* Predicate Push Down (PPD) optimization

|  |  |  |  |
| --- | --- | --- | --- |
| [favorite](http://stackoverflow.com/questions/21150859/is-there-a-performance-difference-for-these-two-hive-queries-joining-two-tables) | Suppose both table A and B have ds as their partition key.  **Method 1**  SELECT \*  FROM A JOIN B ON A.userid=B.userid  WHERE A.ds='2014-01-01' AND B.ds='2014-01-01'  **Method 2**  SELECT \*  FROM (  SELECT \* FROM A WHERE A.ds='2014-01-01'  ) JOIN (  SELECT \* FROM B WHERE B.ds='2014-01-01'  ) ON  A.userid=B.userid  Will the 2nd query be faster?  I'm wondering how WHERE and JOIN works in Hive. Is a where clause applied to the source table before the join when possible (as long as the clause contains only one table alias, like the ones above), or is it always applied only after the tables are joined (e.g. A.userid > B.userid has to be applied after the join)?  [hql](http://stackoverflow.com/questions/tagged/hql) [hive](http://stackoverflow.com/questions/tagged/hive)   |  |  | | --- | --- | | [share](http://stackoverflow.com/q/21150859)[improve this question](http://stackoverflow.com/posts/21150859/edit) | asked Jan 16 '14 at 0:00  [https://www.gravatar.com/avatar/3fb3e6187718a7dfb12b504930cbdf7b?s=32&d=identicon&r=PG](http://stackoverflow.com/users/2015256/xiangpeng-zhao)  [Xiangpeng Zhao](http://stackoverflow.com/users/2015256/xiangpeng-zhao)  **98**7 | |
|  |  |

1 Answer

|  |  |
| --- | --- |
|  | Your question is actually about predicate pushdown in hive. Well in the case above the execution will be exactly the same as hive will push the predicate A.ds='2014-01-01' AND B.ds='2014-01-01' to the mappers before the join.  In a more general case, the JOIN (inner join) is actually pretty easy and can be summed up to : **If it can push, it will push.** It can push the predicate when there is only one table involved (where a.x > 1) and can't push when there is more than 1 table involved (A.userid > B.userid) as the mapper reads a split of one of the tables only..  The more complex part is OUTER JOIN and furtunelty explained very clearly [here](https://cwiki.apache.org/confluence/display/Hive/OuterJoinBehavior" \l "OuterJoinBehavior-PredicatePushdownRules).  P.S. predicate pushdown is controlled by hive.optimize.ppd which is true by default. |

## Definitions

|  |  |
| --- | --- |
| Preserved Row table | The table in an Outer Join that must return all rows.  For left outer joins this is the Left table, for right outer joins it is the Right table, and for full outer joins both tables are Preserved Row tables. |
| Null Supplying table | This is the table that has nulls filled in for its columns in unmatched rows.  In the non-full outer join case, this is the other table in the Join. For full outer joins both tables are also Null Supplying tables. |
| During Join predicate | A predicate that is in the JOIN ON clause.  For example, in 'R1 join R2 on R1.x = 5' the predicate 'R1.x = 5' is a During Join predicate. |
| After Join predicate | A predicate that is in the WHERE clause. |

## Predicate Pushdown Rules

The logic can be summarized by these two rules:

1. During Join predicates cannot be pushed past Preserved Row tables.
2. After Join predicates cannot be pushed past Null Supplying tables.

This captured in the following table:

|  | **Preserved Row Table** | **Null Supplying Table** |
| --- | --- | --- |
| Join  Predicate | Case J1:  Not Pushed | Case J2:  Pushed |
| Where  Predicate | Case W1:  Pushed | Case W2:  Not Pushed |

See [Examples](https://cwiki.apache.org/confluence/display/Hive/OuterJoinBehavior" \l "OuterJoinBehavior-Examples) below for illustrations of cases J1, J2, W1, and W2.

### **Hive Implementation**

Hive enforces the rules by these methods in the SemanticAnalyzer and JoinPPD classes:

Rule 1: During QBJoinTree construction in Plan Gen, the parseJoinCondition() logic applies this rule.  
Rule 2: During JoinPPD (Join Predicate PushDown) the getQualifiedAliases() logic applies this rule.

## Examples

Given Src(Key String, Value String) the following Left Outer Join examples show that Hive has the correct behavior.

### **Case J1: Join Predicate on Preserved Row Table**

explain

select s1.key, s2.key

from src s1 left join src s2 on s1.key > '2';

STAGE DEPENDENCIES:

Stage-1 is a root stage

Stage-0 is a root stage

STAGE PLANS:

Stage: Stage-1

Map Reduce

Alias -> Map Operator Tree:

s1

TableScan

alias: s1

Reduce Output Operator

sort order:

tag: 0

value expressions:

expr: key

type: string

s2

TableScan

alias: s2

Reduce Output Operator

sort order:

tag: 1

value expressions:

expr: key

type: string

Reduce Operator Tree:

Join Operator

condition map:

Left Outer Join0 to 1

condition expressions:

0 {VALUE.\_col0}

1 {VALUE.\_col0}

filter predicates:

0 {(VALUE.\_col0 > '2')}

1

handleSkewJoin: false

outputColumnNames: \_col0, \_col4

Select Operator

expressions:

expr: \_col0

type: string

expr: \_col4

type: string

outputColumnNames: \_col0, \_col1

File Output Operator

compressed: false

GlobalTableId: 0

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

### **Case J2: Join Predicate on Null Supplying Table**

explain

select s1.key, s2.key

from src s1 left join src s2 on s2.key > '2';

STAGE PLANS:

Stage: Stage-1

Map Reduce

Alias -> Map Operator Tree:

s1

TableScan

alias: s1

Reduce Output Operator

sort order:

tag: 0

value expressions:

expr: key

type: string

s2

TableScan

alias: s2

Filter Operator

predicate:

expr: (key > '2')

type: boolean

Reduce Output Operator

sort order:

tag: 1

value expressions:

expr: key

type: string

Reduce Operator Tree:

Join Operator

condition map:

Left Outer Join0 to 1

condition expressions:

0 {VALUE.\_col0}

1 {VALUE.\_col0}

handleSkewJoin: false

outputColumnNames: \_col0, \_col4

Select Operator

expressions:

expr: \_col0

type: string

expr: \_col4

type: string

outputColumnNames: \_col0, \_col1

File Output Operator

compressed: false

GlobalTableId: 0

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

### **Case W1: Where Predicate on Preserved Row Table**

explain

select s1.key, s2.key

from src s1 left join src s2

where s1.key > '2';

STAGE PLANS:

Stage: Stage-1

Map Reduce

Alias -> Map Operator Tree:

s1

TableScan

alias: s1

Filter Operator

predicate:

expr: (key > '2')

type: boolean

Reduce Output Operator

sort order:

tag: 0

value expressions:

expr: key

type: string

s2

TableScan

alias: s2

Reduce Output Operator

sort order:

tag: 1

value expressions:

expr: key

type: string

Reduce Operator Tree:

Join Operator

condition map:

Left Outer Join0 to 1

condition expressions:

0 {VALUE.\_col0}

1 {VALUE.\_col0}

handleSkewJoin: false

outputColumnNames: \_col0, \_col4

Select Operator

expressions:

expr: \_col0

type: string

expr: \_col4

type: string

outputColumnNames: \_col0, \_col1

File Output Operator

compressed: false

GlobalTableId: 0

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

### **Case W2: Where Predicate on Null Supplying Table**

explain

select s1.key, s2.key

from src s1 left join src s2

where s2.key > '2';

STAGE PLANS:

Stage: Stage-1

Map Reduce

Alias -> Map Operator Tree:

s1

TableScan

alias: s1

Reduce Output Operator

sort order:

tag: 0

value expressions:

expr: key

type: string

s2

TableScan

alias: s2

Reduce Output Operator

sort order:

tag: 1

value expressions:

expr: key

type: string

Reduce Operator Tree:

Join Operator

condition map:

Left Outer Join0 to 1

condition expressions:

0 {VALUE.\_col0}

1 {VALUE.\_col0}

handleSkewJoin: false

outputColumnNames: \_col0, \_col4

Filter Operator

predicate:

expr: (\_col4 > '2')

type: boolean

Select Operator

expressions:

expr: \_col0

type: string

expr: \_col4

type: string

outputColumnNames: \_col0, \_col1

File Output Operator

compressed: false

GlobalTableId: 0

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