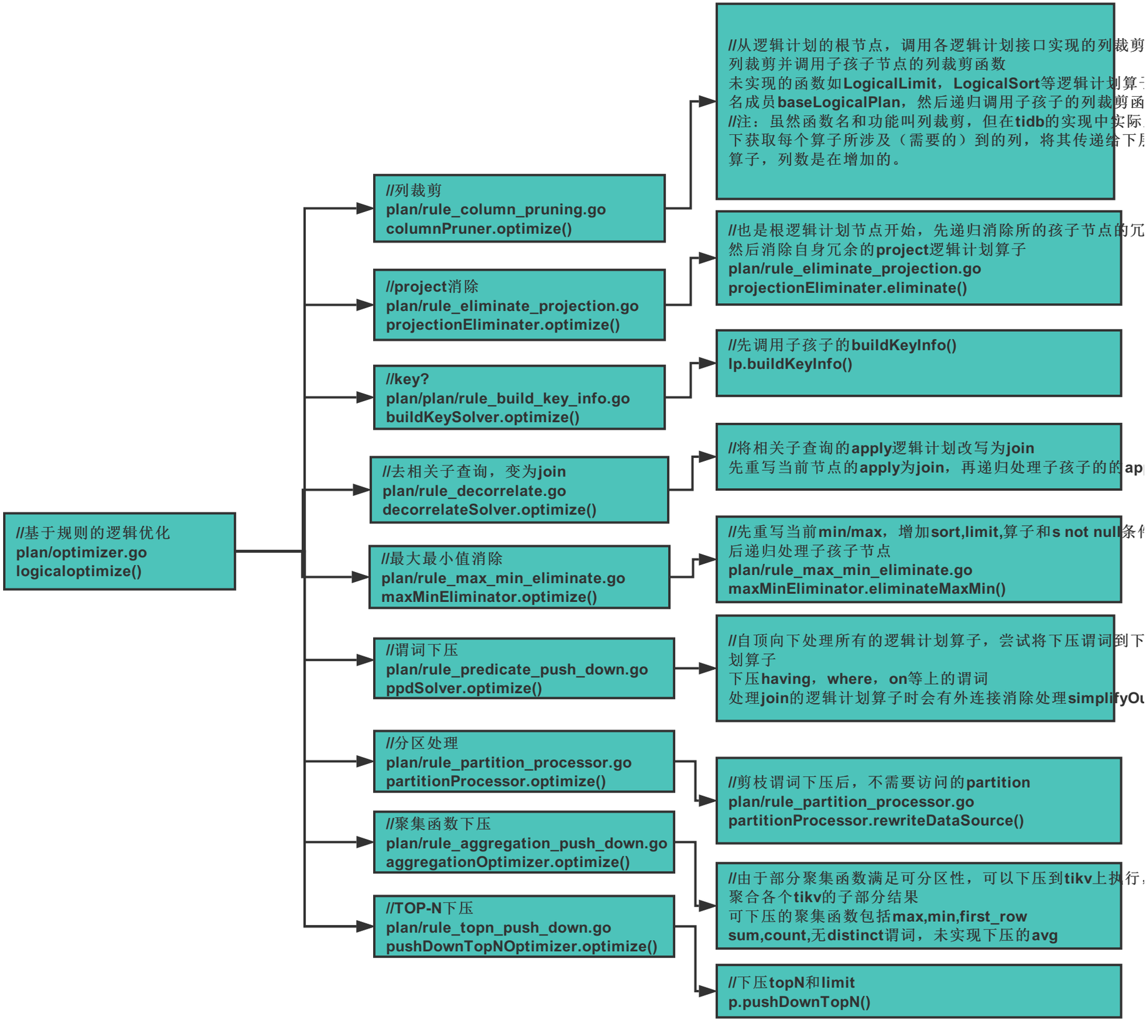
//基于规则的逻辑优化
plan/optimizer.go
logicaloptimize()
//列裁剪
plan/rule_column_pruning.go
columnPruner.optimize()
//从逻辑计划的根节点，调用各逻辑计划接口实现的列裁剪
列裁剪并调用子孩子节点的列裁剪函数
未实现的函数如LogicalLimit，LogicalSort等逻辑计划算
名成员baseLogicalPlan，然后递归调用子孩子的列裁剪函
//注：虽然函数名和功能叫列裁剪，但在tidb的实现中实际
下获取每个算子所涉及（需要的）到的列，将其传递给下
算子，列数是在增加的。
//project消除
plan/rule_eliminate_projection.go
projectionEliminater.optimize()
//也是根逻辑计划节点开始，先递归消除所的孩子节点的冗
然后消除自身冗余的project逻辑计划算子
plan/rule_eliminate_projection.go
projectionEliminater.eliminate()
//key?
plan/plan/rule_build_key_info.go
buildKeySolver.optimize()
//先调用子孩子的buildKeyInfo()
lp.buildKeyInfo()
//去相关子查询，变为join
plan/rule_decorrelate.go
decorrelateSolver.optimize()
//将相关子查询的apply逻辑计划改写为join
先重写当前节点的apply为join，再递归处理子孩子的的ap
//最大最小值消除
plan/rule_max_min_eliminate.go
maxMinEliminator.optimize()
//先重写当前min/max，增加sort,limit,算子和s not null条
后递归处理子孩子节点
plan/rule_max_min_eliminate.go
maxMinEliminator.eliminateMaxMin()
//谓词下压
plan/rule_predicate_push_down.go
ppdSolver.optimize()
//自顶向下处理所有的逻辑计划算子，尝试将下压谓词到下
划算子
下压having，where，on等上的谓词
处理join的逻辑计划算子时会有外连接消除处理simplifyOu
//分区处理
plan/rule_partition_processor.go
partitionProcessor.optimize()
//剪枝谓词下压后，不需要访问的partition
plan/rule_partition_processor.go
partitionProcessor.rewriteDataSource()
//聚集函数下压
plan/rule_aggregation_push_down.go
aggregationOptimizer.optimize()
//由于部分聚集函数满足可分区性，可以下压到tikv上执行，
聚合各个tikv的子部分结果
可下压的聚集函数包括max,min,first_row
sum,count,无distinct谓词，未实现下压的avg
//TOP-N下压
plan/rule_topn_push_down.go
pushDownTopNOptimizer.optimize()
//下压topN和limit
p.pushDownTopN()