List of Planner Operators

This section lists the operators used by the ClustrixDB Planner to find an optimal execution plan. For more on using and interpreting EXPLAIN, see Understanding the Xpand Explain Output.

Operators

ClustrixDB has a streaming model which streams rows starting from containers (tables and indexes) and through a graph of operators. ClustrixDB has a rich set of operators that occasionally increase as more functionality and optimizations are added. We use one place for documenting our operators, logical and physical. The physical operators show up in explain output and logical ones do not, so this is a superset of operators you will see.

Common Operators

These are the common operators one will see in the explain output from MySQL prompt.

Operator	Arguments	Details	Example
display	(REFS [output_cast]) INPUT_ROW	Display output to user. output_cast: Optional argument, if present display should promoter values for mysql protocol REFS : schema references to output INPUT_ROW : rows	
stream_combine	() INPUT_ROW	Take an input that generates separate streams (such as an index_scan that must distribute) and combine it into a single stream. This means that operators below the stream_combine will run in parallel. INPUT_ROW := bulk operator for separate input paths	
stream_merge	() INPUT_ROW	Similar to stream_combine, take an input that generates separate streams and combine it into a single stream, but this operator preserves the ordering of the input when combining. The sort keys are picked up from the opt_ctx. INPUT_ROW := bulk operator for separate input paths	
index_scan	(NAMESPACE RELOID REPOID COLS . PROPS) EQUAL LOW HIGH NULLF?	Scan the representation. Namespaces are unique per relation. NAMESPACE := vs32 RELOID := object ID of relation REPOID := object ID of index COLS := list of columns needed PROPS := list of 0 or more of: read_reverse, read_all, read_latest EQUAL := range denoting values to constrain in both low and high (range equal ...) LOW := additional low constraint. (range open/closed ...) HIGH := high constraint; similar to LOW NULLF := NULL filters see ap_select.ct	
table_scan	(NAMESPACE OID COLS . PROPS) . EXPR	NAMESPACE := vs32 that represents which namespace OID := object ID of table COLS := list of columns needed PROPS := list of 0 more of: update, use_index, force_index EXPR := list of constraints to apply to the output	
filter	() INPUT_ROW EXPR...	Apply filter to input. EXPR := item operator; null, 0, and false are filtered out INPUT_ROW := bulk operator providing rows	
row_limit	CALC_FOUND_ROWS INPUT_ROW LVAL OVAL	Limit the number of rows produced. This operator can stop previous input operators from producing rows once it has enough rows. CALC_FOUND_ROWS := boolean INPUT_ROW := bulk operator providing rows LVAL := item operator for limit expression OVAL := optional item operator for offset expression	
row_count	NAME INPUT_ROW	Counts the number of rows produced. Assigns to NAME.	
user_const	VALUE	User provided constant. VALUE := constant.	
const	VALUE	VALUE := constant.	

ref	(NAMESPACE . NAME)	NAMESPACE := vs32 of the namespace NAME := string of the column name	
param	(VALUE)	VALUE := u32 denoting which prepared parameter it is	
range	MODE EXPR...	MODE := closed, open, equal EXPR := expression for each column in the range	
force_prop	PROP INPUT_ROW	Physical operator that forces a property on its input. PROP is one of the following (sort KEYS) := force sort over list of KEYS (limit LIMVAL OFFVAL) := force limit	
func	OP EXPRS...	Function OP := symbol of the operator EXPRS := 1 or more item operators	
mtable_scan	(NAMESPACE SET? NELEMS . TYPES) MTABLE	Scan a memory table as if it were a real table. NAMESPACE := valuespace where the columns appear SET? := whether this is an ordered set, or just a bag of blobs NELEMS := The number of elements in the mtable TYPES := oidvs for types of the columns MTABLE := item operator that provides the mtable blob	
mtable_build	(COLS) . EXPRS	Build a memory table. IN sets are sometimes converted to memory tables. COLS := list of column names to use (in the 0 namespace) EXPRS := list of (tuple () (param...) (param...) ...) Denotes psuedo-mtable that should be built from previous constants.	
mtable_find	(NAMESPACE . TYPES) INPUT MTABLE EQUAL	Find a value in Memory table. NAMESPACE := valuespace where the columns appear TYPES := oidvs for types of the columns INPUT := bulk input MTABLE := item operator that provides the mtable blob EQUAL := range denoting values to constrain in both low and high (range equal ...)	
table_lock	(RELOID LOCK) EXPR	Grabs one or more table locks for the duration of the subexpression. RELOID := oid of table to lock LOCK := Lock symbol EXPR := row generator that is passed through	
table_locks	NUMLOCKS EXPR	Placeholder for properly costing table locks. NUMLOCKS := vs32 of the number of table locks to grab EXPR := row generator	
pk_lock	(NAMESPACE OID COLS LOCK) INPUT . PKREFS	Used to take fine grained locks NAMESPACE := vs32 OID := object ID of table COLS := columns to bind LOCK := Row lock symbol INPUT := bulk operator PKREFS := list of refs indicating primary key values to lock	
pk_lookup	(NAMESPACE RELOID REPOID COLS . PROPS) INPUT PKSARGE	Used to lookup which fine grained lock to take. NAMESPACE := vs32 RELOID := object ID of relation REPOID := object ID of index COLS := columns to bind/rebind INPUT := bulk operator PKSARGE := range denoting exact sarge on baserep	
table_write	(OID REFS . PROPS) INPUT_ROW	OID := object ID of table REFS := schema references containing values to use for each column PROPS := list of 0 or more of: ignore, replace, impdflt, serialize, multirow, lock_dst_x, lock_dst_s INPUT_ROW := bulk operator that produces rows columns are taken from input row schema	
table_delete	(OID REFS) INPUT_ROW	OID := object ID of table REFS := schema references containing values to use for each column INPUT_ROW := bulk operator that produces rows columns are taken from input row schema	
table_update	(OID OLD_REFS NEW_REFS . PROPS) INPUT_ROW	OID := object ID of table OLD_REFS := schema references representing old column values NEW_REFS := schema references representing column values PROPS := list of 0 or more of: ignore, serialize INPUT_ROW := bulk operator that produces rows with old/new column values	

Aggregate Operators

Aggregate Operators perform GROUP BY and DISTINCT operations.

Operator	Arguments	Details	Example
aggregate	(KEY AGG_EXPR ONEOF) INPUT_ROW	Logical operator, implemented as one of the physical aggregates	-
hash_aggregate	(KEY AGG_EXPR ONEOF) INPUT_ROW	use a sigma, produces unsorted output	
sigma_aggregate	(KEY AGG_EXPR ONEOF) INPUT_ROW	use a sigma, produces output sorted on KEY	
stream_aggregate	(KEY AGG_EXPR ONEOF) INPUT_ROW	non-blocking, maintains incoming sort order (if any)	
dist_stream_aggregate	(KEY AGG_EXPR ONEOF) INPUT_ROW	stream_aggregate that occurs on multiple nodes (partitioned input)	
hash_aggregate_partial	(KEY AGG_EXPR ONEOF) INPUT_ROW	hash_aggregate that occurs on multiple nodes	
hash_aggregate_combine	(KEY AGG_EXPR ONEOF) INPUT_ROW	hash_aggregate that consumes results from _partial	
distinct	_KEYS INPUT_ROW	Logical distinct operator, implemented as one of physical distinct operators.	
sigma_distinct	_KEYS INPUT_ROW	produces output having only one row per distinct set of values for _KEYS.	
sigma_distinct_combine	_KEYS INPUT_ROW	does a sigma distinct on results from sigma_distinct_partial.	
sigma_distinct_partial	_KEYS INPUT_ROW	sigma_distinct that occurs on multiple nodes	
sort	(KEY) INPUT_ROW	Logical sort operator. Force an ordering on the input. KEY := list of (STYPE REF) if STYPE := sort type (ASC, DSC, IFNULL) REF := schema reference else STYPE:= sort type GROUP REF := list of schema references	
sigma_sort	() INPUT_ROW [LIMIT]	Force an ordering on the input using a sigma. Context has sort keys so don't show up in input always.	

Notation	Explanation
schema reference	(NAMESPACE . NAME) NAMESPACE: vs32 that represents which namespace (AUTOGEN = 0) NAME:string of the column name
KEY	List of schema references to use as keys we are grouping by
_KEYS	keys to distinct by
AGG_EXPR	List of (NAME FUNC DISTINCT REF)
NAME	For aggregates, this means new schema name for this value in the AUTOGEN (= 0) namespace
FUNC	Aggregate function as a symbol (sum, max, min, count)
DISTINCT	Does this aggregate function operate on distinct input?
REF	Schema reference
ONEOF	List of schema references to pass for aggregate oneof
INPUT_ROW	Bulk operator providing rows
LIMIT	(optional) item operator representing max # rows

Join Operators

Join operators perform a left of nested loop join. Since joins are distributed, hash_join etc. don't fit in with the streaming model of ClustrixDB.

Operator	Format	Details	Example

inner_join	STRAIGHT? INPUT1 INPUT2 . _EXPR	Logical operator, implemented as a physical joins	-
nljoin	ONE_TO_ONE INPUT1 INPUT2	Nested-loop join. Each row of INPUT1 is read and forwarded to correct next place for INPUT2	
nljoin_par	ONE_TO_ONE INPUT1 INPUT2	Same as nljoin, differs in certain parallelization properties, primarily exists to make mechanics of optimizer work.	
left_join	STRAIGHT? INPUT1 INPUT2 . _EXPR	STRAIGHT is always true, so INPUT1 is read before input 2. Also if there is no matching row in INPUT2, NULL value is substituted.	
left_semi_join	(PROBEREF) INPUT1 INPUT2 . _EXPR	Perform a left semi-join between INPUT1 and INPUT2, returning any rows from the left side for which the right side returns 1 or more	
item_join	JTYPE INPUT_ROW IT_EXPR	item_join is a placeholder for subquery expressions where item operators contain bulk operators. IT_EXPR is item operator.	
msjoin	ONE_TO_ONE INPUT1 INPUT2	Merge sort nested-loop join. This is similar to nljoin, but is able to preserve sort order.	
outer	() INPUT1	Right half of a left outer join.	
outer_fwd	() INPUT1 [INPUT2]	Right half of a left outer join. This form of outer performs the outer calculation on a single node, which makes it much more efficient, but can only be applied in some cases. The optional INPUT2 bulk operator does not affect cardinality.	
dual	()	Generate a single row.	

Notation	Explanation
STRAIGHT	force reading the left relation before the right
INPUT1	bulk operator for the left side
INPUT2	bulk operator for the right side
EXPR	0 or more item operators representing join constraints
PROBEREF	optional ref indicating whether the semi-join succeeded
ONE_TO_ONE	whether INPUT2 only has a cardinality of 1 or not
JTYPE	is oneof (inner) - attach via inner_join (semi PROBEREF) - attach via left_semi_join

Other Operators

These are the remaining operators, some are logical and some physical.

Operator	Format	Details	Example
genoid	NAME INPUT_ROW	Create a unique id. NAME := name to assign a new oid INPUT_ROW := bulk operator, one oid is generated for each row	
lock_choice	NUMLOCKS INPUT	Enforces either all row locks or all table locks for all reads in INPUT. NUMLOCKS := vs32 of the number of locks to grab INPUT := bulk operator for the rows to lock	
any	() BULK_EXPR EXPR	BULK_EXPR := rows to look through EXPR := test to apply for matches	
check_scalar check_scalar_expr	() INPUT_ROW REF INPUT_ROW	Errors out if input contains more than 1 row. INPUT_ROW := bulk operator providing rows REF := reference to return	

compute	(NAMES TYPE) INPUT_ROW EXPR0 ...	NAMES := a list of new schema names for these values (in the AUTOGEN namespace) TYPE := normal, fake, serialized INPUT_ROW := bulk operator for rows EXPRn := item operators for the expressions named in NAMES. Expressions (and names) are ordered from least dependant to most, so expressions later in the list can reference names defined earlier, but not vice-versa.	
error	(RESULT_CODE STRING ARG1...) INPUT_ROW		
exists	() BULK_EXPR	BULK_EXPR := rows to look through	
ferel_row	(NS FEREL TYPE)	NS := namespace FEREL := ferel (for column description) TYPE := node type to create (t_input, etc)	
multi_distinct	((NS1 NUMKEY . COLSET1) ...) INPUT	NS1 := namespace for first sigma NUMKEY := colindex for # of keys to distinct on COLSET1 := list of refs for first sigma ... INPUT := bulk operator for row generator	
parallel_apply	() SOURCE . INPUT	SOURCE := bulk operator to apply INPUTs against INPUT := 1 or more bulk operators that take SOURCE as input	
read_sigma	(NS . COLS)	NS := namespace for sigmal COLS := list of column names	
tree_filter	() INPUT_ROW EXPR...	A special type of filter that pre-evaluates its expression(s), and then only executes the input if the expression(s) evaluates to true. This op thus requires that its expressions have no dependencies on the input. EXPR := item operator; null, 0, and false are filtered out INPUT_ROW := bulk operator providing rows	
tuple	() COL1...	COL1... := item operator expression for each column Used to represent multicolumn comparisons (in, =, etc)	
union_all	(REFS1 REFS2 REFSOUT) INPUT_ROW1 INPUT_ROW2	REFS1 := schema references used from INPUT_ROW1 REFS2 := schema references used from INPUT_ROW2 REFSOUT := schema references representing UNION display columns INPUT_ROW1 := bulk operator providing rows INPUT_ROW2 := bulk operator providing rows	
var_assign	(SCOPE NAME) VALUE	SCOPE := global/session/user NAME := variable name VALUE := value to assign variable to	
var_read	(SCOPE NAME TYPE)	SCOPE := global/session/user NAME := variable name TYPE := oidtype of variable (only for user)	