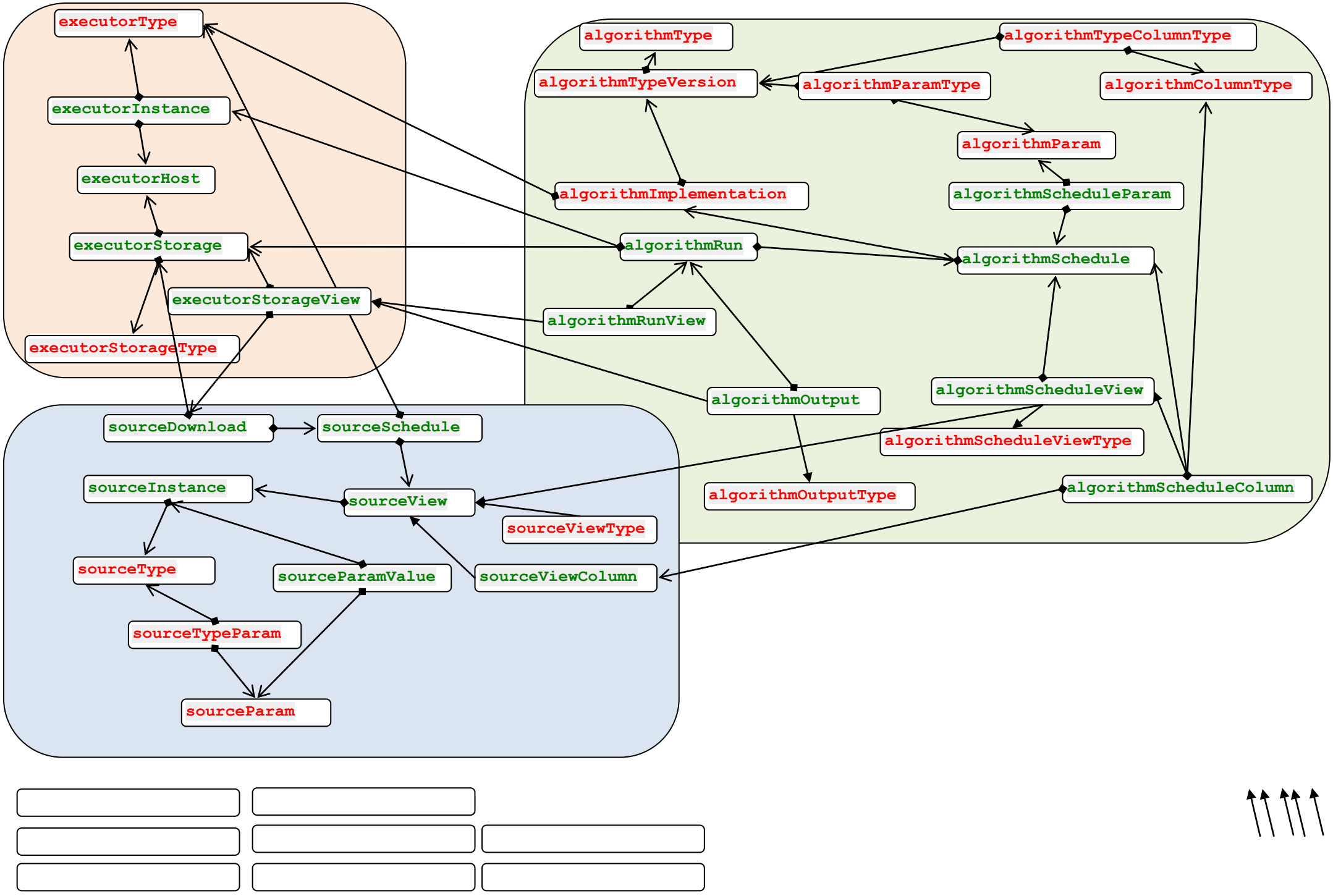
executorType
executorInstance
executorHost
executorStorage
executorStorageView
executorStorageType
algorithmType
algorithmTypeColumnType
algorithmTypeVersion
algorithmParamType
algorithmColumnType
algorithmParam
algorithmImplementation
algorithmScheduleParam
algorithmRun
algorithmSchedule
algorithmRunView
algorithmOutput
algorithmScheduleView
algorithmScheduleViewType
algorithmOutputType
algorithmScheduleColumn
sourceDownload
sourceSchedule
sourceInstance
sourceView
sourceViewType
sourceType
sourceParamValue
sourceViewColumn
sourceTypeParam
sourceParam

Starting from running **EXECUTOR** on given **HOST** - that action is adding rows into **executorInstance** and **executorHost** tables.

Each host can run several executors, depending on possibilities and needs. Each executor has type - one of types defined in **executorType** table.

Examples of **EXECUTORs** are: Local, R, Spark, SAS, SSAS. Each **EXECUTOR** has many **ALGORITHMs** to be run on that **EXECUTOR**.

Several **EXECUTORs** can run in one Java **JVM** sharing common **CONTEXT**. For HOST there could be many **STORAGEs** defined in table **executorStorage**.

Each **STORAGE** has type defined in **executorStorageType**. Examples of storage types: LOCAL, HDFS_CSV, HDFS_PARQUET, DB, SHARED.

Each **STORAGE** contains downloaded and materialized **VIEWS** from **SOURCE**, each **VIEW** is in table **executorStorageView**.

To define a data, first one should define **SOURCE** - that is in table **sourceIntance**.

Each **SOURCE** has type - one of types in table **sourceType**. Examples of source: JDBC, FTP, SHARED_FOLDER, WEB_SERVICE.

To define a source there is a need to define parameters like Connection String, User, Password, Ftp Host, Ftp Port - all parameters are in **sourceParam** table.

Each **SOURCE TYPE** might have different parameters requested to define connection to source, **JDBC** needs: Connection String, User, Password, Driver; **FTP** needs: Ftp Host, Ftp Port, User, Password. Mapping of necessary parameters are in table **sourceTypeParam** - this is **M2M** between **sourceType** and **sourceParam**.

When defining **SOURCE INSTANCE** - user is defining values for all requested parameters. **SOURCE INSTANCE** is in table **sourceInstance**.

All parameters for given **SOURCE INSTANCE** are in table **sourceParamValue**.

Each **SOURCE INSTANCE** can have many **VIEWS** - structures with defined schema available to download to **STORAGE**. For **JDBC** is it a list of tables/views. For **FTP** it is list of folders/files. Views are filled on request while connecting to **SOURCE INSTANCE** and are stored in table: **sourceView**. Views might be versioned.

Each **SOURCE VIEW** has many columns stored in table **sourceViewColumn**.

Each **SOURCE VIEW** has type - it is one of type defined in **sourceViewType** table: TABLE, VIEW, FILE, SQL, FOLDER, WEB_METHOD.

SOURCE VIEW could be scheduled to be downloaded, schedule can be periodic or on request (once). Definition of downloading **SOURCE VIEW** is in table **sourceSchedule**.

When downlading **SOURCE VIEW** into **executorStorageView** object, **sourceDownload** is created to check progress.

All ML algorithms are splitted into several **ALGORITHM TYPEs** defined in table **algorithmType**. Examples are: Prediction, Classification, Clustering, Market Basket. All ML algorithms for given type have the same type of inputs and outputs, but different implementations (R, SAS, Spark ML, ...).

All parameters for algorithms are in **algorithmParam** table. Sample parameters are: Alpha, Minimum Confidence, Time Series Periods.

All columns needed for algorithm are in **algorithmColumnType** table. Sample columns are: TimeMonths, Value, Group, Predicted, Input.

Mappings between **ALGORITHM VERSION** and **ALGORITHM PARAMETER** are in **algorithmParamType** table.

Mappings between **ALGORITHM VERSION** and **ALGORTHM COLUMN** are in **algorithmTypeColumnType** table.

Algorithm types are versioned creating **ALGORITHM VERSION**, it means we can run Prediction v1.0 or Prediction v1.2. All versions are in table: **algorithmTypeVersion**. The reason of versioning is to keep backward compatibility even with changing/extending algorithm types by new columns/parameters. So, it could be possibility of use new version (upgrade) after adding all new required columns and parameters.

All implementations for all algorithms are in table **algorithmImplementation**. Each implementation is written exactly for one **EXECUTOR TYPE**. For example **LocalPredictionSpark** implementation is for **SPARK** executor type, **PredictionTimeSeriesR** is for **R** executor type.

To run any implementation of algorithm, **ALGORITHM SCHEDULE** must be defined in table **algorithmSchedule**.

For given **ALGORITHM SCHEDULE** input **ALGORITHM PARAMETERs** should be set in table **algorithmScheduleParam**. That table contains value for all parameters to run algorithm.

For given **ALGORITHM SCHEDULE** input **VIEWS** should be selected in table **algorithmScheduleView**.

For given **algorithmScheduleView** user can define type - one of type in table **algorithmScheduleViewType**, sample types: TRAINING, TESTING, VALIDATION, INPUT.

Each **ALGORITHM SCHEDULE** could be took by **EXECUTOR** and run creating **ALGORITHM RUN** object in table **algorithmRun**.

For given **ALGORITHM RUN** all views are collected in table **algorithmRunView**.

Each **ALGORITHM RUN** is creating many outputs **ALGORITHM OUTPUT** in table **algorithmOutput**.

Each **ALGORITHM OUTPUT** is a file in **STORAGE**.