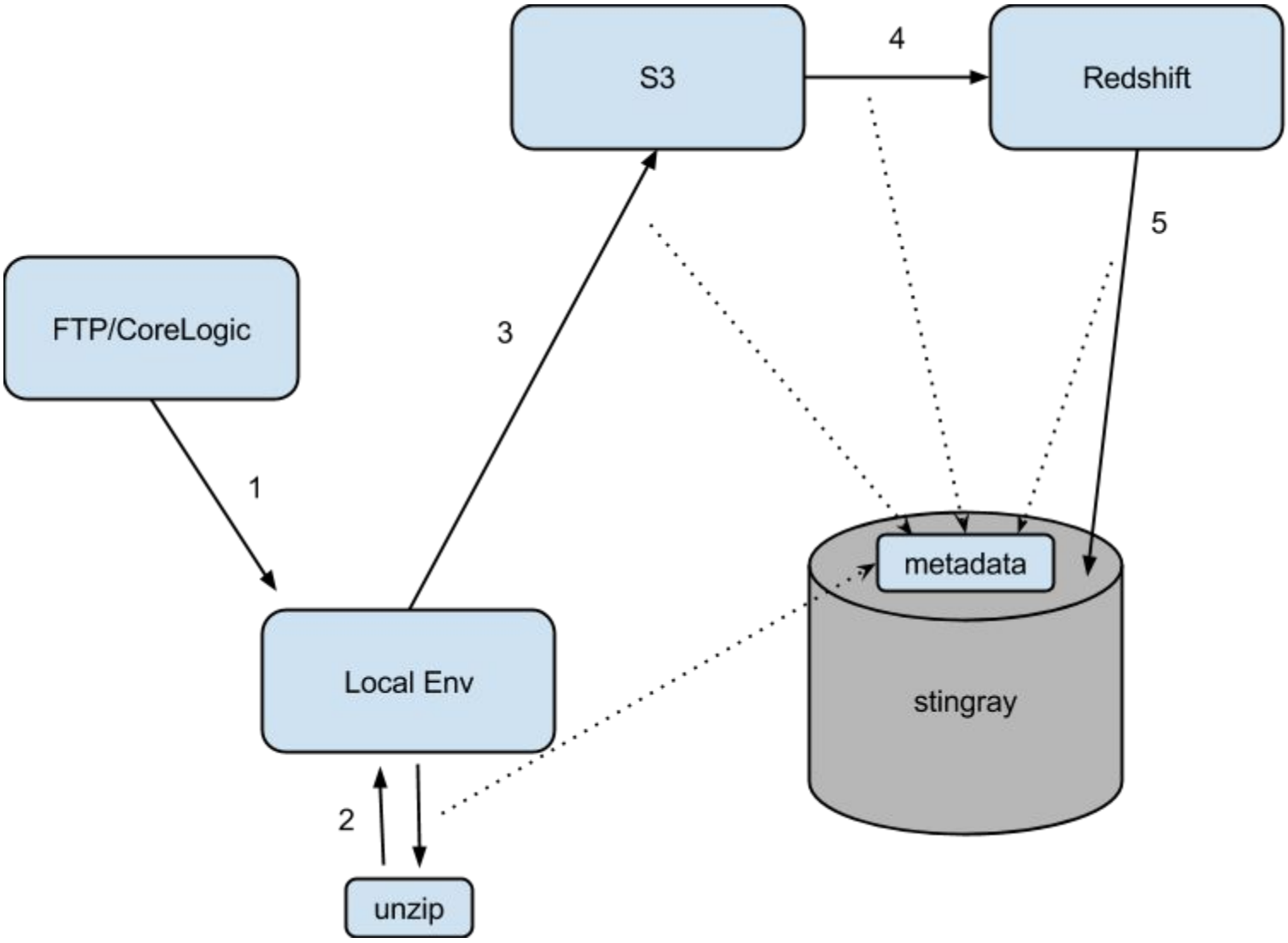
COLD Importer State Machine Schema Design

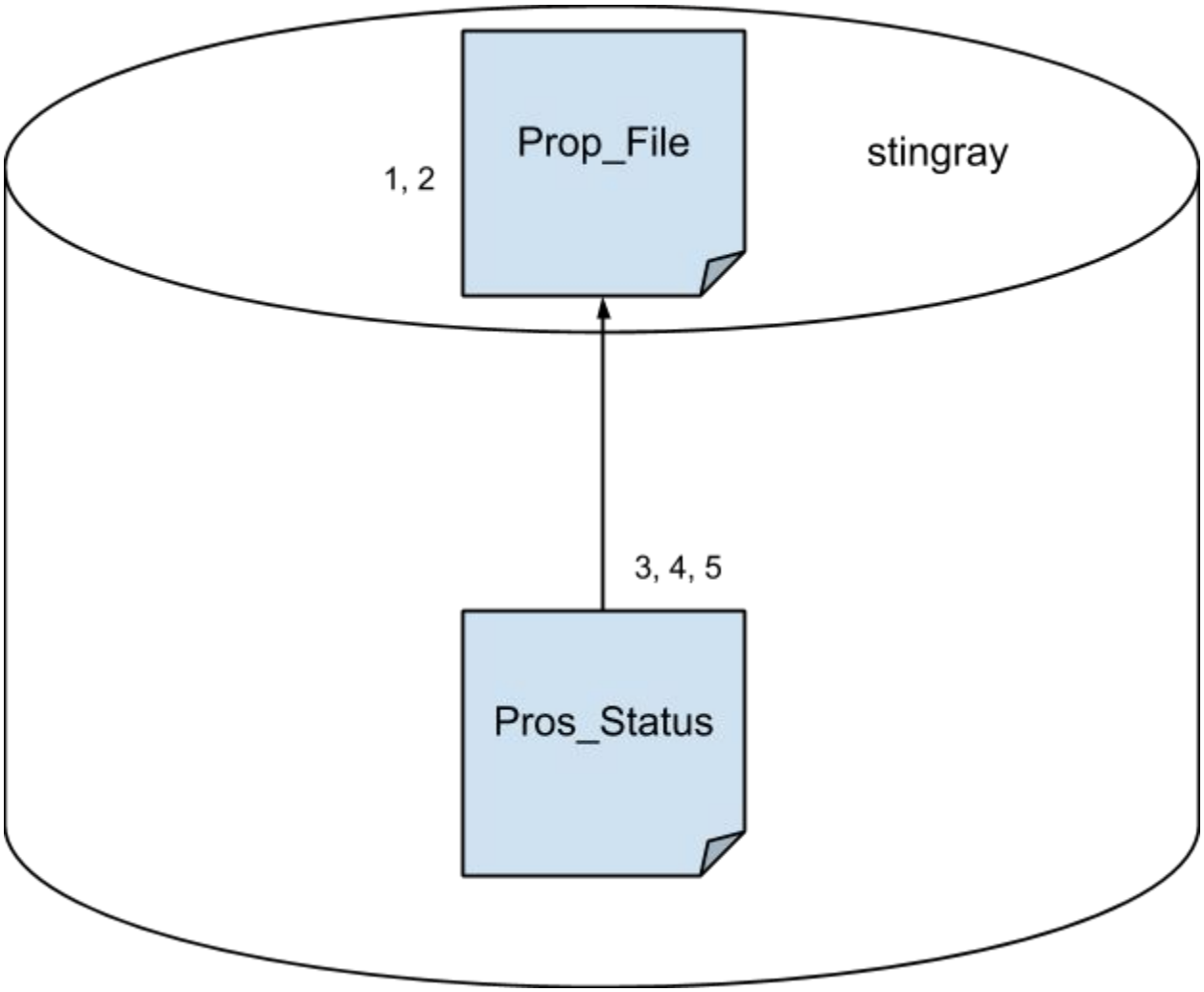


Overview

Even though we would like our importers to successfully complete every time, this model is not realistic in a world where servers fail and networking breaks on a daily basis. Because of this, we need to incorporate a ‘state machine’ in the schema. For every step of the process, we need to know if the action was completed, and if not, how far the program executed before halting. This doc defines the schema for the objects in charge of keeping track of the current progress in the workflow. The processing status for the ‘state machine’ will be stored in stingray.

Processing Status Overview

You can think of the table hierarchies as this:



All of the objects in the diagram pictured above represent tables of processing status in the *stingray* database. The *stingray* database will store the actual data in the pipeline as well as the state machine processing status for each step.

The main processing status table for each file of data will be stored in the *cold_importer_property_file_processsing_status* table. Each row in the *cold_importer_property_file_processsing_status* table represents an individual file of property data that is going through the COLD Importer pipeline. Each time a file finishes any of the sub-steps, the *step* and *last_updated* fields should be updated in the

cold_importer_property_file_processsing_status table. Also, *workflow_complete* should return true once the data file has been stored successfully in stingray. The numbers by each object refer to the steps in the workflow defined at the top of the document.

The *cold_importer_property_file_processsing_status* refers to the rest through a foreign key constraint. Again, all of the objects above represent tables of metadata storing the processing status in the stingray database. The purpose is to keep track of where we currently are in the pipeline. The *stingray* table of stores the processing status on the step of importing data into *stingray* from Redshift.

Relationships between Tables

Schemas

cold_importer_processing_status - APPEND ONLY

Column	Type	Description
id	ID/bigint	*id of this particular object*
name	String	*name of this particular object*
step_complete	Boolean	*Is the current step completed or in progress*
created_date	Time	*timestamp when this object was created*
processing_status_type	enum	*Either S3_ProcessingStatus, Redshift_Processing_Status, or Stingray_Processing_Status*
cold_importer_property_file_processing_status_id	ID	*Foreign key reference to file being uploaded from server*

Class/Object Design

- ColdImporterS3ProcessingStatus, ColdImporterRedshiftProcessingStatus, and ColdImporterStingrayProcessingStatus all inherit from this class (POJO's)
- ColdImporterPropertyFileProcessingStatus does **NOT** inherit from this

ColdImporterProcessingStatus {

// variables
id: Long/ID primary key
name: String
stepComplete: Boolean
createdDate: Calendar
coldImporterPropertyFileID: Long/ID foreign key

// function getters
getID() ----> return the ID of this object
getName() ----> returns the name of this object
getStepComplete() ---> return if this step is **fully** complete in pipeline
getLastUpdated() ----> return calendar timestamp of created data to this property

getType() ----> return the type of this processing status object
getColdImporterPropertyFileProcessingStatusID()

- return the ColdImporterPropertyFileProcessingStatus object by it's ID

// function setters
setID() ----> sets the ID of this object
setName() ----> sets the name of this object
setStepComplete() ---> sets this step is **fully** complete in pipeline
settLastUpdated() ----> sets calendar timestamp of created data to this property
setType() ----> sets type of the processing status object
setColdImporterPropertyFileProcessingStatusID(Long newID)

- set the ColdImporterPropertyFileProcessingStatus object by it's ID

}

ColdImporterS3ProcessingStatus extends ColdImporterProcessingStatus {

// variables
id: Long/ID primary key
name: String
stepComplete: Boolean
createdDate: Calendar
coldImporterPropertyFileProcessingStatusID: Long/ID foreign key

// function getters
getID() ----> return the ID of this object
getName() ----> returns the name of this object
getStepComplete() ---> return if this step is **fully** complete in pipeline
getLastUpdated() ----> return calendar timestamp of created data to this property
getColdImporterPropertyFileProcessingStatusID()

- return the ColdImporterPropertyFileProcessingStatus object by it's ID

// function setters
setID(Long Long ColdImporterS3ProcessingStatusID) ----> sets the ID of this object
setName(String name) ----> sets the name of this object
setStepComplete(Boolean step) ---> sets this step is **fully** complete in pipeline
setLastUpdated(Calendar Time)

- sets calendar timestamp of created data to this property

setColdImporterPropertyFileProcessingStatusID(Long ID)

- set the ColdImporterPropertyFileProcessingStatus object by it's ID

}

ColdImporterRedshiftProcessingStatus extends ColdImporterProcessingStatus {

// variables
id: Long/ID primary key
name: String
stepComplete: Boolean
createdDate: Calendar
coldImporterPropertyFileProcessingStatusID: Long/ID foreign key

// function getters
getID() ----> return the ID of this object
getName() ----> returns the name of this object
getStepComplete() ---> return if this step is **fully** complete in pipeline
getLastUpdated() ----> return calendar timestamp of created data to this property
getColdImporterPropertyFileProcessingStatusID()
- return the ColdImporterPropertyFileProcessingStatus object by it's ID

// function setters
setID(Long ColdImporterRedshiftProcessingStatusID) ----> sets the ID of this object
setName(String name) ----> sets the name of this object
setStepComplete(Boolean step) ---> sets this step is **fully** complete in pipeline
setLastUpdated(Calendar time)
- sets calendar timestamp of created date to this property

setColdImporterPropertyFileProcessingStatusID(Long ID)
- set the ColdImporterPropertyFileProcessingStatus object by it's ID

}

ColdImporterStingrayProcessingStatus extends ColdImporterProcessingStatus {

// variables
id: Long/ID primary key
name: String
stepComplete: Boolean
createdDate: Calendar
coldImporterPropertyFileProcessingStatusID: Long/ID foreign key

// function getters
getID() ----> return the ID of this object
getName() ----> returns the name of this object
getStepComplete() ---> return if this step is **fully** complete in pipeline
getLastUpdated() ----> return calendar timestamp of created data to this property

getColdImporterPropertyFileProcessingStatusID()

- return the ColdImporterPropertyFileProcessingStatus object by it's ID

// function setters

setID(Long ColdImporterRedshiftProcessingStatusID) ----> sets the ID of this object
setName(String name) ----> sets the name of this object
setStepComplete(Boolean step) ---> sets this step is **fully** complete in pipeline
setLastUpdated(Calendar time)

- sets calendar timestamp of created date to this property

setColdImporterPropertyFileProcessingStatusID(Long ID)

- set the ColdImporterPropertyFileProcessingStatus object by it's ID

}

ColdImporterProcessingStatusDAO {

findByID(Long ColdImporterProcessingStatusID)

- returns all of the ID's that match the param

findByName(String ColdImporterProcessingStatusName)

- returns all of the names that match the param

findByStepComplete(Boolean stepComplete)

- returns all of the ColdImporterProcessingStatus objects that match the boolean param

findByColdImporterPropertyFileProcessingStatusID(Long ID)

- returns a list of all the ColdImporterProcessingStatus objects whose ColdImporterPropertyFileProcessingStatusID matches the param

findByColdImporterPropertyFileProcessingStatusType(Type type)

- returns a list of all the ColdImporterProcessingStatus objects whose ColdImporterPropertyFileProcessingStatusType matches the param

}

cold_importer_property_file_processing_status - CRUD functionality

Column	Type	Description
id	ID/bigint	*id of this particular ProcessingStatus object*
name	String	*name of particular property_file*
current_workflow_step	enum	*should be updated at completion of each step for property*
workflow_complete	Boolean	*Is the complete workflow done for this object*
log_msg	[String]	*Way to output more information about download*
last_updated	Time	*timestamp when the last time this object was updated*
created_date	Time	*When this file object was initially created, don't update*

Class/Object Design

- The *cold_importer_property_file_processing_status* database will be supportable for all CRUD operations
- It should be a one to many relationship with each subtable:
    - every row in *cold_importer_property_file_processing_status* should have a unique id for the property
    - But each subtable can have many rows pointing to a particular property file
- It is not atomic, a file can be partially downloaded or decompressed with error and the action will still persist
- created_date is the only column that is immutable

ColdImporterPropertyFileProcessingStatus {

// variables
id: ID/bigint primary key
name: String
currentStep: enum
workflowComplete: Boolean
logMesages: [String]
lastUpdated: Calendar
createdDate: Calendar

// function getters

getID() ---> return the id of the property_file

getName() -----> return the name of the property_file

getCurrentStep() ------> return the step the file has currently completed

getWorkflowComplete() -----> return worfklowComplete boolean

getLogMessages() ----> return array of log messages associated with this property file

getLastUpdated() -----> return calendar timestamp of last update to this property

getCreatedDate() -----> return calendar timestamp of created data to this property

// function setters

setID(Long newID) ---> set the id of the property_file

setName(String name) -----> set the name of the property_file

setCurrentStep(enum step) ------> set the step the file has currently completed

setWorkflowComplete(Boolean complete) -----> set workflowComplete boolean

setLogMessages([String] messages)

- set array of log messages associated with this property file

setLastUpdated(Calendar time)

- set calendar timestamp of last update to this property

setCreatedDate(Calendar time)

- set calendar timestamp of created data to this property

}

ColdImporterPropertyFileProcessingStatusDAO {

findByID(Long ColdImporterPropertyFileProcessingStatusID)

- returns all ColdImporterPropetyFileProcessingStatus objects that match the parameter - should be 1

findByName(String ColdImporterPropertyFileProcessingStatusName)

- returns all ColdImporterPropetyFileProcessingStatus objects that match the parameter - should be 1

findByWorkflowComplete(Boolean stepComplete)

- returns all of the ColdImporterPropetyFileProcessingStatus objects that match the boolean parameter

findByCurrentStep(enum currentStep)

- returns all of the ColdImporterPropetyFileProcessingStatus objects that match the enum currentStep parameter

findByLastUpdated(Calendar time)

- returns all of the ColdImporterPropetyFileProcessingStatus objects that were updated after the time parameter

findByLogMessages(String logMessage)

- returns all of the ColdImporterPropetyFileProcessingStatus objects that have a logMessage that Regex matches the string parameter

findByCreatedDate(Calendar time)

- returns all of the ColdImporterPropetyFileProcessingStatus objects that were created on the day of the time parameter

}

Testing

ColdImporterPropertyFileProcessingStatusTest {

// makes sure that this POJO correctly holds all of the data for the lifetime of this object

// Test:

- correctly created upon a new download
- fields are inserted properly from the total POJO and property file POJO
- this object persists throughout the lifetime of the workflow of the POJO object it is referring to (file downloaded)
- updated correctly at each step

}

ColdImporterPropertyFileProcessingStatusDAOTest {

// makes sure that the DAO correctly updates and gets the object from *stingray*

// Test:

- every field is retrieved and stored properly

}

ColdImporterS3ProcessingStatusTest {

// makes sure the status is updated when the file is uploaded into S3

// Test:

- once the file is in S3, this metadata object becomes updated
- updates the ColdImporterPropertyFileProcessingStatus object as well when the task is complete

}

ColdImporterRedshiftProcessingStatusTest {

// makes sure the status is updated when the file is copied from S3 to Redshift

// Test:

- once the file is in Redshift, this metadata object becomes updated
- updates the ColdImporterPropertyFileProcessingStatus object as well when the task is complete

}

ColdImporterStingrayProcessingStatusTest {

// makes sure the status is updated when the file is imported from Redshift into *stingray*

// Test:

- once the file is in the right table in *stingray* this metadata object becomes updated
- updates the ColdImporterPropertyFileProcessingStatus object as well when the task is complete

}

ColdImporterProcessingStatusDAOTest {

// makes sure that the DAO correctly updates and gets the object from *stingray*

// Test:

- every field is retrieved and stored properly

}