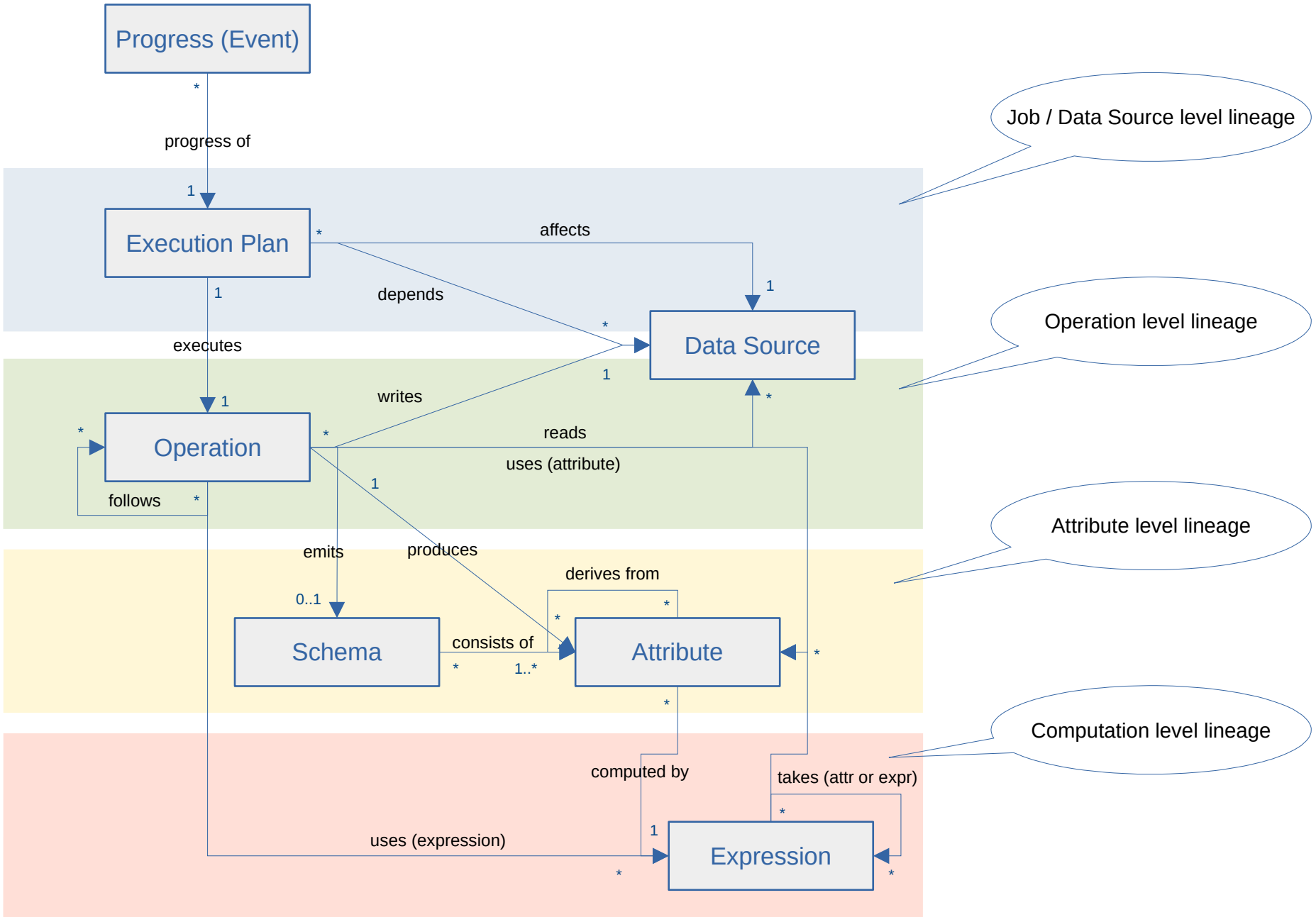

Progress (Event)
progress of
Execution Plan
affects
depends
Data Source
executes
writes
reads
uses (attribute)
Operation
follows
emits
produces
derives from
Schema
consists of
Attribute
computed by
takes (attr or expr)
uses (expression)
Expression
Job / Data Source level lineage
Operation level lineage
Attribute level lineage
Computation level lineage

Entity Definitions

Nodes

Progress
(top-level entity)

Describes the fact (and optionally the result) of execution of a given *Execution Plan.*
The *Progress* events are ordered according to their timestamp and represent (describe) concrete portions of data affected by the given run.
The *Progress entity introduces* a chronological axis to the data lineage. This is the only “dynamic” entity in the Spline data model. The rest entities are static in terms that they don't hold any meta-information that can be logically bound to any specific point in time.

Execution Plan
(top-level entity)

Represents a piece of data transformation pipeline implemented as a standalone application, process, script etc.
Execution Plan consists of exactly one *Write* operation and any number of *Read* and *Transform* operations. (See *Operation*)

Data Source
(top-level entity)

This entity represents a given uniquely named data location of any type where the data is read from of written to.
It could be a file, a table in a database, a Kafka topic, FTP location, REST endpoint etc.
The uniqueness is achieved by using URI as a key.

Operation

An *Operation is a b*uilding block of the data transformation pipelines. It represents a relational operation executed on a set of data rows

$$f(S_{1,}S_{2,}...)\rightarrow S'$$

where S_1, S_2 etc are input sets of data, and S' is the output.
Some operations change the data structure (e.g. SQL SELECT), the others only affect the number of rows (e.g. FILTER or SORT)
Certain operations are binary (JOIN or UNION), the others are unary. Some operations (terminal operations) contain an empty number of input data sets (e.g. *Generate*). The *Read* and *Write* operations are special in meaning that they represent I/O side effect and are always terminal.

Operations form a direct acyclic graph (DAG) with a single *Write* operation completing the chain.
The *Operation* is not a top-level entity and is bound to the *Execution Plan* via the composite-component relation.

Schema

In the Spline model by *Schema* we understand a structure of a data set that an *Operation* deals with. Every *Operation* emits a *Schema.*
The *Schema* describes the *Attributes* and their order. *Schemas* are logically related to the scope of a given *Execution Plan,* but could be shared by different O*perations* (those that don't change the data structure) that belongs to the same *Execution Plan.*

Attribute

Represents a single attribute of the data relation (e.g. a column in a table). It is characterized by a name and optionally a data type.
Attribute belongs to (one of more) *Schema(s)* in the scope of the same *Execution Plan*. Being a part of a *Schema* the *Attribute* is logically bound to the *Operation* that created that attribute (called the operation of origin). Every *Attribute* has exactly one operation of origin. Two attributes with the same name and type created by different operations are considered different. An *attribute* can derive from other attributes (e.g. SELECT a+b AS c), and optionally contain a reference to an Expression that describes how exactly it was calculated. (See *Expression*)

Expression

This is the finest level of abstraction in the Spline data lineage model. An *Expression* represents a mathematical expression that was used to calculate data for a given *Attribute*. *Expressions* form a DAG structure similar to *Operations* DAG, but operates on the attribute level, rather than a data relation level.
Expression is described by it's name (and optionally a data type), can have any arity, and refer other *Expressions* or *Attribute* as operands.

Edges

Progress Of
Connects *Progress to* the *Execution Plan*

Depends / Affects
Connects *Execution Plan to* the *Data Source*

Reads / Writes
Connects *Read* or *Write* operation *respectively* to the *Data Source*

Executes
Connects *Execution Plan to* the *Write* operation, that is the root of the given operation DAG.

Follows
Connects *Operations together in a DAG*

Emits
A relationship between an *Operation* and its output *Schema*

Consists Of
A composite relationship between *Schema* and *Attribute*

Uses
A connection between Operation and *Expressions* (or *Attributes*) that this operation refers as parameters (e.g. connection between FILTER operation and a predicate)

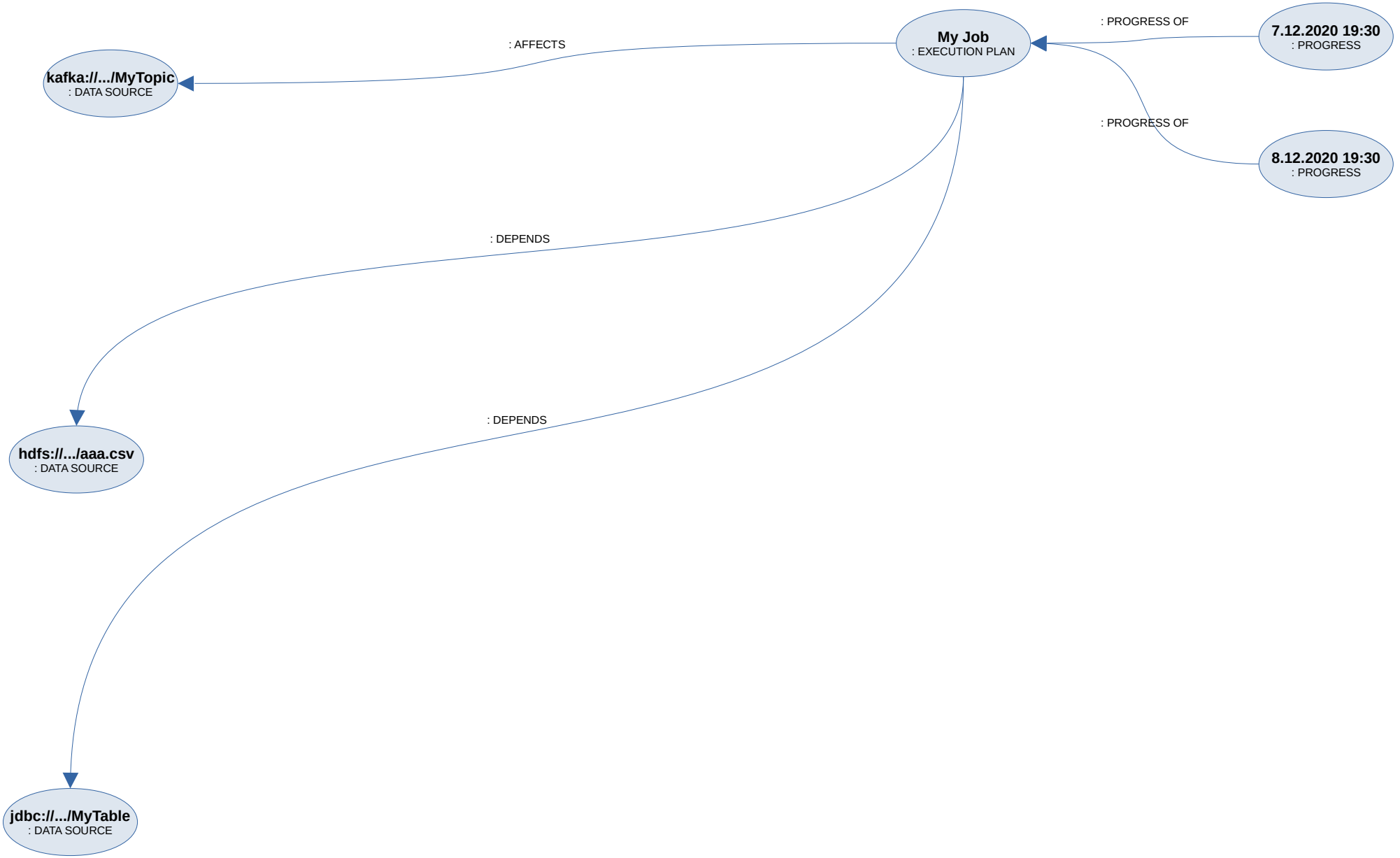
Produces
Connects an *Attribute* to its *Operation* of origin.

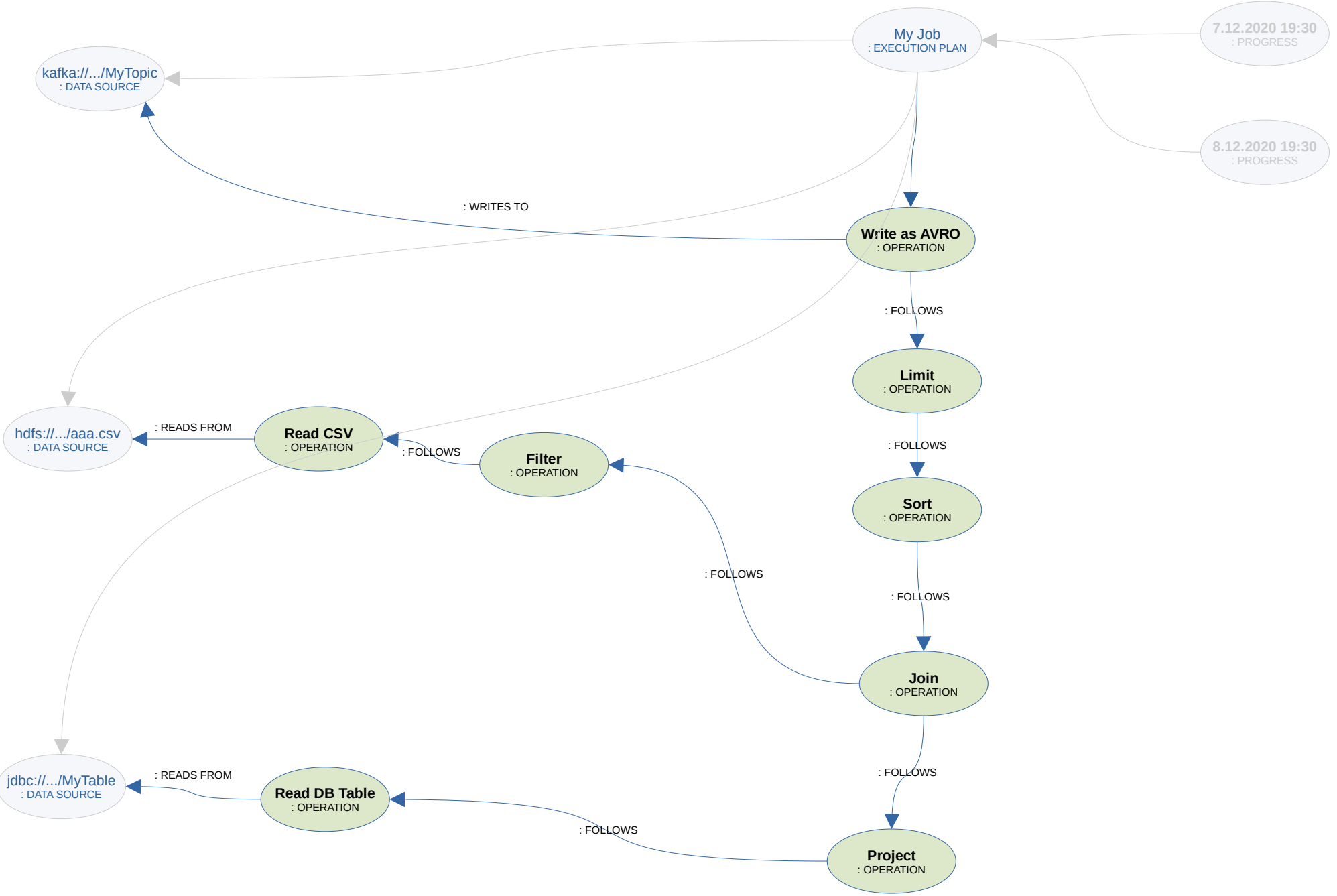
Takes
A connection between an *Expression* and its operands (other *Expressions* or *Attributes*)

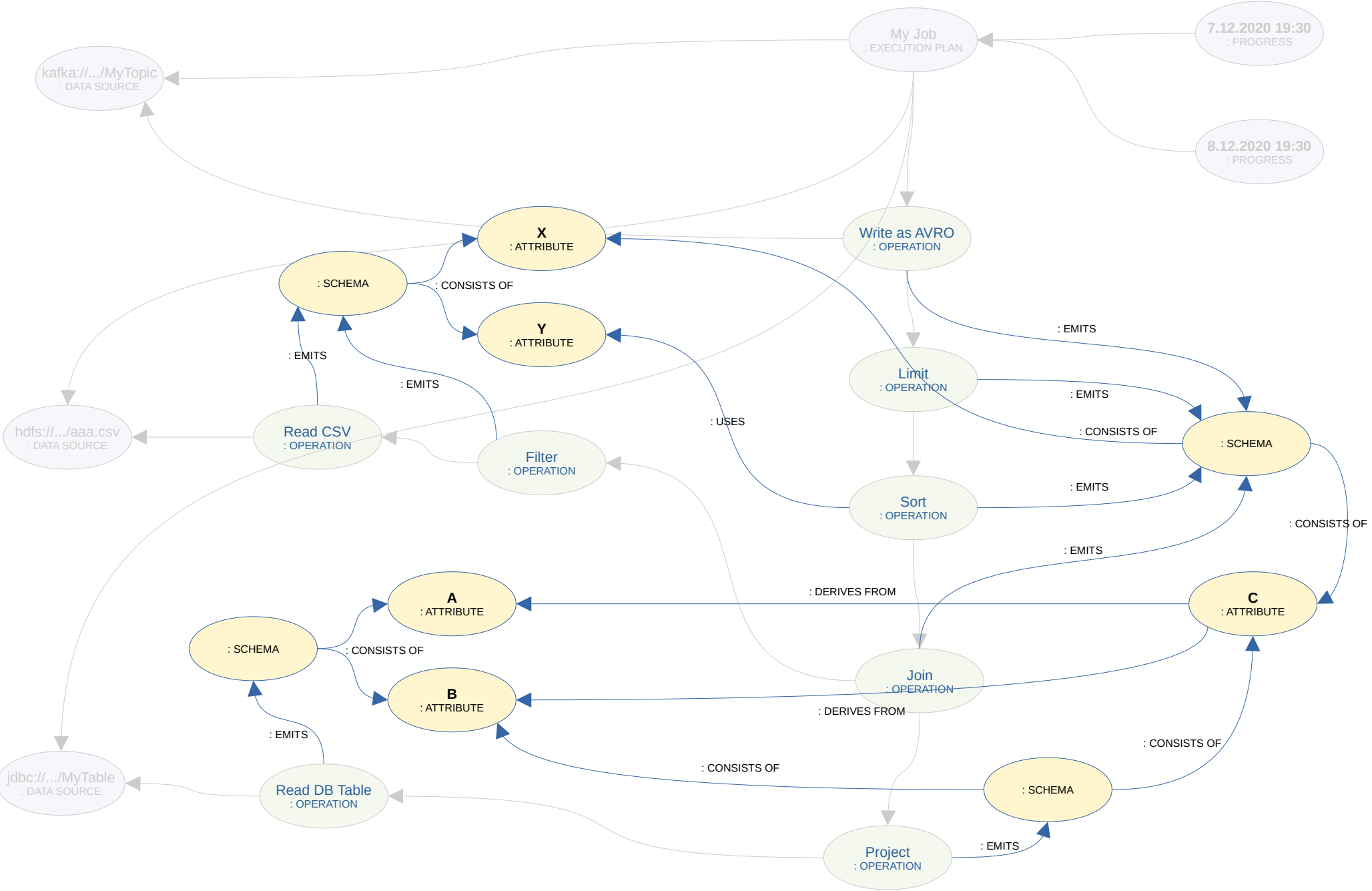
Derives From
Models a dependency between *Attributes*

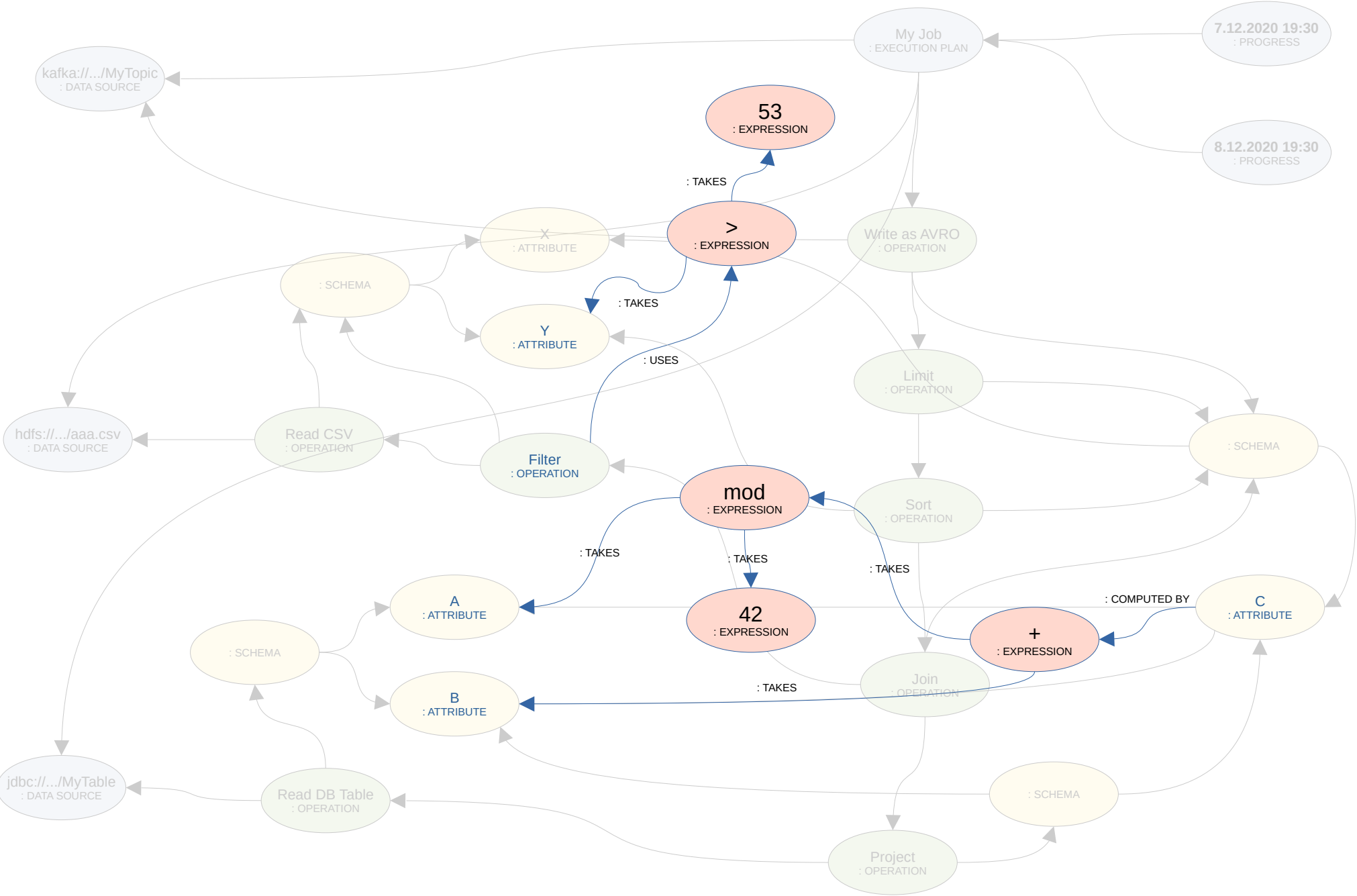
Computed By
Links an *Attribute* with an *Expression* that was used to compute values for the given *Attribute*

Example

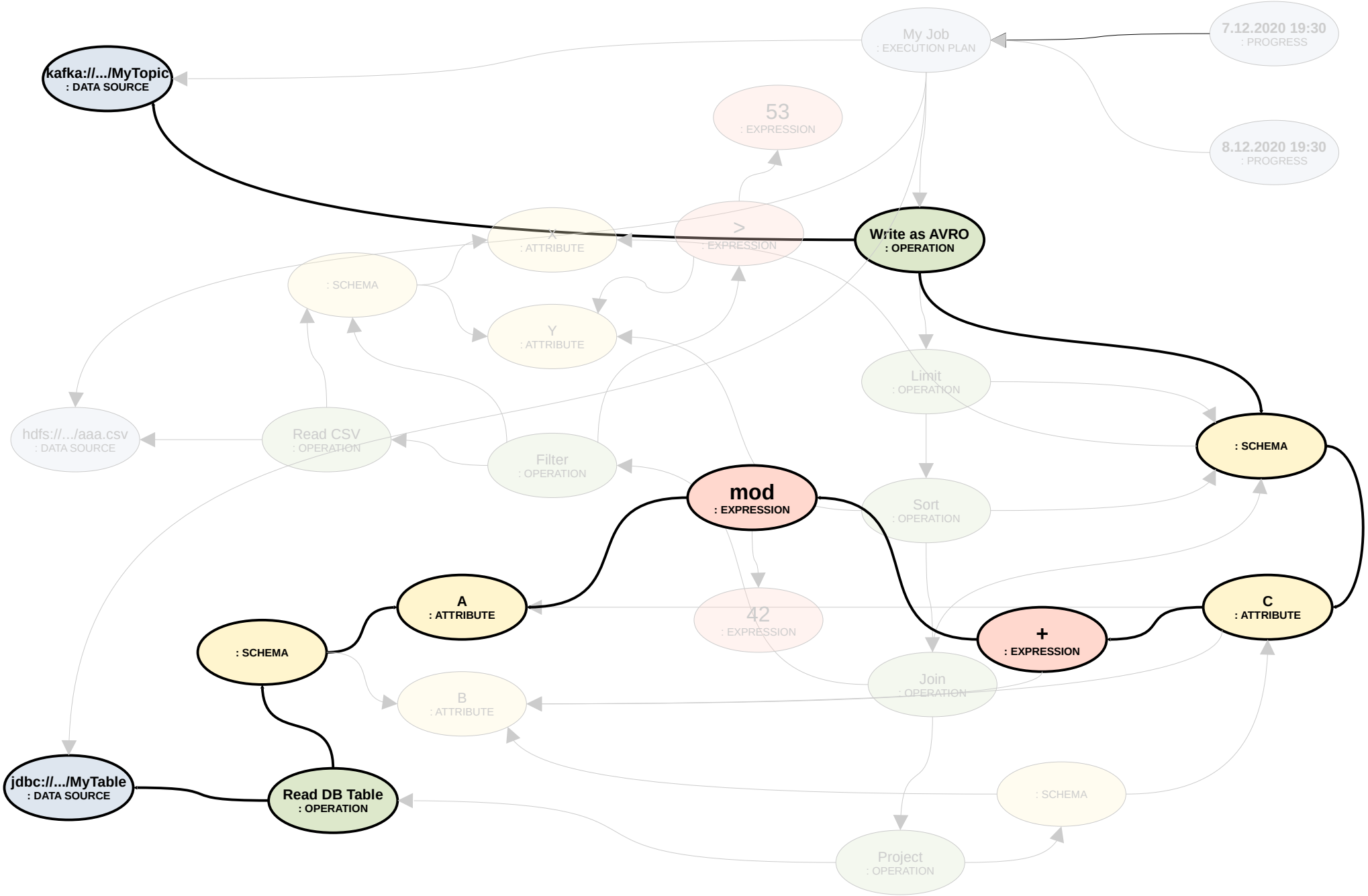

kafka://.../MyTopic
: DATA SOURCE
: AFFECTS
My Job
: EXECUTION PLAN
: PROGRESS OF
7.12.2020 19:30
: PROGRESS
: PROGRESS OF
8.12.2020 19:30
: PROGRESS
: DEPENDS
: DEPENDS
hdfs://.../aaa.csv
: DATA SOURCE
jdbc://.../MyTable
: DATA SOURCE


My Job
: EXECUTION PLAN
7.12.2020 19:30
: PROGRESS
8.12.2020 19:30
: PROGRESS
kafka://.../MyTopic
: DATA SOURCE
: WRITES TO
Write as AVRO
: OPERATION
: FOLLOWS
Limit
: OPERATION
: FOLLOWS
Sort
: OPERATION
: FOLLOWS
Join
: OPERATION
: FOLLOWS
Project
: OPERATION
: FOLLOWS
Filter
: OPERATION
: FOLLOWS
Read CSV
: OPERATION
: READS FROM
hdfs://.../aaa.csv
: DATA SOURCE
: FOLLOWS
Read DB Table
: OPERATION
: READS FROM
jdbc://.../MyTable
: DATA SOURCE


My Job
: EXECUTION PLAN
7.12.2020 19:30
: PROGRESS
8.12.2020 19:30
: PROGRESS
kafka://.../MyTopic
: DATA SOURCE
X
: ATTRIBUTE
Write as AVRO
: OPERATION
: SCHEMA
: CONSISTS OF
Y
: ATTRIBUTE
: EMITS
: EMITS
: EMITS
Limit
: OPERATION
: EMITS
: CONSISTS OF
: SCHEMA
hdfs://.../aaa.csv
: DATA SOURCE
Read CSV
: OPERATION
Filter
: OPERATION
: USES
Sort
: OPERATION
: EMITS
: CONSISTS OF
: EMITS
A
: ATTRIBUTE
: DERIVES FROM
C
: ATTRIBUTE
: SCHEMA
: CONSISTS OF
Join
: OPERATION
B
: ATTRIBUTE
: DERIVES FROM
: EMITS
: CONSISTS OF
: CONSISTS OF
jdbc://.../MyTable
: DATA SOURCE
Read DB Table
: OPERATION
: SCHEMA
Project
: OPERATION
: EMITS


My Job
: EXECUTION PLAN
7.12.2020 19:30
: PROGRESS
8.12.2020 19:30
: PROGRESS
kafka://.../MyTopic
: DATA SOURCE
53
: EXPRESSION
: TAKES
>
: EXPRESSION
X
: ATTRIBUTE
Write as AVRO
: OPERATION
: SCHEMA
: TAKES
Y
: ATTRIBUTE
: USES
Limit
: OPERATION
hdfs://.../aaa.csv
: DATA SOURCE
Read CSV
: OPERATION
Filter
: OPERATION
: SCHEMA
mod
: EXPRESSION
Sort
: OPERATION
: TAKES
: TAKES
: TAKES
A
: ATTRIBUTE
42
: EXPRESSION
: COMPUTED BY
C
: ATTRIBUTE
: SCHEMA
+
: EXPRESSION
Join
: OPERATION
: TAKES
B
: ATTRIBUTE
jdbc://.../MyTable
: DATA SOURCE
Read DB Table
: OPERATION
: SCHEMA
Project
: OPERATION

My Job
: EXECUTION PLAN
7.12.2020 19:30
: PROGRESS
8.12.2020 19:30
: PROGRESS
kafka://.../MyTopic
: DATA SOURCE
53
: EXPRESSION
X
: ATTRIBUTE
>
: EXPRESSION
Write as AVRO
: OPERATION
: SCHEMA
Y
: ATTRIBUTE
Limit
: OPERATION
hdfs://.../aaa.csv
: DATA SOURCE
Read CSV
: OPERATION
Filter
: OPERATION
: SCHEMA
mod
: EXPRESSION
Sort
: OPERATION
A
: ATTRIBUTE
42
: EXPRESSION
C
: ATTRIBUTE
+
: EXPRESSION
: SCHEMA
Join
: OPERATION
B
: ATTRIBUTE
jdbc://.../MyTable
: DATA SOURCE
Read DB Table
: OPERATION
: SCHEMA
Project
: OPERATION


My Job
: EXECUTION PLAN
7.12.2020 19:30
: PROGRESS
8.12.2020 19:30
: PROGRESS
kafka://.../MyTopic
: DATA SOURCE
53
: EXPRESSION
X
: ATTRIBUTE
>
: EXPRESSION
Write as AVRO
: OPERATION
: SCHEMA
Y
: ATTRIBUTE
Limit
: OPERATION
hdfs://.../aaa.csv
: DATA SOURCE
Read CSV
: OPERATION
Filter
: OPERATION
: SCHEMA
mod
: EXPRESSION
Sort
: OPERATION
A
: ATTRIBUTE
42
: EXPRESSION
C
: ATTRIBUTE
: SCHEMA
+
: EXPRESSION
B
: ATTRIBUTE
Join
: OPERATION
jdbc://.../MyTable
: DATA SOURCE
Read DB Table
: OPERATION
: SCHEMA
Project
: OPERATION