Table Pushdown Query Optimizer

Objective:

Design of a new query rewrite mechanism called Table Pushdown. The query must have a subquery within it and the outer table must have a join with the inner table specified in the subquery. The rewrite **transforms the query such that the outer table is pushed inside the subselect**.[1]

Outcome:

By rewriting the query as such, the ultimate goal is to generate bushy plans faster. The immediate outcome is to generate queries which reduce the overall query runtime.

Conditions under which Rewrite is triggered[1]-

- 1. There must be a Subquery
 - a. If true, the subquery must be fetching data from only 1 table.
 - b. There must be a join condition between one column in the Subquery relation and another in the Outer Table.
- 2. The Outer Table should satisfy the following
 - a. There is a join with the Subquery on the primary key column[* User's query must ensure that the join column is either unique or is PK for the table as currently Presto Connectors/Engine do not provide a way to reason about uniqueness of a column].
 - b. There should not be any GROUP BY/ORDER BY or similar clauses.
- 3. If the above two are satisfied, then having a GROUP BY clause in the inner Subquery on the join column will not affect the semantics of the grouping.

*Special Comment Format=

/* #distinct@ tab1 = col1,col2#*/

REGEX-

State after the Rewrite has been successful-

- 1. The Outer Table has been moved inside the Subquery.
- 2. The join clause on the Subquery column and the Outer Table primary key column is moved inside the Subquery. The actual column name in the Subquery is used here, not the alias.
- 3. All predicate clauses on the outer table are also moved into the Subquery(WHERE clauses).
- 4. If the outer table has any other join conditions, these are replaced by the subquery join column name as the outer table has been moved inside and is now not known outside.

A sample Query from TPC-H (Q17[2]) is shown below in its original form and its rewritten form satisfying the above steps-

Original Query-

SELECT Sum(l_extendedprice) / 7.0 AS avg_yearly FROM lineitem, part, (SELECT 0.2 * Avg(l_quantity) AS s_avg, l_partkey AS s_partkey FROM lineitem GROUP BY l_partkey) sub WHERE p_partkey = l_partkey AND p_brand = 'Brand#43' AND p_container = 'LG PACK' AND p_partkey = s_partkey AND l_quantity < s_avg;

Rewritten Query-

SELECT Sum(l_extendedprice) / 7.0 AS avg_yearly FROM lineitem, (SELECT 0.2 * Avg(l_quantity) AS s_avg, l_partkey AS s_partkey FROM lineitem, **part** WHERE p_brand = 'Brand#43' AND p_container = 'LG PACK' AND p_partkey = l_partkey GROUP BY l_partkey) sub WHERE s_partkey = l_partkey AND l_quantity < s_avg;

Design Diagrams-

In Presto, the Optimizer can be implemented in two ways-

One is by implementing the PlanOptimizer interface which is a visitor based mechanism and calls various visitXXX() methods to perform the query optimization. Examples- PredicatePushdown.

The other way is to implement the Rule<T> interface where T is the type of the node on which this is called. This method is based on matching patterns, and creating new nodes and returning to the calling function. Here, we perform the optimization using the second method. We implement the Rule for TablePushdown on the JoinNode as this is the first node that is to be evaluated and encountered while traversing the plan tree from the root OutputNode-

A new class TablePushdown, which implements Rule<JoinNode> is to be created.

Next, we present the the class diagram of the TablePushdown class along with the methods that will be called.



The apply() method enters at the first JoinNode encountered from the root of the Plan Tree. Within the apply method, <u>verifyJoinCondition()</u> identifies if the conditions for the rule are satisfied by the query-

- verifyJoinOnUniqueColumn()- Within the JoinNode, we get the join condition and extract the column aliases for the join condition and check if the column in the outer node has unique values or not by comparing against the user input column name.
- 2. <u>verifyPresenceOfJoinsAndGroupBy()</u>- This method verifies two things
 - a. Both children's paths from the JoinNode cannot have a JoinNode
 - b. Only one path from the JoinNode has the GROUP BY clause (subquery table).

Once these conditions are verified, the rewrite mechanism is executed by calling the **planRewriter**() method-

1. First, the outer table is shifted to its intended position in the plan tree (effectively pushed down into the subquery and the join conditions and other associated values are updated).



2. The inner subquery table's join conditions are updated and any preexisting aggregations are pulled above the new inner join as follows-



We also propose to have a session parameter for enabling/disabling this Rule called "optimizer.push-table-through-subquery".

Finally, the sequence of operations that should be performed to rewrite the query are shown as

follows-

: doAnalyzeQuery() ED REC ERED REC 2 : logicalPlanner.pla REC STERED REC STERED		1 1		Memo	Context	
REC ^A ERED 2: logicalPlanner.pla REC STERED REC STERED	JNREGISI			NREGISTEI	RED UNREGIST	
REC <u>2:logicalPlanner.pla</u> REC STERED REC STERED REC STERED	JNREGIST	FERED UNRE	EGISTERED U	NREGISTEI	RED UNREGIST	
iregistered (Iregistered (Iregistered (REGIST		EGISTERED U	NREGISTEI	RED UNREGIST	FERED UNREG
REGISTERED (JNJEGIST		EGISTERED U	NREGISTEI		
IL CICICICE C	INFGIST		GISTERED U	NREGISTE		
REGISTERED (: plahStatement()		GISTERED U	NREGISTE	RED UNREGIST	
	1SanityChecker.vali	idateIntermediatePlan()	GISTERED U	NREGISTEI	RED UNREGIST	FERED UNREG
REGISTERED U	JNREGIST	6 : optimizer optimize()	GISTERED U	NREGISTE	RED UNREGIST	
REGISTERED U	INREGIST		GISTER 7:new N	Nemo()	RED UNREGIS	
REGISTERED (JNREGIST		GISTERED U	8 : new Context()	RED UNREGIST	
REGISTERED L	JNREGIST		GIST 9: exploreGrou	()REGISTEI	RED UNFLEGIST	
REGISTERED U	JNREGIST				RED UNREGIST	
REGISTERED U						
	JNREGIST		11 : context.memo.ge	tNode()	RED UNREGIS	
REGISTERED L	JNREGIST		GISTE12 : transform	NREGISTEI	RED UNREGIST	
REGISTERED U	JNREGIST	FERED UNRE	EGISTE <u>RED U</u>	NREGIST ¹³	rule.apply()	
	JNREGIST		EGISTERED U	NREGISTE		
	INREGIST		GISTERED U	NREGISTE	RED UNREGIST	
REGISTERED (JNREGIS7	FERED UNRE	GISTERED U	NREGISTEI	RED UNREGIST	FERED UNREG
VPLEORE					RE	GISTERE
	arser()	IREGISTE	RED UNF		RED UNRE	GISTERE
VRE2: verifyAnds	SetTableDistin	ctColumns()	RED UNR	3 : rule.apply		
<u> </u>		IDE LISTE			MING ONKE	GISTERE
VREGISTER		ILLUIS IL	RED UNF	EGISTE		GISTERE
NREGISTER		IREGISTE	4 : getUniqueColum	nTableMap()		GISTEREL GISTEREL GISTEREL
NREGISTEF NREGISTEF NREGISTEF		IREGISTE	4 : getUniqueColum	nTableMap()		GISTEREL GISTEREL GISTEREL GISTEREL
NREGISTEF NREGISTEF NREGISTEF		IRE GISTE IREGISTE IREGISTE	4 : getUniqueColum	nTableMap()	5 : vetifyJojnConditio	GISTEREI GISTEREI GISTEREI SISTEREI
NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE	4 : getUniqueColum RED UNI	nTableMap() EGISTE	5 : vetifyJojnConditie	GISTERE GISTERE GISTERE () GISTERE JniqueColumn()
NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE IREGISTE	4 : getUniqueColum RED UNI RED UNI	nTableMap()	5 : vetifyJonConditi 6 - ventyJoinOnL	GISTERE GISTERE GISTERE Ons() GISTERE JniqueColumn()
NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE	4 : getUniqueColum RED UNI RED UNI RED UNI RED UNI	EGISTE EGISTE EGISTE EGISTE EGISTE	5 : vetifyJoInConditio	GISTERE GISTERE OISTERE JniqueColumn() GISTERE ndGroupBy()
NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE	4 : getUniqueColum RED UNI RED UNI RED UNI	nTableMap()	5 : vetifyJonConditie 6 - ventyJoinOnL ifyPresenceOfJoinsAr	GISTERE GISTERE OSTERE Dins() JniqueColumn() GISTERE IdGroupBy()
NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE	4 : getUniqueColum RED UNI RED UNI RED UNI RED UNI RED UNI	EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE	5 : vetityJonConditie 6 - ventyDoinOnL ifyPresence@fJoinsAr	GISTERE GISTERE GISTERE JniqueColumn() GISTERE dGroupBy() RE
NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE	RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI	EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE	5 : vetifyJoInConditio 6 : vetifyJoInConditio 6 : vetifyJoinOnL ifyPresence@fJoinsAr 8 : planRewrite lateOuterTableAndInn	GISTERE GISTERE DISTERE JniqueColumn() GISTERE ndGroupBy() RE () ISTERE erTablePath()
NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE	4 : getUniqueColum RED UN RED UN RED UN RED UN RED UN RED UN	nTableMap() EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE	5 : vetifyJonConditie 6 - venifyJoinOnL ifyPresenceOfJoinsAr 8 - planRewriter lateOuterTableAndInn	GISTERE GISTERE DISTERE JniqueColumn() GISTERE dGroupBy() r() ISTERE erTablePath() GISTERE
NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE	4 getUniqueColum RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI	EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE	5 : vetityJonConditie 6 - venttyJoinOnU ifyPresenceOfJoinsAr 9 - planRewrite lateOuteTTableAndInn 10 - updateInnerTa	GISTERE GISTERE Ons() GISTERE IniqueColumn() GISTERE r() ISTERE erTablePath() GISTERE ble() TERE
NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE	RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI	EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE	5 : vetifyJoInConditie 6	GISTERE GISTERE Dns() JniqueColumn() GISTERE ndGroupBy() erTablePath() GISTERE ible() ctures()
NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE	4 : getUniqueColum RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI	nTableMap() EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE	5 : veityJonConditi 6 - verityJoinOnL ityPresence@fJoinsAr 8 - planRewrite lateOuteTableAndInn 10 - updateInnerTa 11 : clearAllDataStru	GISTERE GISTERE DISTERE DISTERE DISTERE DISTERE dGroupBy() c() ISTERE erTablePath() GISTERE ctures() ERE
NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE	4 getUniqueColum RED UNI RED UNI	EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE	5 : vetityJonConditie 5 : vetityJonConditie 6 : venttyDoinOnL ifyPresenceOfJoinsAr 9 : planRewriter lateOuterTableAndInn 10 : updateInnerTa 11 : chertAllDataStru	GISTERE GISTERE GISTERE Dons() JniqueColumn() GISTERE erTablePath() CISTERE ctures() GISTERE
NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE	A : getUniqueColum RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI RED UNI	EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE	5 : vetifyJoInConditie 6ventyJoinOnU ifyPresenceOfJoinsAr 8planRewriter lateOuterTableAndInn 10tpdateInnerTa 11 : ctearAilDataStru	GISTERE GISTERE GISTERE JniqueColumn() GISTERE adGroupBy() erTablePath() GISTERE ble() ctures() ERE GISTERE GISTERE
NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF NREGISTEF		IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE IREGISTE	4 : getUniqueColum RED UNI RED UNI	nTableMap() EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE EGISTE	5 : vetifyJonConditi 6 - verifyJoinOnL ifyPresence@fJoinsAr 8 - planRewrite lateOuterTableAndInn 10 - updateInnerTa 11 : clearAllDataStru	GISTERE GISTERE DISTERE DISTERE DISTERE DISTERE dGroupBy() CISTERE DISTERE CUIPS() CISTERE GISTERE GISTERE

Tasks not handled as part of this feature design-

- 1. Recursive Subquery Table Pushdown.
- 2. Table Pushdown in the presence of Multiple Outer table with inner table join

References:

- 1. https://15721.courses.cs.cmu.edu/spring2017/papers/15-optimizer2/a8-sen.pdf
- 2. http://www.tpc.org/tpc documents current versions/pdf/tpc-h v2.18.0.pdf