CampaignFinanceDataPipeline

Sponsors: borden.jeremy@gmail.com
Author: irkydink@gmail.com

This doc will become or augment the pipeline folder readme.

Project goal: Establish an ETL pipeline for collecting and loading content to the Code for America Raleigh Brigade open NC campaign finance application.
There is a schema at
https://files.slack.com/files-pri/T02ETHD9P-F01F8RZNSVD/c4a_schemaupdated.png
to support the ETL (Extract, Transform, Load) strategy detailed below.

Project near term changes:
Presently the application is basic content search and is populated almost solely from a report export from the NC State Board of Elections (BOE) datasets.

The pipeline project near term objectives are the following:

- Instantiate a linux server presence to act as a content landing zone, content collection and curation facility to the application.
- Add development and application user access to the server.
- Bash environment automation to control repetitive and scheduled activities.
- Python environment with an import standard to configure #import beyond basic library services.
- Postgress instance to support application dev, test and intermediate ETL tasks.
- Nodejs instance to support development and testing of content status dashboard elements.

ETL Specific:
- Recognize the present application circumstance is a work in progress with .js and sql internal functions addressing application content issues. The near term goal can be summarized by understanding the pipeline should rationalize and standardize content in such a manner there is no special functionality internal to the application that addresses content rationalization issues. To that end, the following starter specification is offered.

For project laymen, understand that content while presently limited to being exported from BOE datasets represents campaign present state of “reported” activity. The input is form, human data entry and file (.pdf, .xlsx, or .tiff) formats. I can’t even begin to cover the ways this datastream is unreliable so just accept that it is.
For the application to provide researcher value, it will need content integration from a wider variety of digital interactions. Further, the principal sponsor will likely tune his notion of what the

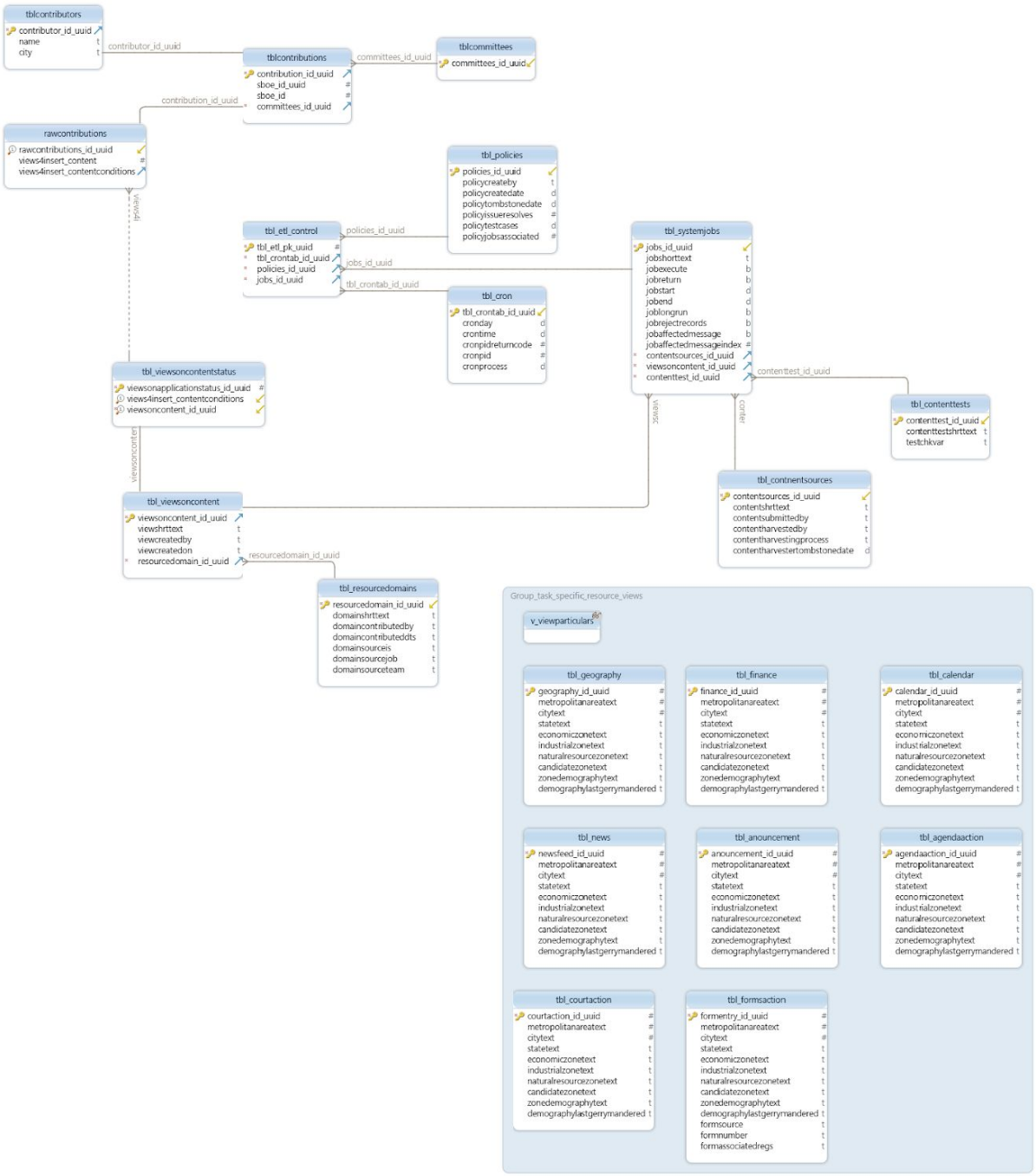
application should deliver or focus on and I expect that will come in the form of “something not done by other sites/resources.” To that end, the ETL pipeline will try to address present reported content shortcomings as well be positioned to curate, collect and interleave presently unidentified sources of information on a recurring schedule. Further, the pipeline project will create a public facing set of content state/reliability assets to guide the judgement of application readers.

The server will be configured to support the following content collection interactions:

- Ingest/collection
    - File sftp
    - API (third party site connections)
    - Crowdsource activity (twitter, msg, public webform, blog (etc))
    - Webscrape (different than API or search return) we are navigating public/private web presence and copy/downloading/building-ref-2 content
    - websearch-return (different than file/api/webscrape) these tasks require some automated interaction and form submission to generate downloadable or screen readable content.
- System/job control
    - Cron schedule interaction
    - Calendar functions to alter cron task for special conditions
- Source curation
    - File sftp
        - System bool operations (folder changed, transfer (here, expected, complete, fail, retransmit, inprocess), calendar operations, rowCount, infileRecordPosition)
        - Content Structure operations (file new, isaFragment, isaRetransmit, isDelimited,iseol-eofMarked)
        - Import Rationalization
            - Rows, columns, recordRegex, nlStructure
            - Headers? headersAlign2regex?, headersAlign2nl?, headersAlign2Labels?
            - recordAugmentation?, tagInHistory?, tagCrowdsourced?, tagSupportAction
    - API (third party site connections)
        - System bool operations (accountUsed,port, URLchanged, chgExpected, getComplete, getFail, getRetransmit, getActive), calendar operations, location,login, nav,throttle, threads)
    - Crowdsource
        - Apps, fingerprints,
    - Webscrape
        - System bool operations (accountUsed,port, URLchanged, chgExpected, getComplete, getFail, getRetransmit, getActive), calendar operations, location,login, nav,throttle, threads)
    - Websearch

- System bool operations (accountUsed,port, URLchanged, chgExpected, getComplete, getFail, getRetransmit, getActive), calendar operations, location,login, nav,throttle, threads) Social networks, emerging contexts, calendars, people, topical site activity

About the Schema



Notes On Assets
From the existing application schema, 4 tables (renamed for ETL)
tblContributors
tblContributions
tblCommittees
tblRawContributions (fk modify)

New Tables to support ETL pipeline:
tbl_ETL_control
The etl control table as as the interface to system located etl operations. Through this interface application owners and users alike can see application content status, open issues with source content and such.

Tbl_policies
This table holds reference to a view of processing policies applied during operating time frames. Policies are the rules being applied during ETL to raw content inputs that rationalize query supporting content within the application. Rules include regex, grouping, tagging and metadata operations that enhance application content.

tbl_systemJobs
The list of bash, python and api calls triggered by daily crontab activity.

Tbl_cron
This table reflects system crontab operations. Updated from command processes, this table provides system visibility for application users.

tbl_contentTests
This table is a reference to all the different tests against content traversing the etl pipeline. System jobs are (in whole or in part) packages of these jobs, some of which are required depending on the pipeline use case (ie. sftp or api). Tests may or may not be unique to individual content sources depending on how the data comes in and requires curation.

tbl_contentSources
Content sources are records of all the places that traverse the pipeline. Notice that the desitnation is in itself a sometimes content source. System activity, jobs and statuses may be gotten from application query making the destination a source depending on the curation process or test.

v_viewParticulars
viewParticulars are function/service level artifacts that support the creation of "definitially-complete" information assets as pg views. Individually, these are the "test-completed" assets by which content is reliably tagged for inclusion into application objects and classes.

v_viewParticulars are created from domain specific resources listed below.
Tbl_geography
Tbl_finance
Tbl_calendar
tbl_news
tbl_anouncement
Tbl_agendaAction

Tbl_courtAction
Tbl_formsAction
++ additional sources to be created as needed.

tbl_resourceDomains
A table of "object-level" content resources. Some of which are domain specific (like an address and associated geospatial content, or natural language artifacts and other content pulled from 3rdparty pipeline activity. These artifacts are the product of normalization and rationalization processes combined from contentTests and viewParticulars.

This table serves two functions. First, it provides the numerical linkage for presentation layer notifications on "by how much" a particular artifact may have been "influenced" by a pipeline issue or failure. Second, it provides to the application developer fully rationalized data structures or content artifacts for inclusion in the application site.

(example). select * from view contributorsFirstTime where firstname is 'charlie' or nickname "chuckles the donor'; If in the raw data "chuckles" is a nickname for Charles Moneybags, 200 CruellaDeville Court Raleigh, NC 27612. we can build the reference so contributions return.

tbl_viewsOnResources
View on resources are application-level collections of integrated information resources compiled from raw pipeline actions. Views created here are the resource from which the application interface draws on directly.

tbl_viewsOnContent
This table is a collection of views curated for specific page tasks.