Overview

When creating the database for Mapping Violence, several conversations were had on the benefits and drawbacks of various database types, such as relational and document-based. Given the desire for flexibility, time for development, and event-based specificity, the team decided to use a document-based, noSQL database MongoDB.

With this structure several collections were created. The collections centered on the core idea of tracking points of interest (POIs) - incidents of violence tied to a specific location. This served as the primary collection, storing information and metadata about the events as singular objects. For example, the POI collection includes these fields (and many more):

- Location
- Date
- Identity(ies) of Victim(s)
- Researcher Notes
- Sources

Each event is its own record in the database.

The team conceptualized data entry as a wiki wherein users could collaboratively create and edit these POIs. This led to the development of a version control system between each user's edits. A copy of each previous version is stored in the database.

POI Pages

POI Pages are container objects that represent a wiki entry. It keeps track of the current version of the POI and all previous versions.

Below is the schema for POI Pages:

- ID (MongoDB generated)
- className (For the ORM)
- current (A DBRef to a POI object)
- previous (An array of DBRefs to POI objects)
- creator (A user object)
- dateCreated ([Date POI Page was created | Date last POI version was created])
- status (A value of an enumeration denoting whether the POI Page is a draft ["DRAFT"], internal ["IN_POOL"], or public ["PUBLIC"]

POI

This is where the primary data is stored. Jim's spreadsheet outlines the top-level fields stored in POIs. Some fields are given more context here.

- Date
    - The primary documents did not always have an exact date (day, month, year) included. Rather than project undocumented specificity, the team developed a custom date grammar that allows for flexibility in recording timing of an event.
    - Here is the help text for entering a date that describes the grammar:
        - Enter the date the event happened using the following options without brackets: [Year], [Modifier1 Year], [Year Month], [Year Modifier1 Month], [Year Month Date], or [Year Month Modifier2 Date].

          Examples: "2001", "Late 2001", "2001 October", "2001 Mid October", "2001 October 27", "2001 October Around 27".

          - Year must be an integer between 1850 and 9999 (inclusive).
          - Month is the non-abbreviated spelling of one of the twelve months (e.g. January).
          - Date is an integer between 1 and 31 (or 30).
          - Modifier1 is either: "Early", "Mid", "Late" (without quotes).
          - Modifier2 is "Around" (without quotes).

There are additional second-level fields that are outlined here.

- Victim/Aggressor (Person)
    - Name
    - Race
    - Ethnicity
    - Age
    - Nationality
    - Gender
    - Occupation

API

Most of the data presentation and modification in the data collection application is not done through the API. Instead, presentation and modification is managed by Servlets through Apache Tomcat. The data collection application follows a standard CRUD model.

The API is fairly simple. It exposes an endpoint for all POIs and a parameterized one to return individual POIs. The parameterized endpoint also allows for field projection, returning only the desired fields. Below these endpoints are listed:

- GET /pois
- GET /pois/:id
- GET /pois/:id?f=[fields separated by commas][1]

In addition, there are search endpoints.[2] These allow for full-text searching of the database. Users issue a POST request to the search endpoint to create a search event. They then issue a GET request to the id of the corresponding search event to retrieve all the matching POIs. The endpoints are:

- POST /searches
- GET /searches/:id

(I am currently reviewing the code to determine the valid form of the body of the POST request for creating searches.)

[1] I need to double check the query parameters format.
[2] These will probably be replaced by running Solr on top of Mongo.