Introduction to Spark SQL & Catalyst

Takuya UESHIN

Spark Meetup 2014/09/08(Mon)

Who am I?

- Takuya UESHIN
  - @ueshin
  - github.com/ueshin
- Nautilus Technologies, Inc.
- A Spark contributor

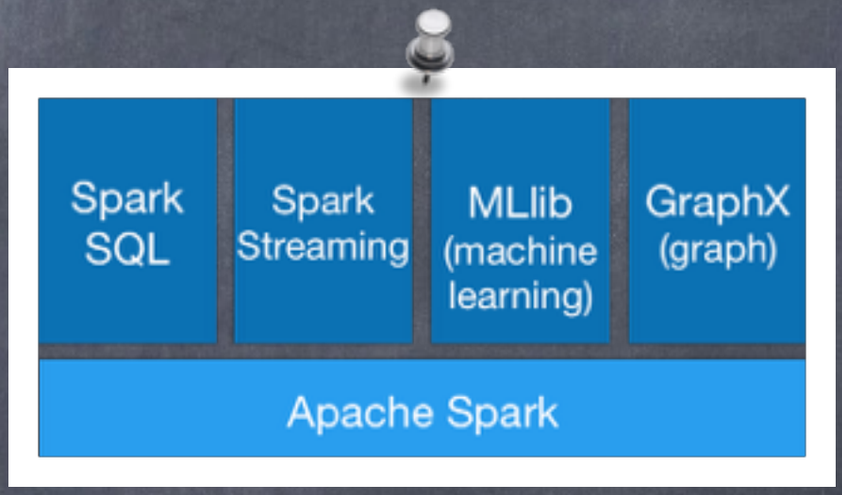
Agenda

- What is Spark SQL?
- Catalyst in depth
- SQL core in depth
- Interesting issues
- How to contribute

What is Spark SQL?

What is Spark SQL?

- Spark SQL is one of Spark components.



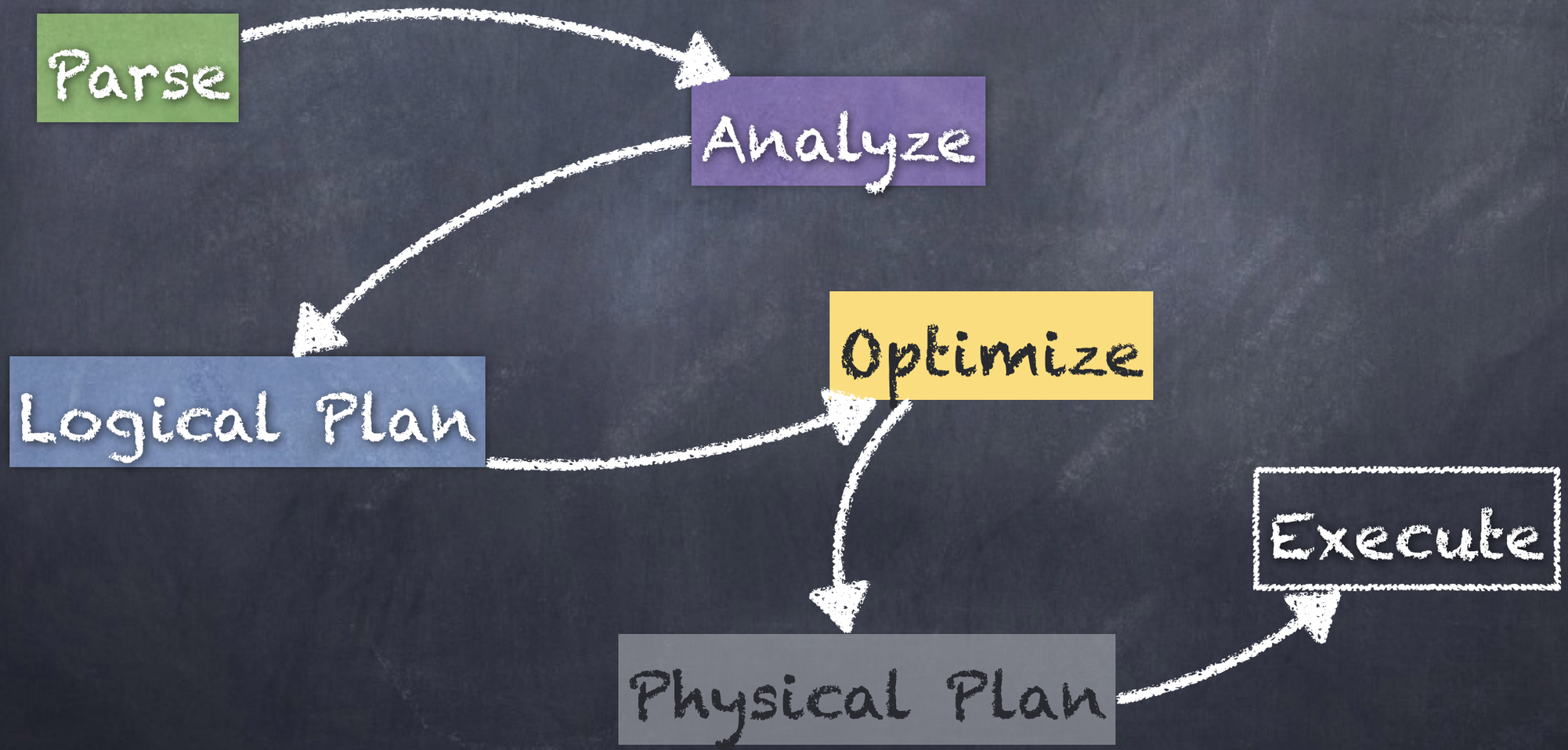
  - Executes SQL on Spark
  - Builds SchemaRDD like LINQ
- Optimizes execution plan.

What is Spark SQL?

- Catalyst provides a execution planning framework for relational operations.
- Including:
  - SQL parser & analyzer
  - Logical operators & general expressions
  - Logical optimizer
  - A framework to transform operator tree.

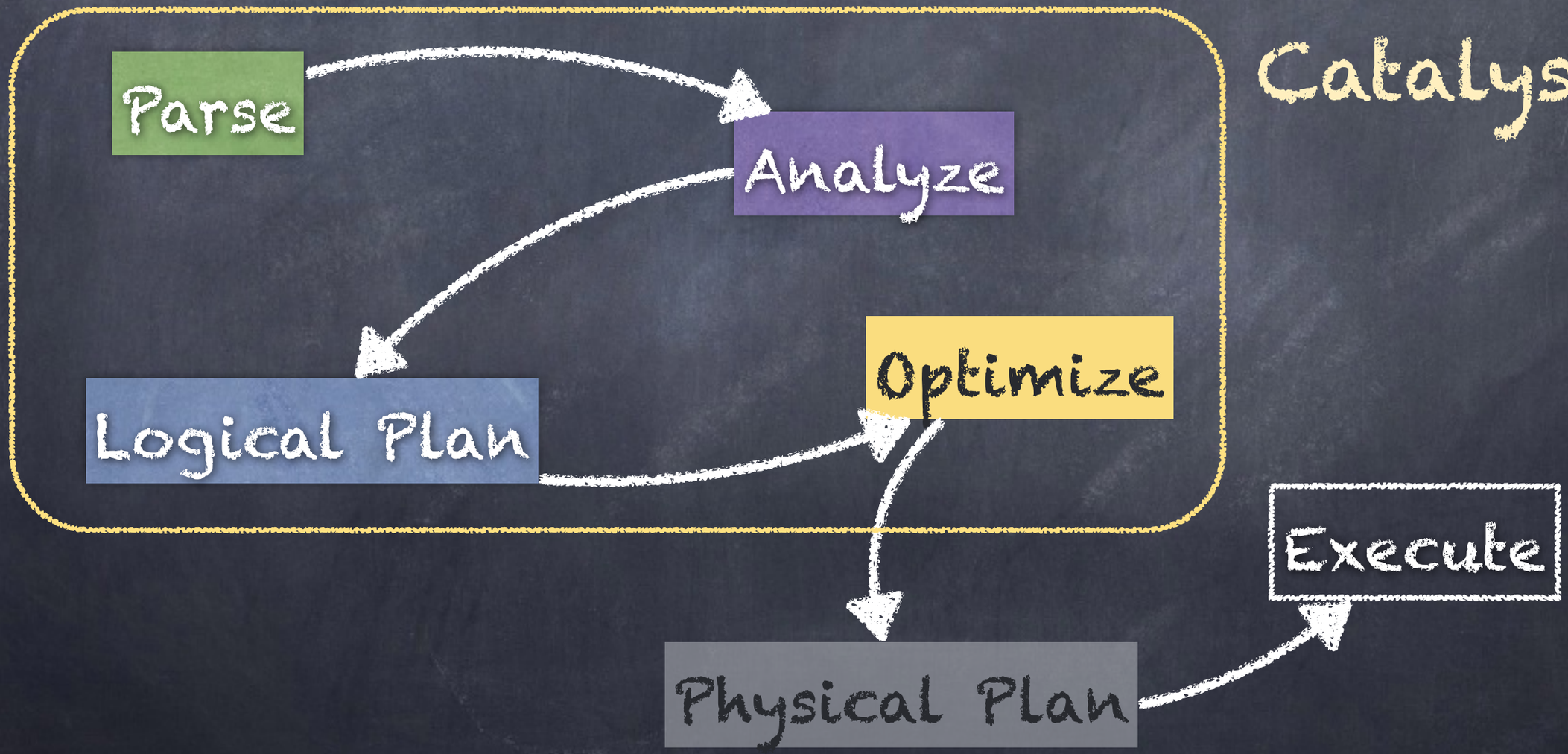
What is Spark SQL?

- To execute query needs some steps.



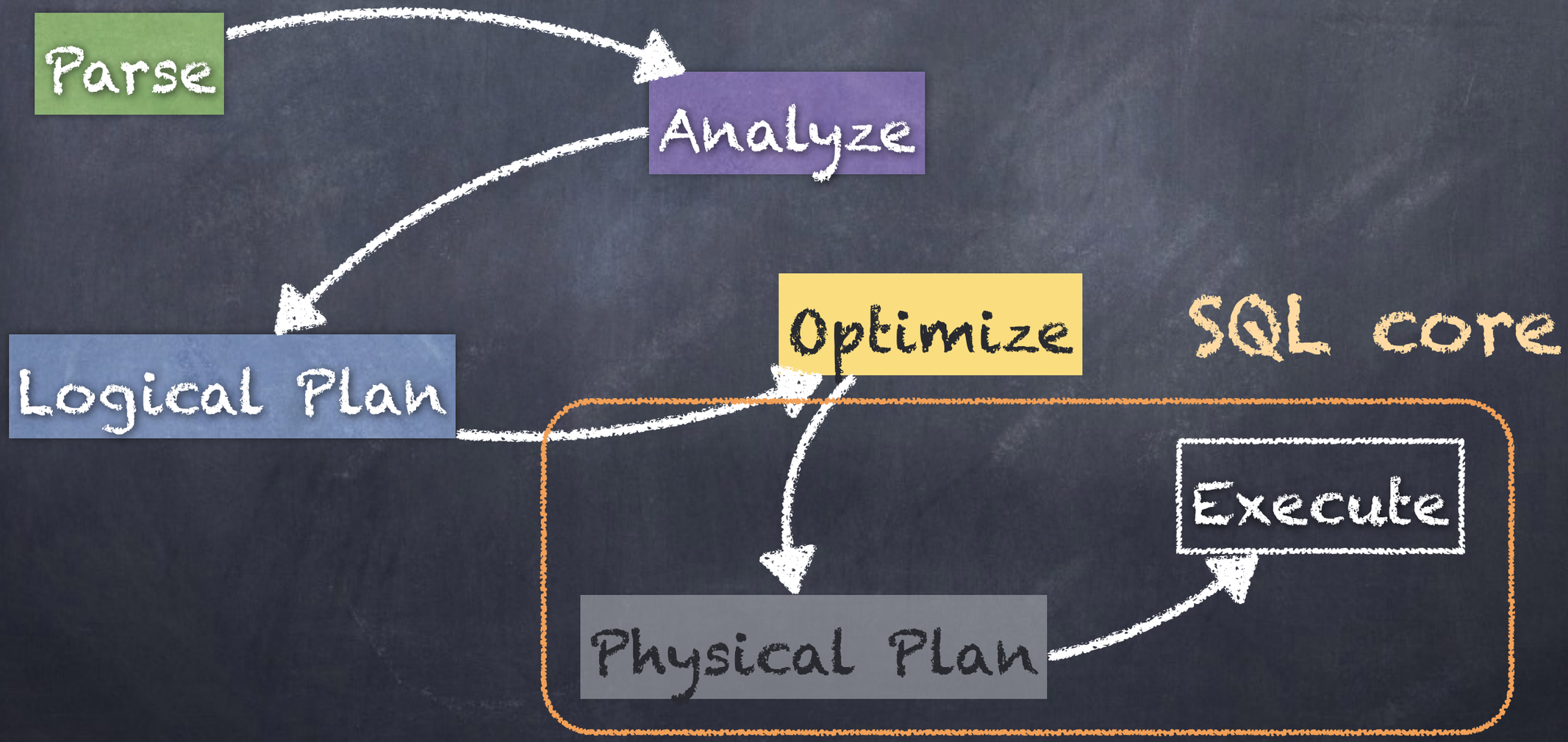
What is Spark SQL?

- To execute query needs some steps.



What is Spark SQL?

- To execute query needs some steps.



Catalyst in depth

Catalyst in depth

- Provides a execution planning framework for relational operations.
  - Row & DataType's
  - Trees & Rules
  - Logical Operators
  - Expressions
  - Optimizations

Row & DataType's

- o.a.s.sql.catalyst.types.DataType
  - Long, Int, Short, Byte, Float, Double, Decimal
  - String, Binary, Boolean, Timestamp
  - Array, Map, Struct
- o.a.s.sql.catalyst.expressions.Row
  - Represents a single row.
  - Can contain complex types.

Trees & Rules

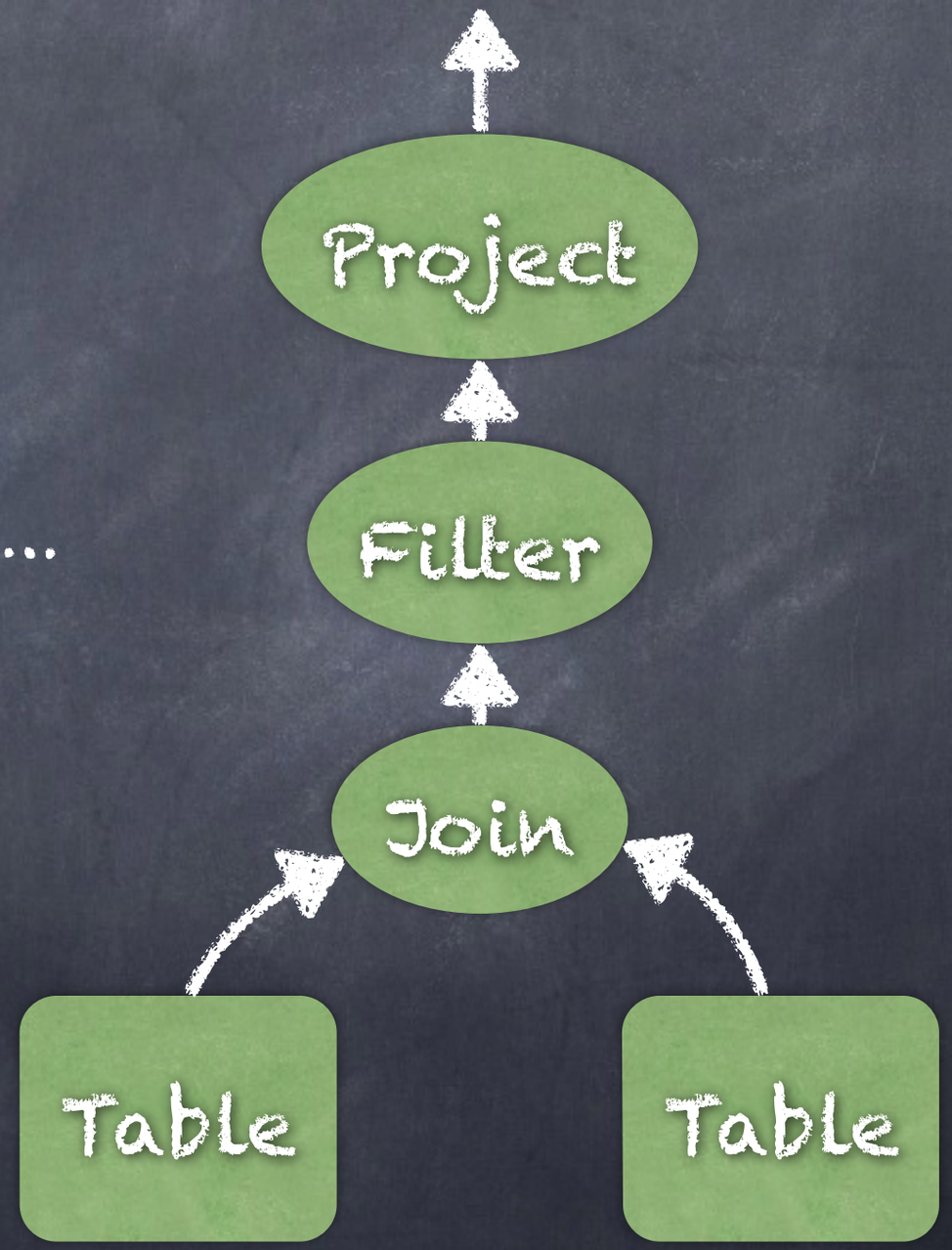
- o.a.s.sql.catalyst.trees.TreeNode
- Provides transformations of tree.
  - foreach, map, flatMap, collect
  - transform, transformUp, transformDown
- Used for operator tree, expression tree.

Trees & Rules

- o.a.s.sql.catalyst.rules.Rule
  - Represents a tree transform rule.
- o.a.s.sql.catalyst.rules.RuleExecutor
  - A framework to transform trees based on rules.

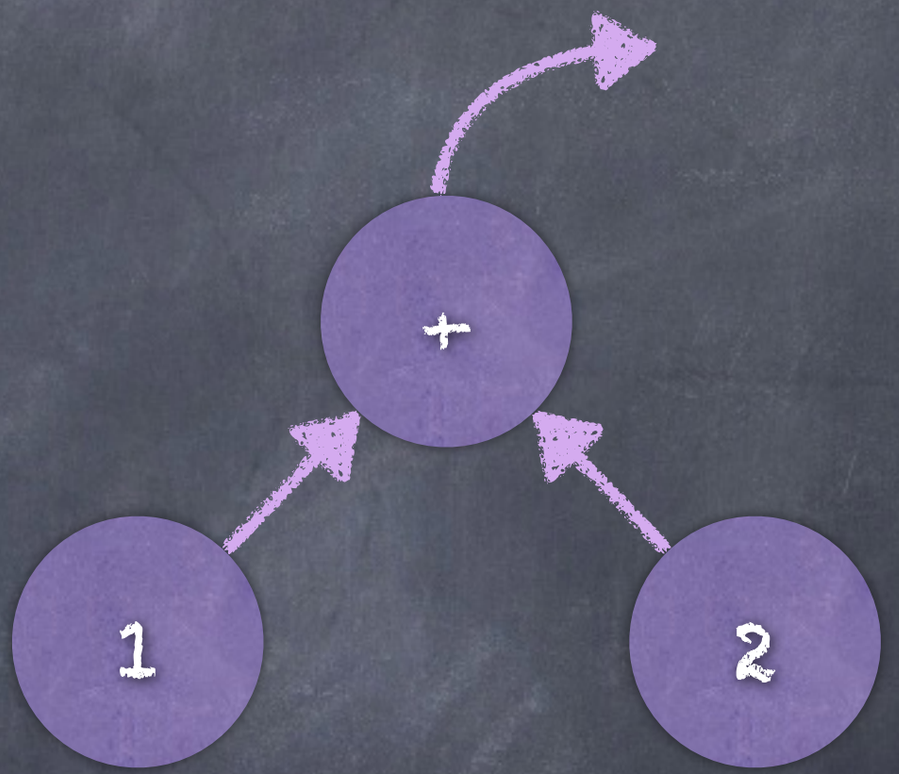
Logical Operators

- Basic Operators
  - Project, Filter, ...
- Binary Operators
  - Join, Except, Intersect, Union, ...
- Aggregate
- Generate, Distinct
- Sort, Limit
- InsertInto, WriteToFile



Expressions

- Literal
- Arithmetics
  - UnaryMinus, Sqrt, MaxOf
  - Add, Subtract, Multiply, ...
- Predicates
  - EqualTo, LessThan, LessThanOrEqual, GreaterThan, GreaterThanOrEqual
  - Not, And, Or, In, If, CaseWhen



Expressions

- Cast
- GetItem, GetField
- Coalesce, IsNull, IsNotNull
- StringOperations
  - Like, Upper, Lower, Contains, StartsWith, EndsWith, Substring, ...

Optimizations

- ConstantFolding
  - NullPropagation
  - ConstantFolding
  - BooleanSimplification
  - SimplifyFilters
- FilterPushdown
  - CombineFilters
  - PushPredicateThroughProject
  - PushPredicateThroughJoin
  - ColumnPruning

Optimizations

- NullPropagation, ConstantFolding
  - Replace expressions that can be evaluated with some literal value to the value.
  - ex)
    - 1 + null => null
    - 1 + 2 => 3
    - Count(null) => 0

Optimizations

- BooleanSimplification
  - Simplifies boolean expressions that can be determined.
  - ex)
    - false AND $right => false
    - true AND $right => $right
    - true OR $right => true
    - false OR $right => $right
    - If(true, $then, $else) => $then

Optimizations

- SimplifyFilters
  - Removes filters that can be evaluated trivially.
- ex)
  - Filter(true, child) => child
  - Filter(false, child) => empty

Optimizations

- CombineFilters
  - Merges two filters.
  - ex)
    - Filter($fc, Filter($nc, child)) =>
      Filter(AND($fc, $nc), child)

Optimizations

- PushPredicateThroughProject
  - Pushes Filter operators through Project operator.
  - ex)
    - Filter('i === 1, Project('i, 'j, child))
      =>
      Project('i, 'j, Filter('i === 1, child))

Optimizations

- PushPredicateThroughJoin
  - Pushes Filter operators through Join operator.
  - ex)
    - Filter("left.i".attr === 1, Join(left, right) =>
      Join(Filter('i === 1, left), right)

Optimizations

- ColumnPruning
  - Eliminates the reading of unused columns.
  - ex)
    - Join(left, right, LeftSemi,
      "left.id".attr === "right.id".attr) =>
      Join(left, Project('id, right), LeftSemi)

SQL core in depth

SQL core in depth

- Provides:
  - Physical operators to build RDD
  - Conversion from Existing RDD of Product to SchemaRDD support
  - Parquet file read/write support
  - JSON file read support
  - Columnar in-memory table support

SQL core in depth

- o.a.s.sql.SchemaRDD
  - Extends RDD[Row].
- Has logical plan tree.
- Provides LINQ-like interfaces to construct logical plan.
  - select, where, join, orderBy, ...
- Executes the plan.

SQL core in depth

- o.a.s.sql.execution.SparkStrategies
  - Converts logical plan to physical.
  - Some rules are based on statistics of the operators.

SQL core in depth

- Parquet read/write support
  - Columnar storage format for Hadoop
- Reads existing Parquet files.
  - Converts Parquet schema to row schema.
- Writes new Parquet files.
  - Currently DecimalType and TimestampType are not supported.

SQL core in depth

- JSON read support
  - Loads a JSON file (one object per line)
  - Infers row schema from the entire dataset.
    - Giving the schema is experimental.
    - Inferring the schema by sampling is also experimental.

SQL core in depth

- Columnar in-memory table support
  - Caches table like RDD.cache, but as columnar style.
- Can prune unnecessary columns when read data.

Interesting issues

Interesting issues

- Support the GroupingSet/ROLLUP/CUBE
  - https://issues.apache.org/jira/browse/SPARK-2663
- Use statistics to skip partitions when reading from in-memory columnar data
  - https://issues.apache.org/jira/browse/SPARK-2961

Interesting issues

- Pluggable interface for shuffles
  - https://issues.apache.org/jira/browse/SPARK-2044
- Sort-merge join
  - https://issues.apache.org/jira/browse/SPARK-2213
- Cost-based join reordering
  - https://issues.apache.org/jira/browse/SPARK-2216

How to contribute

How to contribute

- See: Contributing to Spark
- Open an issue on JIRA
- Send pull-request at GitHub
- Communicate with committers and reviewers
- Congratulations!

Conclusion

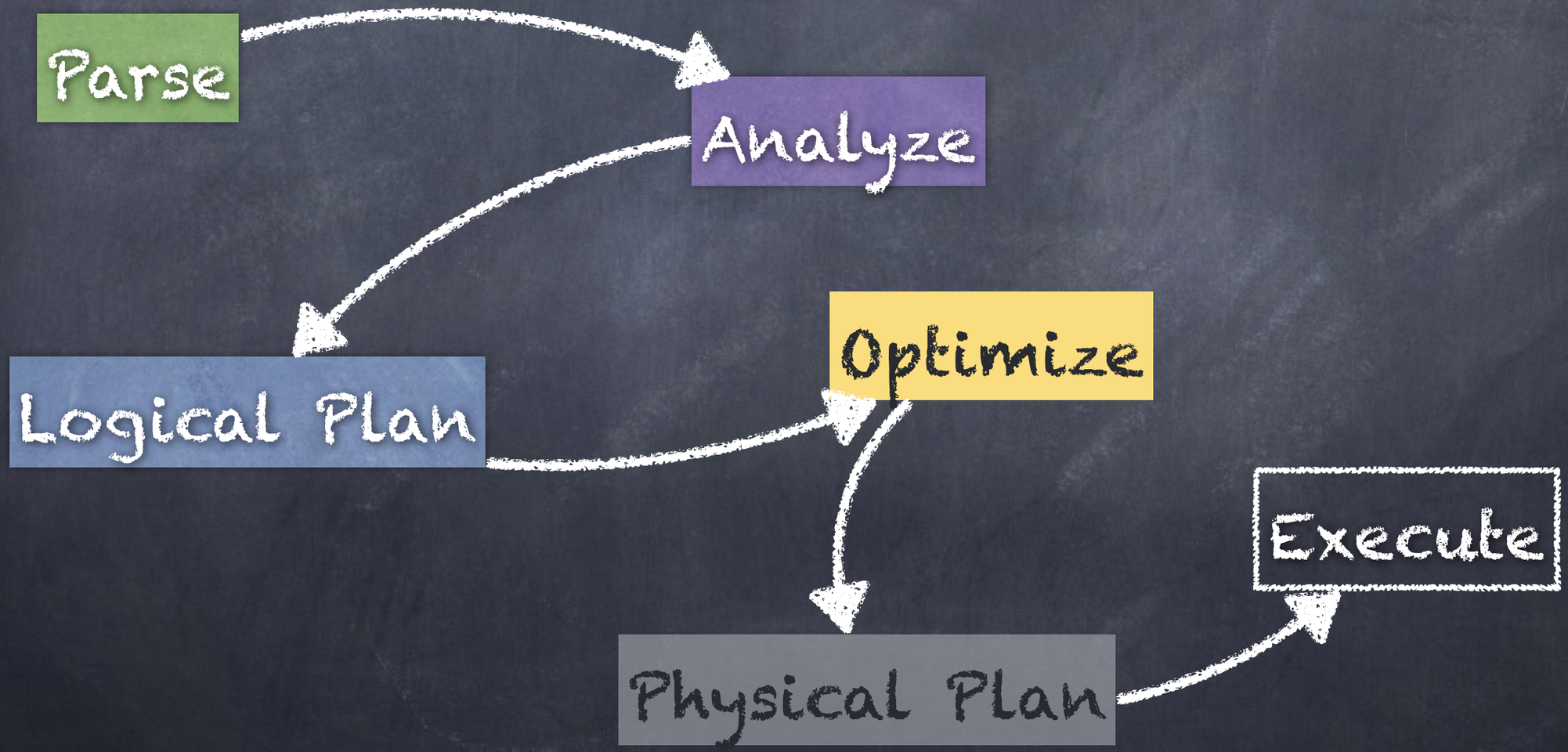
- Introduced Spark SQL & Catalyst
- Now you know them very well!
- And you know how to contribute.
- Let's contribute to Spark & Spark SQL!!

an addition

What are we doing?

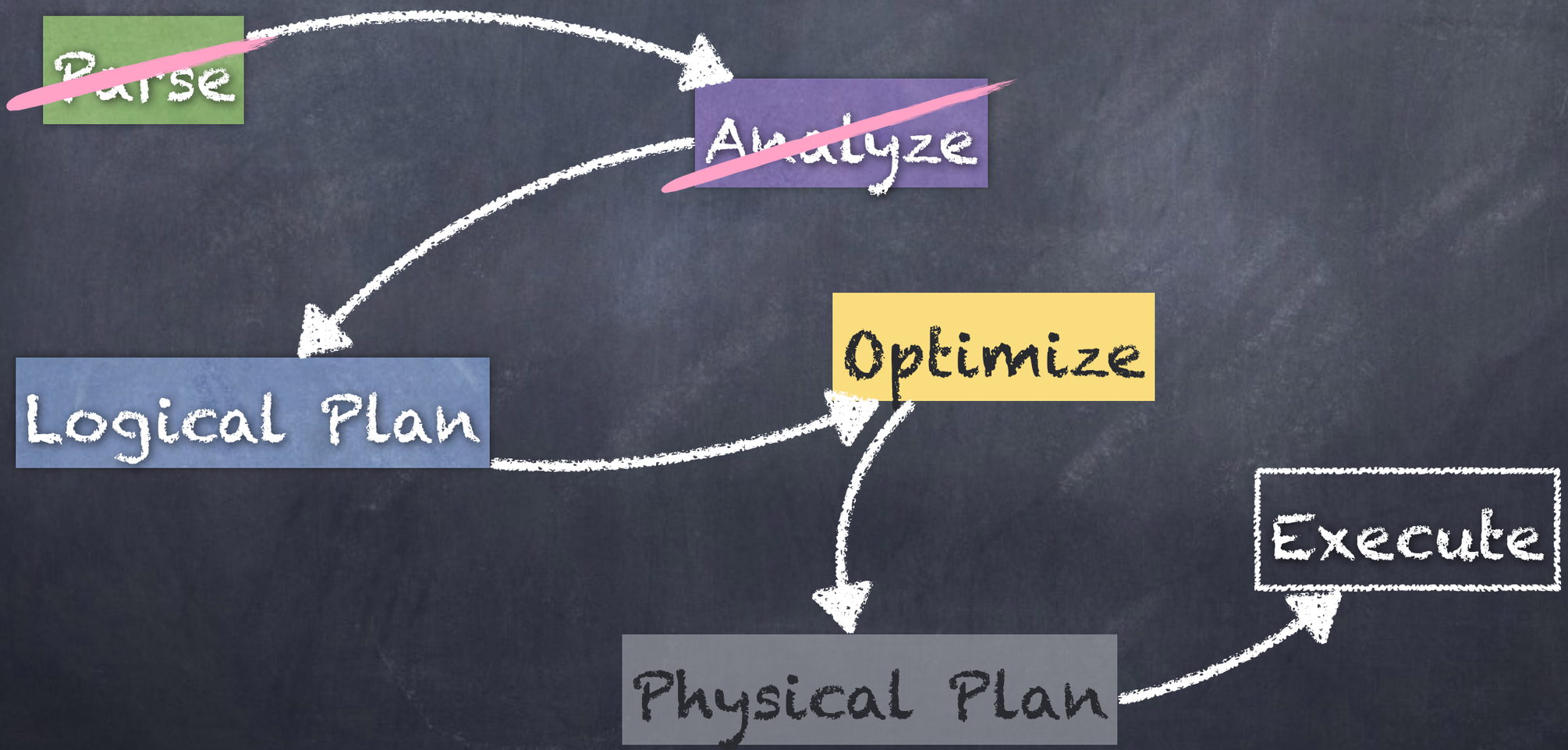
What are we doing?

- To execute query needs some steps.



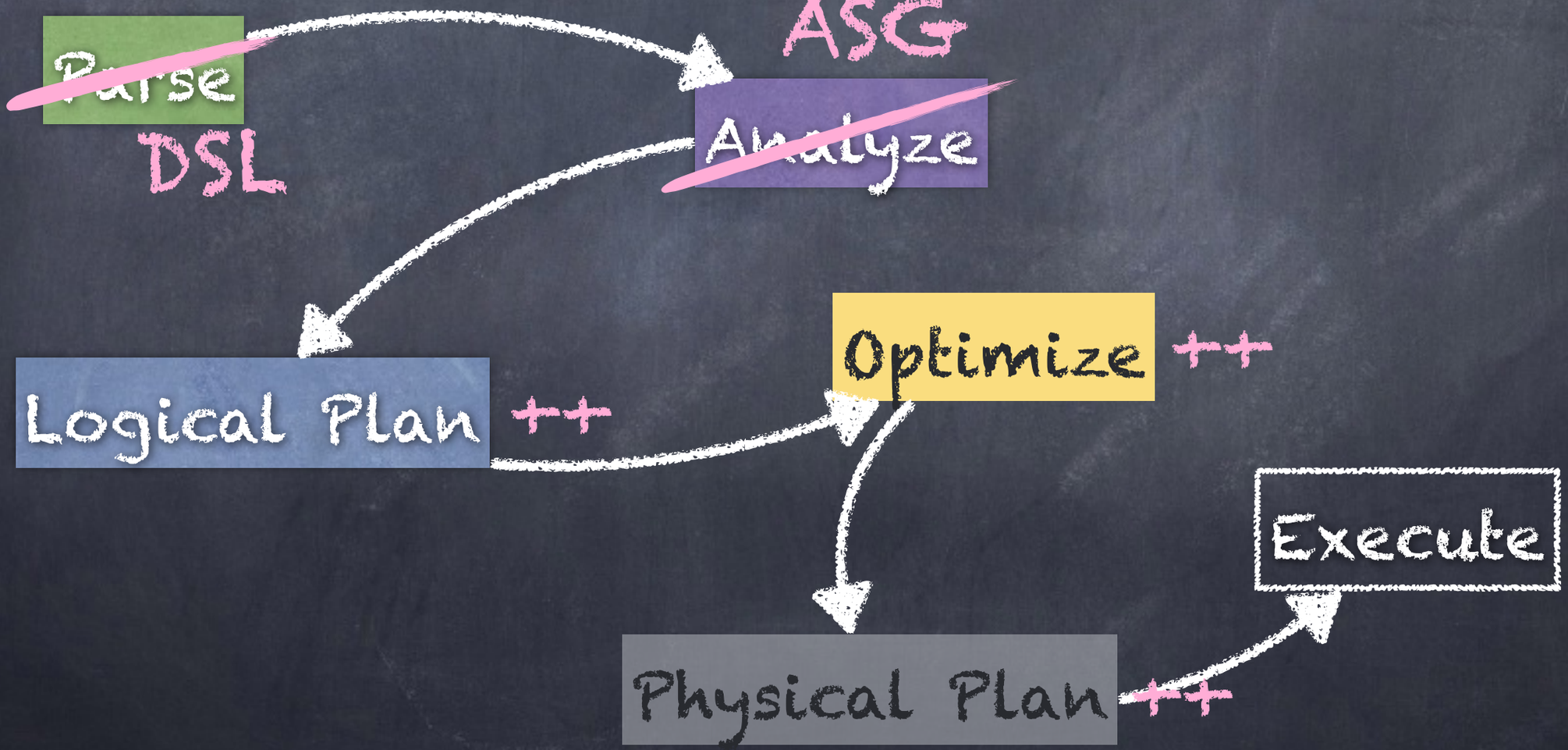
What are we doing?

- To execute ~~query~~ needs some steps.



What are we doing?

- To execute ~~query~~ business logic needs some steps.



Thanks!