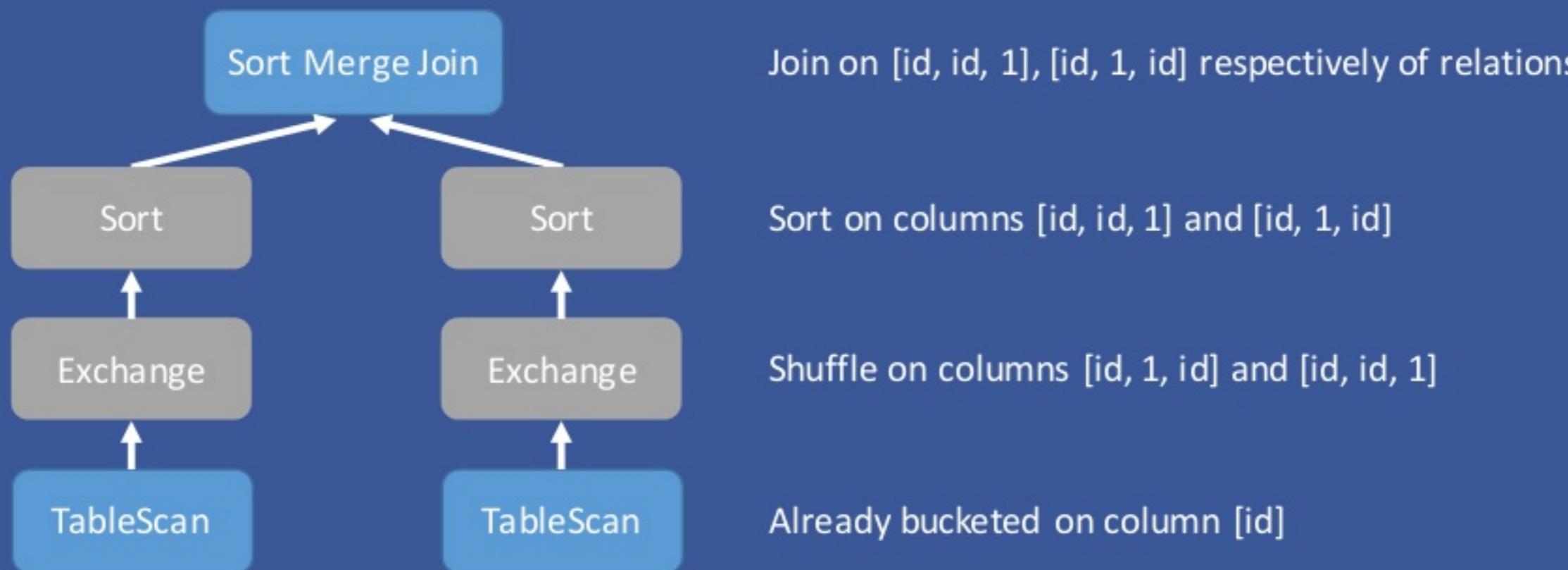
[SPARK-17698] Join predicates should not contain filter clauses

```
SELECT a.id, b.id FROM table1 a FULL OUTER JOIN table2 b ON a.id = b.id AND a.id='1' AND b.id='1'
```



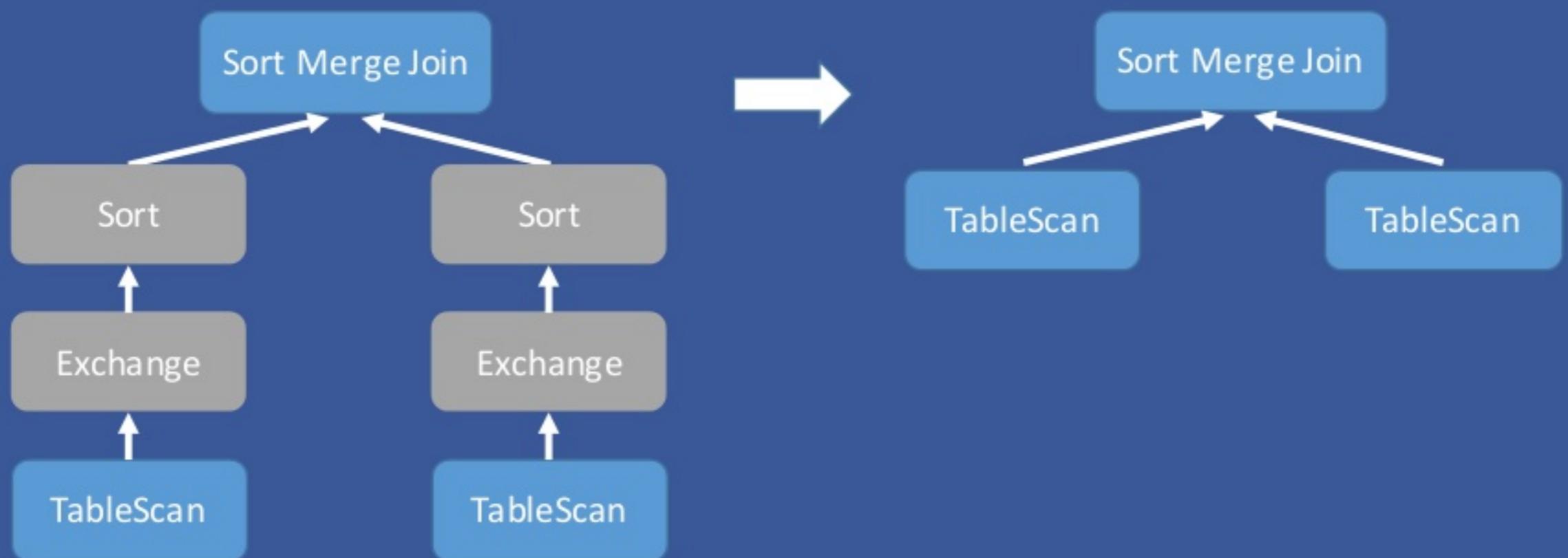
[SPARK-17698] Join predicates should not contain filter clauses

```
SELECT a.id, b.id FROM table1 a FULL OUTER JOIN table2 b ON a.id = b.id AND a.id='1' AND b.id='1'
```



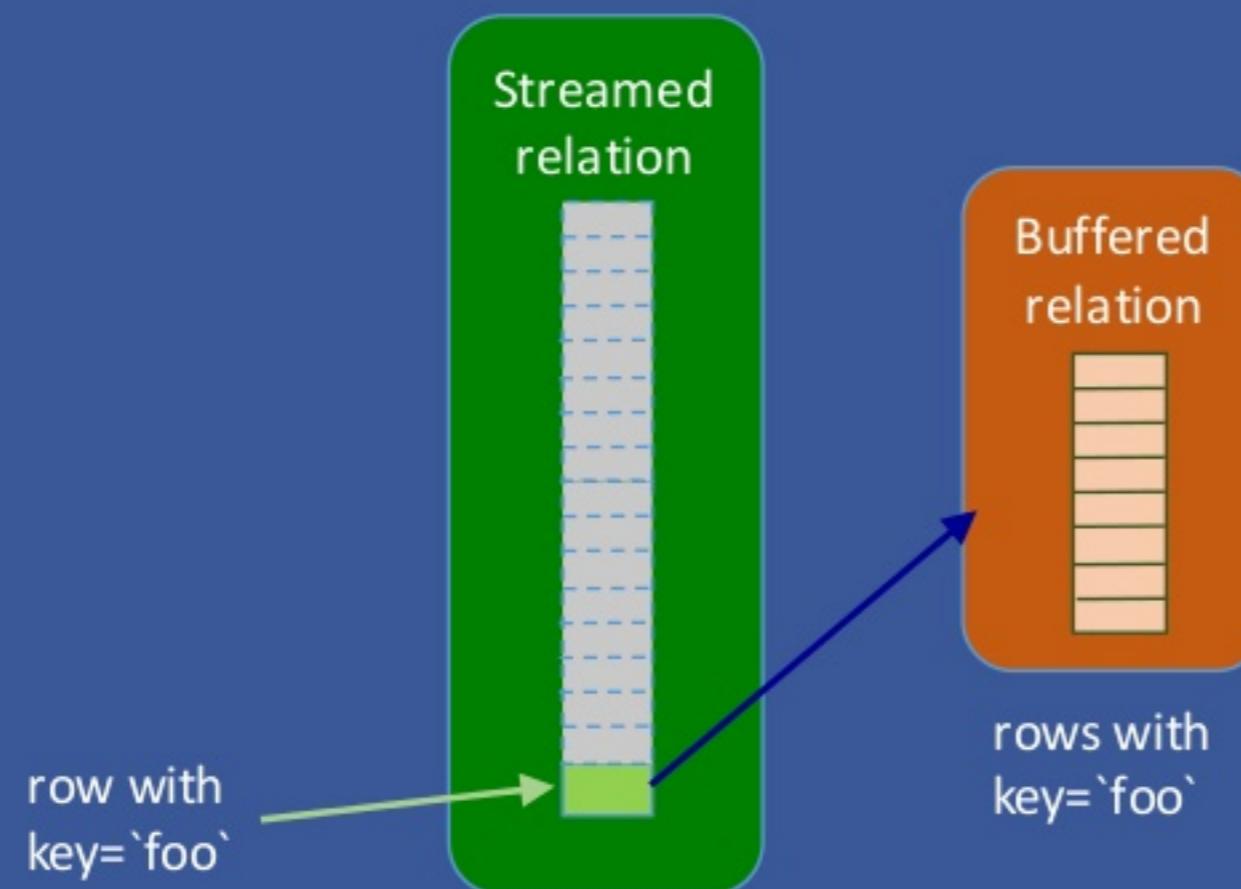
[SPARK-17698] Join predicates should not contain filter clauses

```
SELECT a.id, b.id FROM table1 a FULL OUTER JOIN table2 b ON a.id = b.id AND a.id='1' AND b.id='1'
```



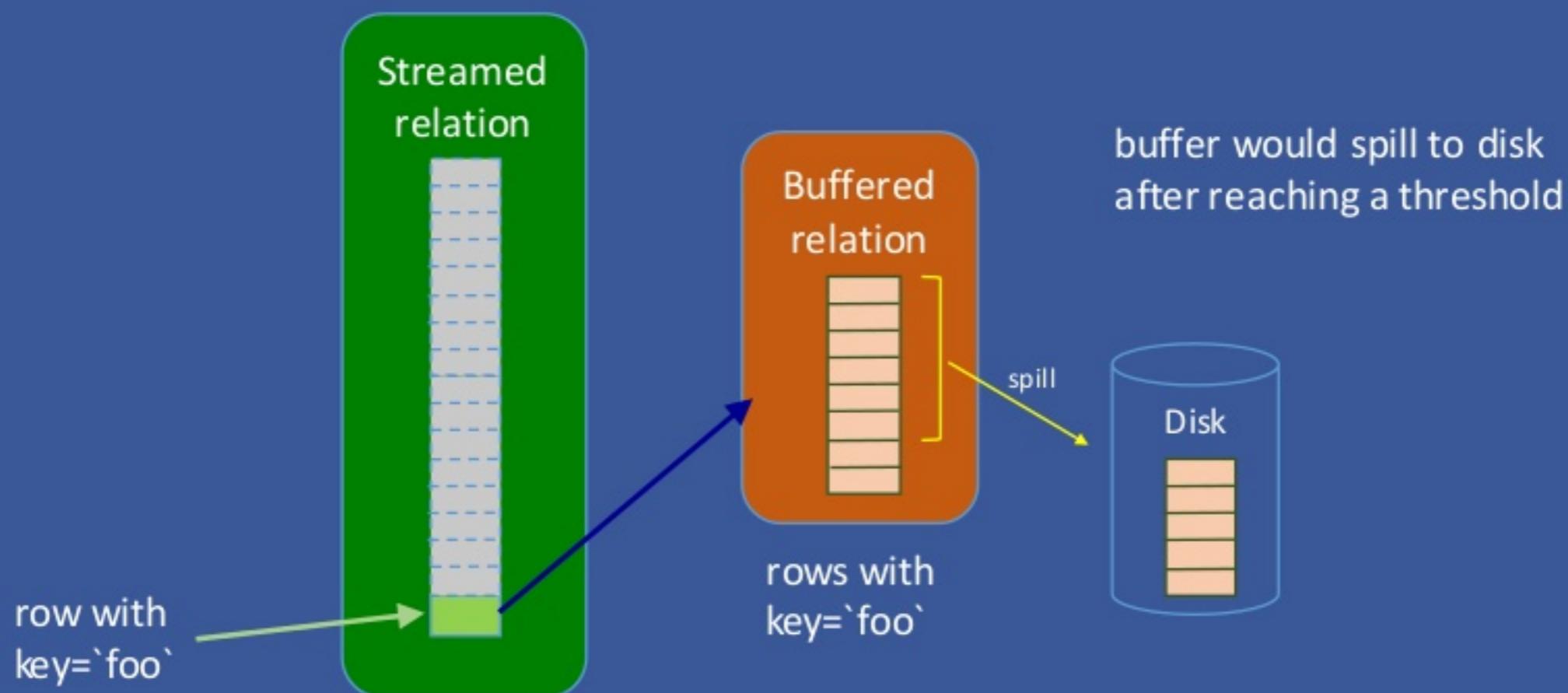
[SPARK-13450] Introduce
ExternalAppendOnlyUnsafeRowArray

[SPARK-13450] Introduce ExternalAppendOnlyUnsafeRowArray



If there are a lot of rows to be buffered (eg. Skew), the query would OOM

[SPARK-13450] Introduce ExternalAppendOnlyUnsafeRowArray



Summary

- Shuffle and sort is costly due to disk and network IO
- Bucketing will pre-(shuffle and sort) the inputs
- Expect at least 2-5x gains after bucketing input tables for joins
- Candidates for bucketing:
 - Tables used frequently in JOINs with same key
 - Loading of data in cumulative tables
- Spark's bucketing is not compatible with Hive's bucketing

Questions ?