Description of physical algorithms

May 23, 2014

1 Filter

Operator filters given rows and output only rows satisfying condition. Output doesn't have to be sorted same way as input.
Input and output columns are the same and they are numbered from 0.
Example:
$Filter(double, double, int) - > (double, double, int)$
$f(condition = "OP\_LOWER(OP\_double\_CONSTANT("4.8"), 1)");$

2 Filter keeping order

Operator filters given rows and output only rows satisfying condition.
Output has to be sorted same way as input.
Input and output columns are the same and they are numbered from 0.
Compiler assumes that this version of filter is slower than regular filter. Example:
$FilterKeepingOrder(double, double, int) - > (double, double, int)$
$f(condition = "OP\_LOWER(OP\_double\_CONSTANT("4.8"), 1)");$

3 Hash group

Operator uses hash table for grouping data by columns given in *groupBy* parameter. It also computes aggregate functions specified in $functions$.
Input columns are numbered from 0.
Output columns consists from grouped columns and computed aggregate functions in the same order as in parameters. Example:
$HashGroup(string, string, int) - > (string, int, int)$
$g(groupBy = "1", functions = "count(), max(2)");$

4 Sorted group

Operator groups data by columns given in *groupBy* parameter. It also computes aggregate functions specified in $functions$.
Input columns are numbered from 0 and input data are sorted by grouped columns. Sort order is arbitrary.
Output columns consists from grouped columns and computed aggregate functions in the same order as in input parameters.
Output has to be sorted the same way as input.
Example:
$SortedGroup(string, string, int) - > (string, int, int)$
$g(groupBy = "1", functions = "count(), max(2)");$

5 Column operations

This is extended projection operator, it eliminates certain columns and computes new ones.
Input columns are numbered from 0.
Output is specified in parameter *out*.
It if contains number operator, copies input to output, otherwise it computes new column.
Example:
$ColumnsOperations(int, int, int, int, int) - > (int, int, int, double)$
$c(out = "0, 3, 4, OP\_TIMES(1, OP\_MINUS(OP\_double\_CONSTANT(1), 2))");$

6 Cross join

Operator computes Cartesian product from given inputs.
Numbering columns from first input is specified in $left$ parameter.
Numbering columns from second input is specified in *right* parameter.
Join outputs only columns given in *out* argument.
Example:
$CrossJoin(string, int)(int, string) - > (string, string)$
$c(left = "0, 1", right = "2, 3", out = "0, 3");$

7 Hash join

Operator computes equijoin using hash table. First input will be stored in hash table.
Numbering columns from first input is specified in $left$ parameter.
Numbering columns from second input is specified in *right* parameter.

Join outputs only columns given in out argument.
Condition is given in parameters $leftPartOfCondition$ and $rightPartOfCondition$.
Example:
$HashJoin(int, int)(int, int, int, int)->(int, int, int, int, int, int)$
$h(left = "0, 1", right = "2, 3, 4, 5", out = "0, 1, 2, 3, 4, 5", leftPartOfCondition = "0, 1", rightPartOfCondition = "5, 2")$;
This example computes join with condition $(0 == 5)\&\&(1 == 2)$.

8 Merge equijoin

Operator computes equijoin from given sorted inputs.
Numbering columns from first input is specified in $left$ parameter.
Numbering columns from second input is specified in $right$ parameter.
Join outputs only columns given in out argument.
Condition is given in parameters $leftPartOfCondition$ and $rightPartOfCondition$ and they also contain information how are inputs sorted.
Example:
$MergeEquiJoin(int)(int)->(int, int))$
$m(left = "0", right = "1", out = "0, 1", leftPartOfCondition = "0 : D", rightPartOfCondition = "1 : D")$;
This example computes join with condition $(0 == 1)$. First input is sorted by column number 0 descending and the second input is sorted by 1 descending.

9 Merge non equijoin

This operator is currently not supported. Represents merge join joining by condition $a_1 < b < a_2$, where a_1 and a_2 are columns from 1st input, b is from 2nd input.

10 Hash anti join

Operator computes equiantijoin using hash table. Second input will be stored in hash table. Numbering columns from first input is specified in $left$ parameter.
Numbering columns from second input is specified in $right$ parameter.
Join outputs only columns given in out argument.
Operator copies to output only rows from first input for which doesn't exist row in second input satisfying given condition.
Condition is given in parameters $leftPartOfCondition$ and $rightPartOfCondition$.
Example:
$HashAntiJoin(int)(int)->(int)$
$h(left = "0", right = "1", out = "0", leftPartOfCondition = "0", rightPartOfCondition = "1")$;
This example computes antijoin with condition $(0 == 1)$.

11 Merge anti join

Operator computes equiantijoin from given sorted inputs.
Numbering columns from first input is specified in $left$ parameter.
Numbering columns from second input is specified in $right$ parameter.
Join outputs only columns given in out argument.
Operator copies to output only rows from first input for which doesn't exist row in second input satisfying given condition. Condition is given in parameters $leftPartOfCondition$ and $rightPartOfCondition$ and they also contain information how are inputs sorted.
Example:
$MergeAntiJoin(int)(int)->(int)$
$m(left = "0", right = "1", out = "0", leftPartOfCondition = "0 : D", rightPartOfCondition = "1 : D");$
This example computes join with condition $(0 == 1)$. First input is sorted by column number 0 descending and the second input is sorted by 1 descending.

12 Table scan

Operator reads whole table given in $name$ and reads columns specified in attribute $columns$. Example:
$TableScan()->(int, int, int, int)$
$t(name = "lineitem", columns = "l\_orderkey, l\_shipdate, l\_extendedprice, l\_discount");$

13 Scan And Sort By Index

Operator reads whole table given in $name$ using $index$ and reads columns specified in attribute $columns$. Example:
$ScanAndSortByIndexScan()->(string, string, int)$
$s(name = "people", index = "index", columns = "user\_name, country, parameter");$

14 Index Scan

Operator reads part of table given in $name$ using $index$ and reads columns specified in attribute $columns$.
Operator reads only rows satisfying condition given in attribute $condtion$.
Example:
$IndexScan()->(int, int)$
$i(name = "customer", index = "index2", columns = "c\_custkey, c\_mktsegment", condition = "OP\_EQUALS(1, OP\_string\_CONSTANT(SEGMENT))");$

15 Sort

Operator sorts given table. Input and output columns are the same and they are numbered from 0.
Parameter *sortedBy* specifies by which columns is table sorted
Parameter *sortBy* specifies by which columns should table be sorted
Example:
$SortOperator(int, int) - > (int, int)$
$s(sortedBy = "0", sortBy = "1 : D");$

16 Union

Operator copies the first input to output and append the data from second input.
Numbering columns from the first input is given in the $left$ parameter.
Numbering columns from the second input is given in the *right* parameter.
Numbering columns from the output is given in the *out* parameter.
Example:
$Union(int, string)(string, int) - > (int, string)$
$u(left = "0, 1", right = "1, 0", out = "0, 1");$