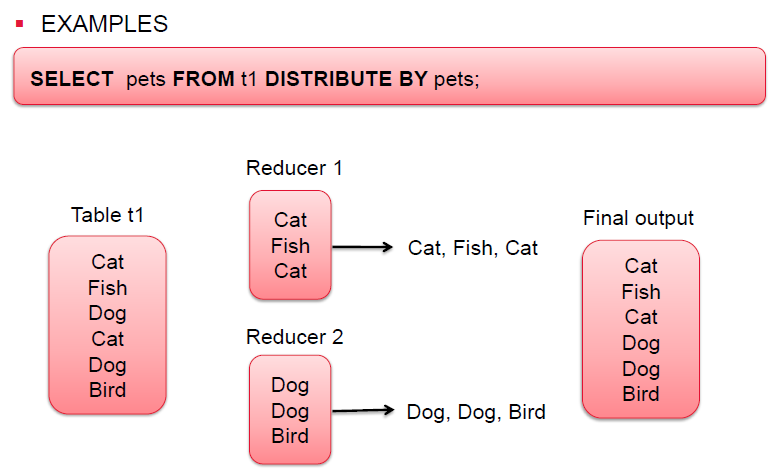
|  |
| --- |
| Create table emp( emp\_id int,empname string,deptid int )ROW FORMAT DELIMITED FIELDS TERMINATED BY ','; |
| 1,Raju,2  2,Nimish,1  3,Ruma,2  4,Subrata,1  5,Gargi,2  6,Siddharth,2  7,Anil,3  8,Ashis,3  9,Palas,2  10,Mr X,5 |
| load data local inpath '/root/emp.data' into table emp; |

|  |
| --- |
| Create table dept( deptid int,deptname string )ROW FORMAT DELIMITED FIELDS TERMINATED BY ','; |
| 1,Sales  2,Marketing  3,HR  4,Engineer  5,Finance |
| load data local inpath '/root/dept.data' into table dept; |

Only distributed by doesn’t give you sorted output it just group the column and pass the same key to same reducer

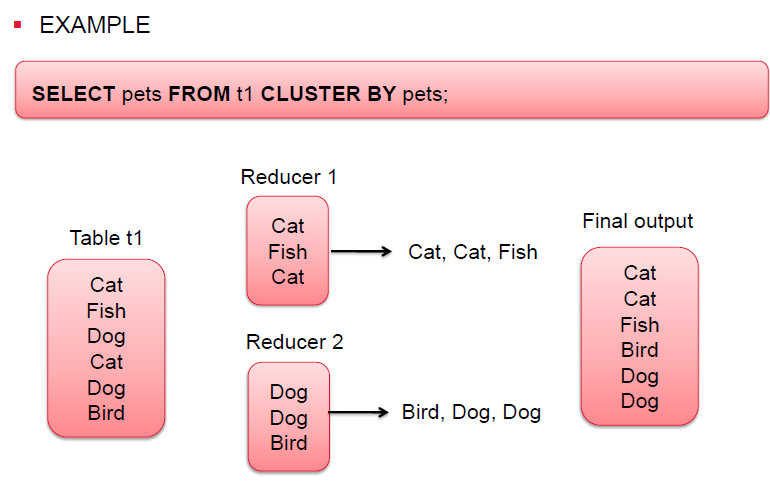


|  |
| --- |
| select \* from emp distribute by deptid; |
| **emp\_id empname deptid**  10 Mr X 5  9 Palas 2  8 Ashis 3  7 Anil 3  6 Siddharth 2  5 Gargi 2  4 Subrata 1  3 Ruma 2  2 Nimish 1  1 Raju 2 |

To sort it we need to use sort by operator but again it sort per reducer basis

|  |
| --- |
| select \* from emp distribute by deptid sort by deptid; |
| **emp\_id**  **empname** **deptid**  4 Subrata 1  2 Nimish 1  9 Palas 2  6 Siddharth 2  5 Gargi 2  3 Ruma 2  1 Raju 2  8 Ashis 3  7 Anil 3  10 Mr X 5 |

We also can use cluster by instead of distribute by and sort by

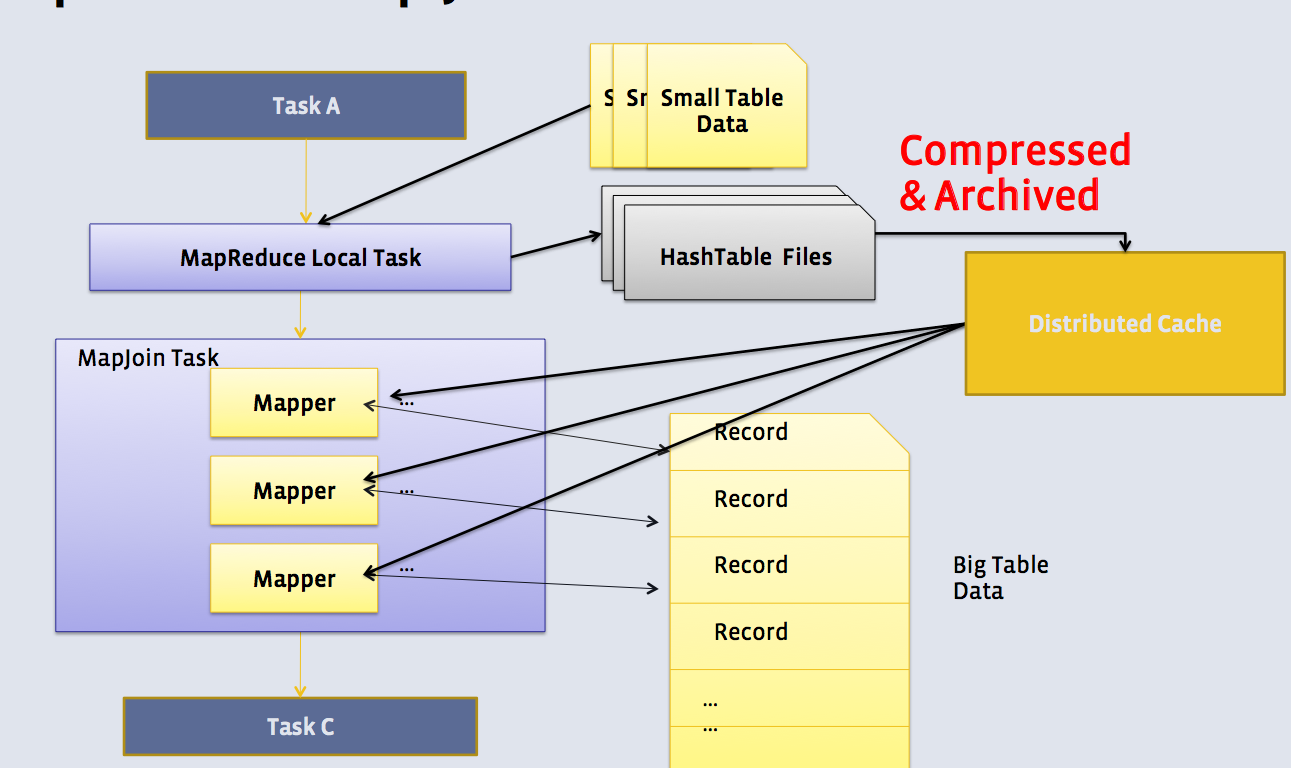


|  |
| --- |
| select \* from emp cluster by deptid; |
| **emp\_id**  **empname** **deptid**  4 Subrata 1  2 Nimish 1  9 Palas 2  6 Siddharth 2  5 Gargi 2  3 Ruma 2  1 Raju 2  8 Ashis 3  7 Anil 3  10 Mr X 5 |

<http://www.openkb.info/2014/11/understanding-hive-joins-in-explain.html>

## Map Join(Broardcast Join)

If one or more tables are small enough to fit in memory, the mapper scans the large table and do the joins. No shuffle and reduce stage.



If the table size is smaller than value of hive.auto.convert.join.noconditionaltask.size then join automatically get converted into map side join as you can see in below explain plan

Current value of hive.auto.convert.join.noconditionaltask.size=223696213

So no reducers are required for map side join and it is

|  |
| --- |
| explain select e.empname,d.deptname,d.deptid from emp e join dept d on e.deptid=d.deptid;  OK  STAGE DEPENDENCIES:  Stage-4 is a root stage  Stage-3 depends on stages: Stage-4  Stage-0 depends on stages: Stage-3  STAGE PLANS:  Stage: Stage-4  Map Reduce Local Work  Alias -> Map Local Tables:  d  Fetch Operator  limit: -1  Alias -> Map Local Operator Tree:  d  TableScan  alias: d  filterExpr: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 46 Basic stats: COMPLETE Column stats: NONE  Filter Operator  predicate: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 46 Basic stats: COMPLETE Column stats: NONE  HashTable Sink Operator  keys:  0 deptid (type: int)  1 deptid (type: int)  Stage: Stage-3  Map Reduce  Map Operator Tree:  TableScan  alias: e  filterExpr: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 104 Basic stats: COMPLETE Column stats: NONE  Filter Operator  predicate: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 104 Basic stats: COMPLETE Column stats: NONE  **Map Join Operator**  condition map:  Inner Join 0 to 1  keys:  0 deptid (type: int)  1 deptid (type: int)  outputColumnNames: \_col1, \_col6, \_col7  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  Select Operator  expressions: \_col1 (type: string), \_col7 (type: string), \_col6 (type: int)  outputColumnNames: \_col0, \_col1, \_col2  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  File Output Operator  compressed: false  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  table:  input format: org.apache.hadoop.mapred.TextInputFormat  output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat  serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe  Local Work:  Map Reduce Local Work  Stage: Stage-0  Fetch Operator  limit: -1  Processor Tree:  ListSink |

**Tips:**

**1. Auto convert shuffle/common join to map join.**  
3 parameters are related:

set hive.auto.convert.join=true;

set hive.auto.convert.join.noconditionaltask=true;

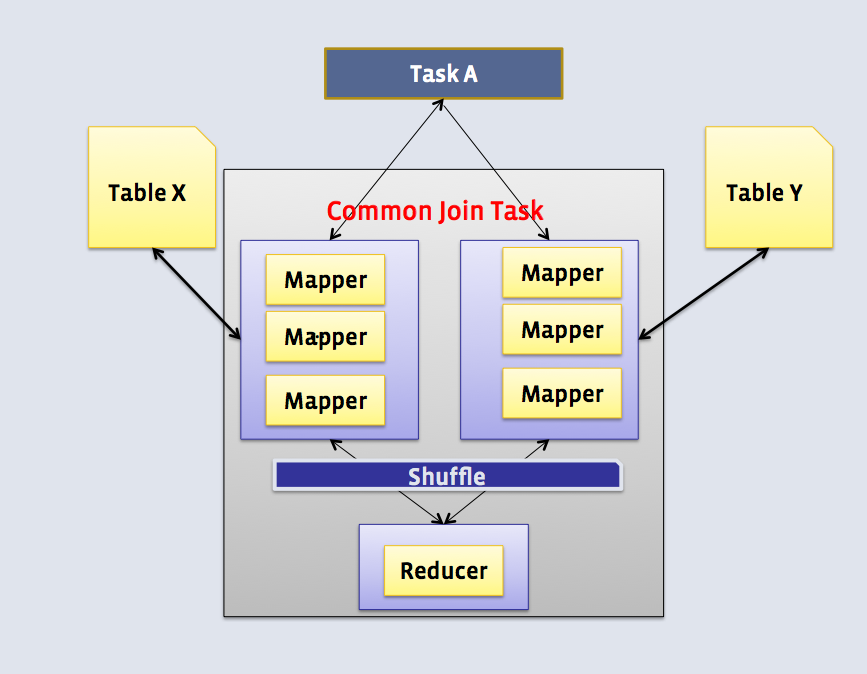
set hive.auto.convert.join.noconditionaltask.size=10000000;

Starting from Hive 0.11, hive.auto.convert.join=true by default.  
You can disable this feature by setting hive.auto.convert.join=false.  
When hive.auto.convert.join.noconditionaltask=true, if estimated size of small table(s) is smaller than hive.auto.convert.join.noconditionaltask.size,  then common join can convert to map join automatically.

Our department table is of 46 bytes and employee table is of 106 bytes as we can see in explain plan

Now we are forcefully set some lower value to below property hive.auto.convert.join.noconditionaltask.size=30

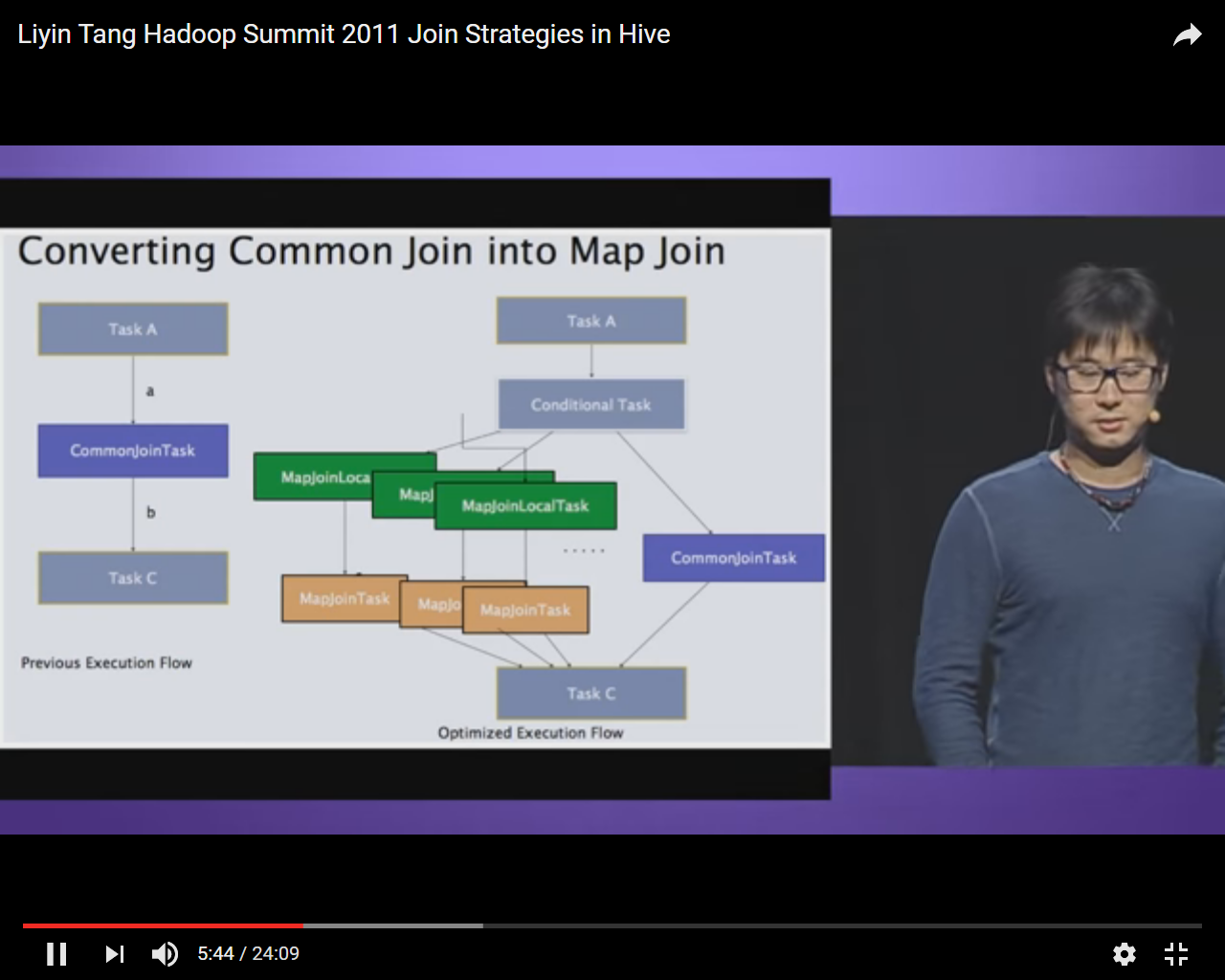
Now as we set the value as 30 bytes , now both tables are larger than 30 bytes ,if we again run the same explain plan , it would turn into common join

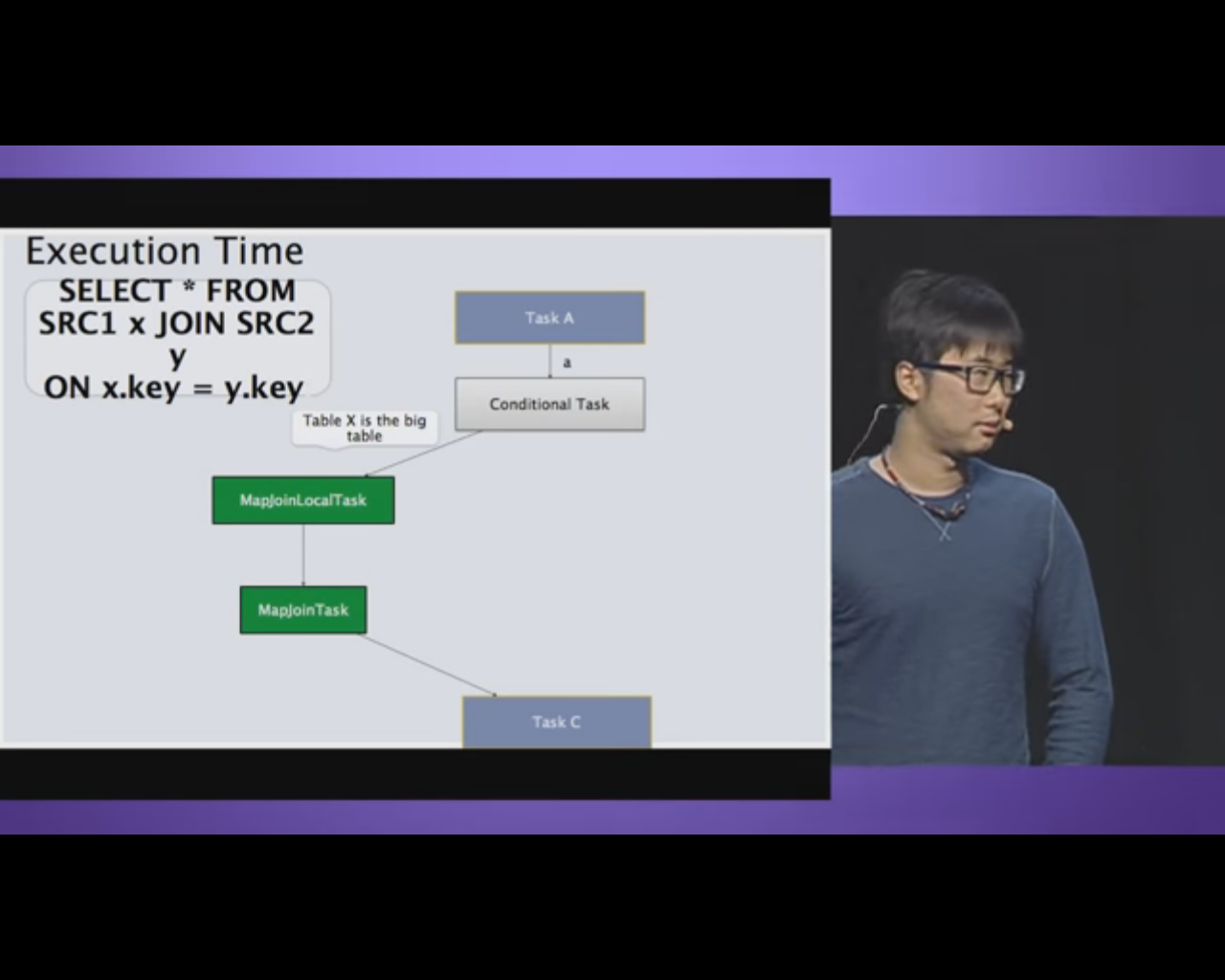
[](http://1.bp.blogspot.com/-P8MRAOLXxFM/VGvTk5oxJ8I/AAAAAAAAA4k/AOpRje2PhsQ/s1600/Screen%2BShot%2B2014-11-18%2Bat%2B3.17.16%2BPM.png)

|  |
| --- |
| hive (default)> explain select e.empname,d.deptname,d.deptid from emp e join dept d on e.deptid=d.deptid;  OK  STAGE DEPENDENCIES:  Stage-5 is a root stage , consists of Stage-6, Stage-7, Stage-1  Stage-6 has a backup stage: Stage-1  Stage-3 depends on stages: Stage-6  Stage-7 has a backup stage: Stage-1  Stage-4 depends on stages: Stage-7  Stage-1  Stage-0 depends on stages: Stage-3, Stage-4, Stage-1  STAGE PLANS:  Stage: Stage-5  Conditional Operator  Stage: Stage-6  Map Reduce Local Work  Alias -> Map Local Tables:  d  Fetch Operator  limit: -1  Alias -> Map Local Operator Tree:  d  TableScan  alias: d  filterExpr: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 46 Basic stats: COMPLETE Column stats: NONE  Filter Operator  predicate: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 46 Basic stats: COMPLETE Column stats: NONE  HashTable Sink Operator  keys:  0 deptid (type: int)  1 deptid (type: int)  Stage: Stage-3  Map Reduce  Map Operator Tree:  TableScan  alias: e  filterExpr: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 104 Basic stats: COMPLETE Column stats: NONE  Filter Operator  predicate: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 104 Basic stats: COMPLETE Column stats: NONE  Map Join Operator  condition map:  Inner Join 0 to 1  keys:  0 deptid (type: int)  1 deptid (type: int)  outputColumnNames: \_col1, \_col6, \_col7  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  Select Operator  expressions: \_col1 (type: string), \_col7 (type: string), \_col6 (type: int)  outputColumnNames: \_col0, \_col1, \_col2  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  File Output Operator  compressed: false  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  table:  input format: org.apache.hadoop.mapred.TextInputFormat  output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat  serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe  Local Work:  Map Reduce Local Work  Stage: Stage-7  Map Reduce Local Work  Alias -> Map Local Tables:  e  Fetch Operator  limit: -1  Alias -> Map Local Operator Tree:  e  TableScan  alias: e  filterExpr: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 104 Basic stats: COMPLETE Column stats: NONE  Filter Operator  predicate: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 104 Basic stats: COMPLETE Column stats: NONE  HashTable Sink Operator  keys:  0 deptid (type: int)  1 deptid (type: int)  Stage: Stage-4  Map Reduce  Map Operator Tree:  TableScan  alias: d  filterExpr: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 46 Basic stats: COMPLETE Column stats: NONE  Filter Operator  predicate: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 46 Basic stats: COMPLETE Column stats: NONE  Map Join Operator  condition map:  Inner Join 0 to 1  keys:  0 deptid (type: int)  1 deptid (type: int)  outputColumnNames: \_col1, \_col6, \_col7  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  Select Operator  expressions: \_col1 (type: string), \_col7 (type: string), \_col6 (type: int)  outputColumnNames: \_col0, \_col1, \_col2  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  File Output Operator  compressed: false  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  table:  input format: org.apache.hadoop.mapred.TextInputFormat  output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat  serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe  Local Work:  Map Reduce Local Work  Stage: Stage-1  Map Reduce  Map Operator Tree:  TableScan  alias: e  filterExpr: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 104 Basic stats: COMPLETE Column stats: NONE  Filter Operator  predicate: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 104 Basic stats: COMPLETE Column stats: NONE  Reduce Output Operator  key expressions: deptid (type: int)  sort order: +  Map-reduce partition columns: deptid (type: int)  Statistics: Num rows: 1 Data size: 104 Basic stats: COMPLETE Column stats: NONE  value expressions: empname (type: string)  TableScan  alias: d  filterExpr: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 46 Basic stats: COMPLETE Column stats: NONE  Filter Operator  predicate: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 46 Basic stats: COMPLETE Column stats: NONE  Reduce Output Operator  key expressions: deptid (type: int)  sort order: +  Map-reduce partition columns: deptid (type: int)  Statistics: Num rows: 1 Data size: 46 Basic stats: COMPLETE Column stats: NONE  value expressions: deptname (type: string)  Reduce Operator Tree:  Join Operator  condition map:  Inner Join 0 to 1  keys:  0 deptid (type: int)  1 deptid (type: int)  outputColumnNames: \_col1, \_col6, \_col7  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  Select Operator  expressions: \_col1 (type: string), \_col7 (type: string), \_col6 (type: int)  outputColumnNames: \_col0, \_col1, \_col2  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  File Output Operator  compressed: false  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  table:  input format: org.apache.hadoop.mapred.TextInputFormat  output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat  serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe  Stage: Stage-0  Fetch Operator  limit: -1  Processor Tree:  ListSink  Time taken: 0.325 seconds, Fetched: 173 row(s) |

<https://www.youtube.com/watch?v=OB4H3Yt5VWM>

|  |
| --- |
| Stage-5 - Conditional Operator ( to know the table size)  Stage-6 -Map Reduce Local Work - department  Stage-3-Map Reduce -- Map Reduce employee --- join based on deptid and create intermediate file  Stage-7 -Map Reduce Local Work employee  Stage-4-Map Reduce -- Map Reduce department --- join based on deptid and create intermediate file  Stage-1-Map Reduce -- Map Reduce employee and department on intermediate files - reduce operation and create final files |







**2. Hint "MAPJOIN" can be used to force to use map join.**  
Before using the hint, firstly make sure below parameter is set to false(Default is true in Hive 0.13).

set **hive.ignore.mapjoin.hint**=false;

Then:

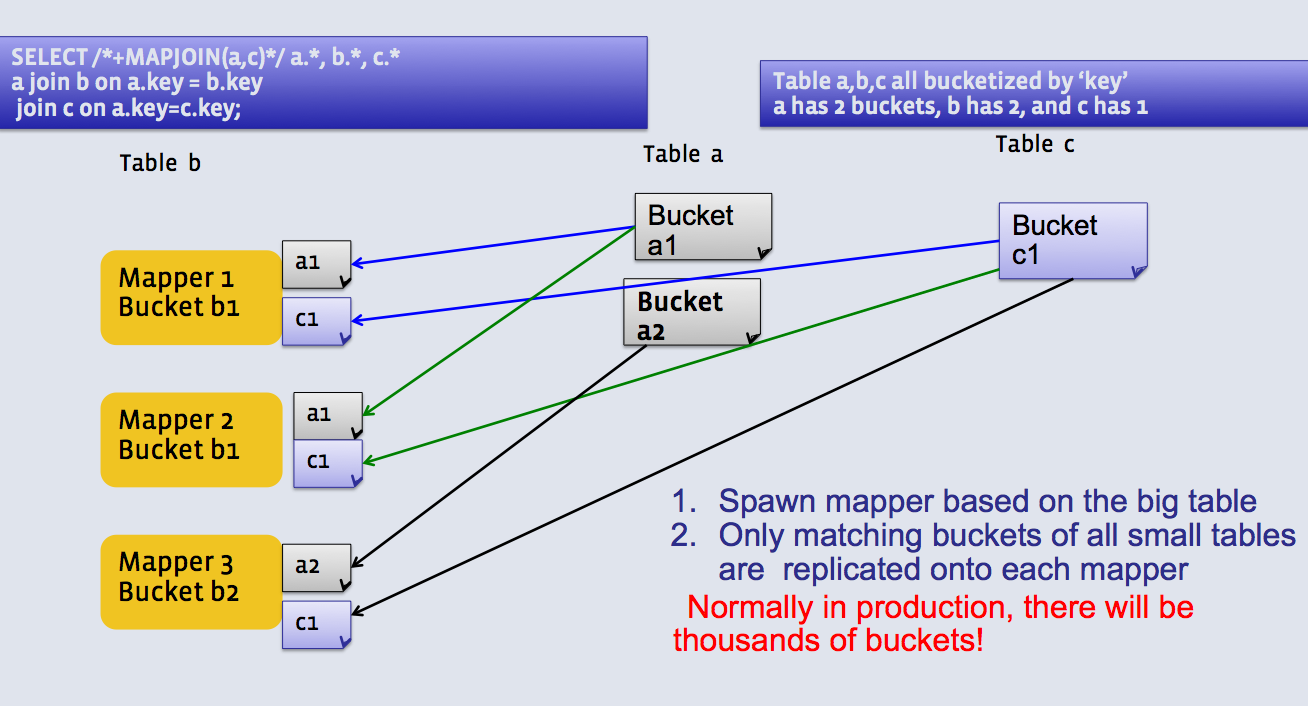
select /\*+ MAPJOIN(a) \*/ a.\* from passwords a, passwords2 b where a.col0=b.col0 ;

Now again it became map side join as we provide hint and set hive.ignore.mapjoin.hint to false

|  |
| --- |
| hive (default)> explain select /\*+ MAPJOIN(d) \*/ e.empname,d.deptname,d.deptid from emp e join dept d on e.deptid=d.deptid;  OK  STAGE DEPENDENCIES:  Stage-3 is a root stage  Stage-1 depends on stages: Stage-3  Stage-0 depends on stages: Stage-1  STAGE PLANS:  Stage: Stage-3  Map Reduce Local Work  Alias -> Map Local Tables:  d  Fetch Operator  limit: -1  Alias -> Map Local Operator Tree:  d  TableScan  alias: d  filterExpr: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 46 Basic stats: COMPLETE Column stats: NONE  Filter Operator  predicate: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 46 Basic stats: COMPLETE Column stats: NONE  HashTable Sink Operator  keys:  0 deptid (type: int)  1 deptid (type: int)  Stage: Stage-1  Map Reduce  Map Operator Tree:  TableScan  alias: e  filterExpr: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 104 Basic stats: COMPLETE Column stats: NONE  Filter Operator  predicate: deptid is not null (type: boolean)  Statistics: Num rows: 1 Data size: 104 Basic stats: COMPLETE Column stats: NONE  Map Join Operator  condition map:  Inner Join 0 to 1  keys:  0 deptid (type: int)  1 deptid (type: int)  outputColumnNames: \_col1, \_col6, \_col7  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  Select Operator  expressions: \_col1 (type: string), \_col7 (type: string), \_col6 (type: int)  outputColumnNames: \_col0, \_col1, \_col2  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  File Output Operator  compressed: false  Statistics: Num rows: 1 Data size: 114 Basic stats: COMPLETE Column stats: NONE  table:  input format: org.apache.hadoop.mapred.TextInputFormat  output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat  serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe  Local Work:  Map Reduce Local Work  Stage: Stage-0  Fetch Operator  limit: -1  Processor Tree:  ListSink  Time taken: 0.208 seconds, Fetched: 64 row(s) |

## Bucket Map Join

Join is done in Mapper only. The mapper processing bucket 1 for table A will only fetch bucket 1 of table B.



### Use case:

When all tables are:

* Large.
* Bucketed using the join columns.
* The number of buckets in one table is a multiple of the number of buckets in the other table.
* Not sorted.

### Cons:

Tables need to be bucketed in the same way how the SQL joins, so it cannot be used for other types of SQLs.

**Tips:**

**1. The tables need to be created bucketed on the same join columns and also data need to be bucketed when inserting.**  
One way is to set "hive.enforce.bucketing=true" before inserting data.  
For example:

create table b1(col0 string,col1 string,col2 string,col3 string,col4 string,col5 string,col6 string)

clustered by (col0) into 32 buckets;

create table b2(col0 string,col1 string,col2 string,col3 string,col4 string,col5 string,col6 string)

clustered by (col0) into 8 buckets;

**set hive.enforce.bucketing = true;**

From passwords insert OVERWRITE table b1 select \* limit 10000;

From passwords insert OVERWRITE table b2 select \* limit 10000;

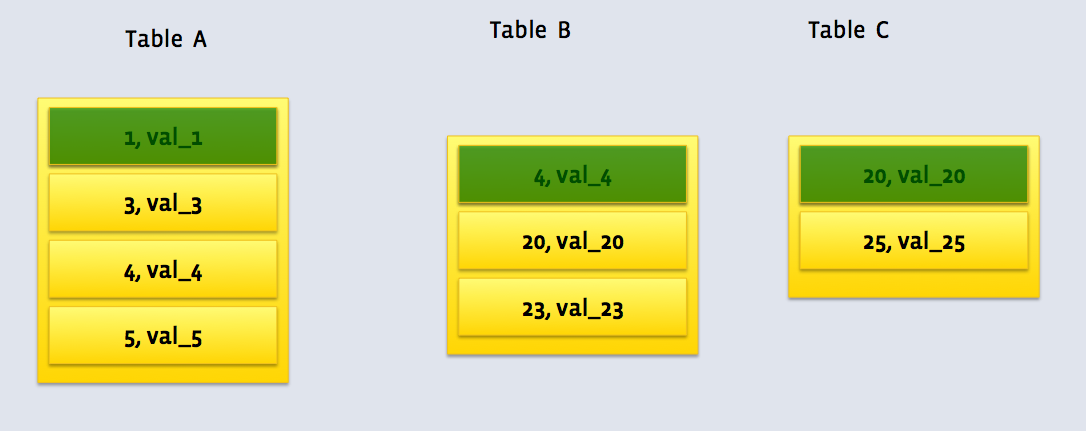
**2.  hive.optimize.bucketmapjoin must to be set to true.**

**set hive.optimize.bucketmapjoin=true;**

select /\*+ MAPJOIN(b2) \*/ b1.\* from b1,b2 where b1.col0=b2.col0 ;

## Sort Merge Bucket(SMB) Map Join

Join is done in Mapper only. The corresponding buckets are joined with each other at the mapper.



### Use case:

When all tables are:

* Large.
* Bucketed using the join columns.
* Sorted using the join columns.
* All tables have the same number of buckets.

### Cons:

Tables need to be bucketed in the same way how the SQL joins, so it cannot be used for other types of SQLs.  
Partition tables might slow down.

**xample:**

hive> explain select c1.\* from c1,c2 where c1.col0=c2.col0;

OK

STAGE DEPENDENCIES:

Stage-1 is a root stage

Stage-0 is a root stage

STAGE PLANS:

Stage: Stage-1

Map Reduce

Map Operator Tree:

TableScan

alias: c1

Statistics: Num rows: 9963904 Data size: 477218560 Basic stats: COMPLETE Column stats: NONE

Sorted Merge Bucket Map Join Operator

condition map:

Inner Join 0 to 1

condition expressions:

0 {col0} {col1} {col2} {col3} {col4} {col5} {col6}

1 {col0}

keys:

0 col0 (type: string)

1 col0 (type: string)

outputColumnNames: \_col0, \_col1, \_col2, \_col3, \_col4, \_col5, \_col6, \_col9

Filter Operator

predicate: (\_col0 = \_col9) (type: boolean)

Select Operator

expressions: \_col0 (type: string), \_col1 (type: string), \_col2 (type: string), \_col3 (type: string), \_col4 (type: string), \_col5 (type: string), \_col6 (type: string)

outputColumnNames: \_col0, \_col1, \_col2, \_col3, \_col4, \_col5, \_col6

File Output Operator

compressed: false

table:

input format: org.apache.hadoop.mapred.TextInputFormat

output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat

serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

Stage: Stage-0

Fetch Operator

limit: -1

Time taken: 0.134 seconds, Fetched: 37 row(s)

**Tips:**

**1. The tables need to be created bucketed and sorted on the same join columns and also data need to be bucketed when inserting.**  
One way is to set "hive.enforce.bucketing=true" before inserting data.  
For example:

create table c1(col0 string,col1 string,col2 string,col3 string,col4 string,col5 string,col6 string)

clustered by (col0) sorted by (col0) into 32 buckets;

create table c2(col0 string,col1 string,col2 string,col3 string,col4 string,col5 string,col6 string)

clustered by (col0) sorted by (col0) into 32 buckets;

set hive.enforce.bucketing = true;

From passwords insert OVERWRITE table c1 select \* order by col0;

From passwords insert OVERWRITE table c2 select \* order by col0;

**2. Below parameters need to set to convert SMB join to SMB map join.**

set hive.auto.convert.sortmerge.join=true;

set hive.optimize.bucketmapjoin = true;

set hive.optimize.bucketmapjoin.sortedmerge = true;

set hive.auto.convert.sortmerge.join.noconditionaltask=true;

**3.  Big table selection policy parameter "hive.auto.convert.sortmerge.join.bigtable.selection.policy" determines which table is for only streaming.**  
It has 3 values:

org.apache.hadoop.hive.ql.optimizer.AvgPartitionSizeBasedBigTableSelectorForAutoSMJ (default)

org.apache.hadoop.hive.ql.optimizer.LeftmostBigTableSelectorForAutoSMJ

org.apache.hadoop.hive.ql.optimizer.TableSizeBasedBigTableSelectorForAutoSMJ

**4. Hint "MAPJOIN" can determine which table is small and should be loaded into memory.**  
**5. Small tables are read on demand which means not holding small tables in memory.**  
**6. Outer join is supported.**