Digital Democracy Project

By

Daniel J. Mangin

Computer Science Department

College of Engineering

California Polytechnic State University

2014-

Advisor:_______________________________

Date Submitted:________________________

Contents

ABSTRACT

Digital Democracy was a project that was funded by the government to have Cal Poly create a website that would make the government of California more open to the general public. Behind the website, the project needed a database to house the necessary information that the site would display and continually update to reflect the most recent proceedings that have occurred. I was hired to be in-charge of maintaining the database, creating scripts that would grab the data we need, and have it update with new information daily.

Digital Democracy Project

Introduction

Digital Democracy is a project whose goal is to make the Legislation more transparent to the people of California by providing them with all of the necessary information in an accessible and user-friendly way. This project originally started in 2014 when Cal Poly received a $1.2 million grant to work on the project. In 2015, I was hired to continue work on maintaining the database for the project as well as continue to get the necessary information for the database.

Digital Democracy originally had only two teams, one that dealt with the data side and one that was involved with the Front-end. However, when the winter quarter was nearing an end, the project managers had hired a larger amount of individuals onto the project, and they had to create a set of teams to maintain organization and efficiency within the project. The teams are as follows:

- Database Team
- Transcription Tool Team
- Drupal Front-End Team
- IT Team
- NLP (Natural Language Processing) Team
- Mechanical Turk Team
- Video Team

While I was not formally on any team initially, I was in-charge of the database throughout the year of employment and I was the sole member of the database team.

Requirements

The goal of Digital Democracy is to provide the user with clear information about the hearings that take place during Congress and have information about the individuals that appear and are involved in these hearings for the user to examine in more detail. Since I was involved with the database, I had to make sure that the database had the correct and cleaned data to provide to the Front-End team of the project.

While I did not have to get information that the hearing was happening or what the people in the hearing were saying (which we call Utterances), I did have to collect information surrounding the hearing. The transcription Tool users would get the Hearing, Utterances, associate the information I provide with the Hearing. I had to collect information about:

- Committees
- Bills
- Legislators
- People

For the Front-End team, I had to collect a lot more information for them to use. I had to make sure that they had information about:

- People
- Legislators
- Lobbyist Information
- Committees
- Bills
- Bill Versions
- Authorships

In addition to collecting all the necessary information listed, I have to modify and add tables as needed throughout the project. While I consulted Prof. Dekhtyar when creating the new tables, I was in-charge of maintaining the structure of the database and ensuring there were no database anomalies. This required me to keep an ER Diagram of the database for ensuring relationships were present and were created logically.

Finally, I had to keep the database filled with the most up-to-date information available every night and maintain backups of the database before every update. For this, I had to set up a normal routine that would keep the database up-to-date and I had to ensure that all the sources I collected my data from were also continually updated as well.

Tools and Tasks

For the database and the transcription tool, we shared an instance of an Amazon Web Service server that we were able to keep our files in and I was able to keep the instances of our databases on that server. The Front-End teams had their own server that they used for themselves and the Back-End teams never touched or had access to. It was on the AWS server that I kept all my scripts, ran them, and set up their automation. I also maintain a git repository of all my files on the server to ensure other people could see the database and to make it easier to communicate with other teams and the leaders of the project.

To gather the data, I needed a language that worked very well for scripting, has the ability to grab information from a wide variety of sources, and could be run on the AWS for automation. The clear choice was Python for this task as it was able to fulfill all of these requirements and could have its functionality extended with the use of plugins and installable packages. The default version of Python was insufficient for the tasks I had to do, so I had to download and install the following plugins on the server:

- Mysql-connector: This was used to improve the Python functionality of connecting to a server and inserting the necessary data into it. Python does come with a default package that could connect to a database, but was unable to connect to a database with restrictions.
- Libxml: A package that provided a way to parse xml files. This was used to clean up the bill texts as they were in xml.

Sources

To gather all the data I needed, I had to collect information from a large variety of sources. Some of these were provided by the government, some were provided by third-party sites, and others were provided internally by other teams.

Leginfo

Tables filled:

- Person
- Legislator
- Term
- Bill
- BillVersion
- Action
- votesOn
- authors
- CommitteeAuthors
- BillVoteSummary
- BillVoteDetail

This was the primary database we used for the project. On the Official California Legislative Information website, they have an ftp server that contains the database files and updates for their database. This database is updated daily with daily files that are sent Monday through Saturday and finally having a full copy of the entire database that is updated every Sunday.

Their database is named capublic with 17 tables containing the necessary data. The daily files add data to the most recent Sunday release and are updated every week on that day late at night. Since capublic allowed for duplicate data to be inserted, the capublic database needed to be entirely deleted every Sunday and re-imported. The daily files just needed to be inserted every day at an hour that ensures that they will make it when the main database does a full update.

The data present in capublic was enough to completely fill all of the tables above with the necessary information. Even though capublic had all of the information we needed already present in the tables, there were numerous problems with capublic that caused us to build new tables for our use. Such examples include the large amount of redundant data present within the tables, the absence of foreign keys in any of the tables, and the poor naming conventions that were used for their tables.

Cal-Access

Tables filled:

- Lobbyist
- Person
- LobbyingFirm
- LobbyistEmployer
- LobbyistEmployment
- LobbyistDirectEmployment
- LobbyingContracts

This was a collection of CSV files containing numerous records for California Legislator. We did not use the majority of these files, instead focusing on one CSV file that logged the registering of lobbyists and companies and the various connections they had with each other. Each of these entries was categorized with a system that had to be parsed so we could divide the entries and place them into the correct tables.

The information provided by the CSV file was sufficient to completely fill the table with all the necessary information. It was possible to retrieve new people from this source, so when we did, we had to add them into the database along with their relevant information.

Sunlight Foundation

The Sunlight Foundation is a site with an API that allows users to collect information about various aspects of Congress. Originally we were going to use this API to collect information about the Committees in California Legislator, but even though we were able to get it and parse the information, the data was not updated frequently enough. This was a problem, because new committees were formed and old committees had members replaced, but the data hadn't been changed. We waited for about a month for the new data until we decided that we should get the information elsewhere.

Despite not being what we hoped for when we needed committees, the API does provide us with the information we need about Districts. We were originally just putting the data directly to the site, but it was decided that we should instead store them into the database. The districts provided the team with the necessary geo-Coordinate data to create a visual map of the districts that was on the site.

Assembly/Senator Website

Tables filled:

- Committee
- servesOn

These sites were used after the issues we had with OpenState and successfully getting the committees and the members of each one. The two sites maintain a list of the various committees that each house has (Assembly or Senate) and the members and positions of the members on that committee. Each one of these committees is organized into a type of Committee that is sorted on the websites main page.

Using python, I was able to scrape these pages for the information, then parse and input it into the database. From the homepage, I used the scripts to grab the names of the committees and what type they were. After this, I followed the embedded link on the site to a list of people that served on the Committee. We grabbed this information to find out who served on each committee.

Excel/CSV Files

Tables Filled:

- Legislator
- BillVersion
- Committee

Throughout the project, I was sent a CSV file with some information that had to be inputted into the database. This happened for many different tables, and had to either create a new script or adapt an old one to accommodate the new CSV files that were given to me. These were mostly sent to me by other people on the team who had collected a large amount of information, then organized it into a spreadsheet in excel.

For the excel sheets, I was able to save them as a CSV file, which python could read, then insert into the database. However, files saved from excel still posed problems because excel would use custom characters for symbols such as quotation marks, apostrophes, and commas that were different than the standard ascii version of them. To remedy this problem, I just replaced these special characters with a token that would be replaced with the ascii representation of that character.

Transcription Tool

Tables Filled:

- Person
- Utterance
- Hearing

- CommitteeHearing
- BillDiscussion

A tool that is ran and maintained by a separate team, but still provides data for the Front-End to use. When a legislative hearing happens, the hearing is recorded and uploaded to youtube so it can be used in the sites video player. Each of these is sent through a program that converts the speech to text in segments. These segments are then included with a start and stop time in the video, and all of these segments are written to a .ttml file.

Several things are missing from the .ttml file when we get it from Green Button. We don't have the people who said it, what bills they are discussing, and the speech to text translation is usually pretty poor and cut in weird places. So, we have editors who use the tool to provide all this information we need and fix the utterances. Once the editor is done, the tool will save the utterance to the database along with all of the other information surrounding the Hearing.

Python Files Used

Here are the names and general descriptions of all the Python Files that I used for gathering the data for Digital Democracy:

File: Action_Extract.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Inserts the Actions from the bill_history_tbl from capublic into DDDB2015Apr.Action
- This script runs under the update script
- Fills table:
    Action (bid, date, text)

Sources:
- Leginfo (capublic)
    - Pubinfo_2015.zip
    - Pubinfo_Mon.zip
    - Pubinfo_Tue.zip
    - Pubinfo_Wed.zip
    - Pubinfo_Thu.zip
    - Pubinfo_Fri.zip
    - Pubinfo_Sat.zip

-capublic
    - bill_history_tbl

File: Author_Extract.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Inserts the authors from capublic.bill_version_authors_tbl into the DDDB2015Apr.authors or DDDB2015Apr.committeeAuthors
- This script runs under the update script
- Fills table:
    - authors (pid, bid, vid, contribution)
    - CommitteeAuthors (cid, bid, vid)

Sources:
- Leginfo (capublic)
    - Pubinfo_2015.zip
    - Pubinfo_Mon.zip
    - Pubinfo_Tue.zip
    - Pubinfo_Wed.zip
    - Pubinfo_Thu.zip
    - Pubinfo_Fri.zip
    - Pubinfo_Sat.zip

- capublic
    - bill_version_author_tbl

File: Bill_Extract.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Inserts the authors from capublic.bill_tbl into DDDB2015Apr.Bill and capublic.bill_version_tbl into DDDB2015Apr.BillVersion
- This script runs under the update script
- Fills table:
    - Bill (bid, type, number, state, status, house, session)
    - BillVersion (vid, bid, date, state, subject, appropriation, substantive_changes)

Sources:
- Leginfo (capublic)
    - Pubinfo_2015.zip
    - Pubinfo_Mon.zip
    - Pubinfo_Tue.zip
    - Pubinfo_Wed.zip
    - Pubinfo_Thu.zip
    - Pubinfo_Fri.zip

- Pubinfo_Sat.zip

-capublic
  - bill_tbl
  - bill_version_tbl

File: billparse.py
Author: ???
Date: 6/11/2015

Description:
- Takes the bill_xml column from the capublic.bill_version_tbl and inserts it into the appropriate columns in DDDB2015Apr.BillVersion
- This script runs under the update script
- Fills table:
  BillVersion (title, digest, text)

Sources:
- Leginfo (capublic)
  - Pubinfo_2015.zip
  - Pubinfo_Mon.zip
  - Pubinfo_Tue.zip
  - Pubinfo_Wed.zip
  - Pubinfo_Thu.zip
  - Pubinfo_Fri.zip
  - Pubinfo_Sat.zip

- capublic
  - bill_version_tbl

File: Cal-Access-Accessor.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Goes through the file CVR_REGISTRATION_CD.TSV and places the data into DDDB2015Apr
- This script runs under the update script
- Fills table:
  LobbyingFirm (filer_naml, filer_id, rpt_date, ls_beg_yr, ls_end_yr)
  Lobbyist (pid, filer_id)
  Person (last, first)
  LobbyistEmployer (filer_naml, filer_id, coalition)
  LobbyistEmployment (pid, sender_id, rpt_date, ls_beg_yr, ls_end_yr)
  LobbyistDirectEmployment (pid, sender_id, rpt_date, ls_beg_yr, ls_end_yr)

LobbyingContracts (filer_id, sender_id, rpt_date, ls_beg_yr, ls_end_yr)

Sources:
- db_web_export.zip (California Access)
    - CVR_REGISTRATION_CD.TSV

File: Committee_CSV_Extract.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Goes through the file Committee_list.csv and places the data into DDDB2015Apr
- This script is used to get type of Committee
- Fills table:
    Committee(Type)

Sources:
- Committee_list.csv

DEPRECIATED SCRIPT. We use Get_Committees_Web.py instead
File: Committee_CSV_Extract.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Gathers JSON data from OpenState and fills DDDB2015Apr.servesOn
-Used for daily update DDDB2015Apr
- Fills table:
    servesOn (pid, year, district, house, cid)

Sources
- OpenState

DEPRECIATED SCRIPT. We use Get_Committees_Web.py instead
File: Committee_CSV_Extract.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Gathers JSON data from OpenState and fills DDDB2015Apr.Committee

- used for daily update of DDDB2015Apr
- Fills table:
  Committee (cid, house, name)

Sources
- OpenState

File: Get_Committees_Web.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Scrapes the Assembly and Senate Websites to gather current Committees and Membership and place them into
  DDDB2015Apr.Committee and DDDB2015Apr.servesOn
- Used for daily update Script
- Fills table:
  Committee (cid, house, name)
  servesOn (pid, year, district, house, cid)

Sources
- California Assembly Website
- California Senate Website

File: Get_Districts.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Gathers JSON data from OpenState and fills DDDB2015Apr.District
- Used in the daily update script
- Fills table:
  District (state, house, did, note, year, geodata, region)

Sources:
- OpenState

DEPRECIATED SCRIPT, Use insert_Contributions_3_CSV.py
File: insert_Contributions.py

Author: Daniel Mangin
Date: 6/11/2015

Description:
- Gathers Contribution Data and puts it into DDDB2015.Contributions
- Used once for the Contributions.json
- Fills table:
    Contribution (pid, year, house, contributor, amount)

Source
- Contributions.json

File: insert_Contributions_CSV.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Gathers Contribution Data and puts it into DDDB2015.Contributions
- Used once for the Insertion of all the Contributions
- Fills table:
    Contribution (id, pid, year, date, house, donorName, donorOrg, amount)

Sources:
- Maplight Data
    - cand_2001.csv
    - cand_2003.csv
    - cand_2005.csv
    - cand_2007.csv
    - cand_2009.csv
    - cand_2011.csv
    - cand_2013.csv
    - cand_2015.csv

File: insert_Gifts.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Gathers Gift Data and puts it into DDDB2015.Gift
- Used once for the Insertion of all the Gifts
- Fills table:
    Gift (pid, schedule, sourceName, activity, city, cityState, value, giftDate, reimbursed, giftIncomeFlag, speechFlag, description)

Source:
- Gifts.txt

File: Leg_Bio_Extractor.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Gathers Senate and Assembly Biographies and puts it into DDDB2015.Legislator.Bio
- Used once for the Insertion of all OfficialBios
- Fills table:
    - Legislator (OfficialBio)

Sources:
- Senate_and_Assembly_Biographies.csv

File: legislator_migrate.py
Author: ???
Date: 6/11/2015

Description:
- Gathers Legislator Data from capublic.legislator_tbl and inserts the data into DDDB2015Apr.Person, DDDB2015Apr.Legislator, and DDDB2015Apr.Term
- Used in the daily update of DDDB2015Apr
- Fills table:
    - Person (last, first)
    - Legislator (pid)
    - Term (pid, year, district, house, party)

Sources:
- Leginfo (capublic)
    - Pubinfo_2015.zip
    - Pubinfo_Mon.zip
    - Pubinfo_Tue.zip
    - Pubinfo_Wed.zip
    - Pubinfo_Thu.zip
    - Pubinfo_Fri.zip
    - Pubinfo_Sat.zip

-capublic
    - legislator_tbl

File: Legislators.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Gets a list of random Legislators and outputs them to a text file
- Only used for providing names for testing reasons

File: Lobbying_Firm_Name_Fix.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Used to correct names of Lobbying Firms gathered during the Lobbying Info
- Used as an import to the Cal-Access-Accessor.py to clean Lobbying Firm Names

File: Motion_Extract.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Gathers the Motions from capublic.bill_motion_tbl and inserts the Motions into DDDB2015Apr.Motion
- Used in the daily update of DDDB2015Apr
- Fills table:
  - Motion (mid, date, text)

Sources:
- Leginfo (capublic)
  - Pubinfo_2015.zip
  - Pubinfo_Mon.zip
  - Pubinfo_Tue.zip
  - Pubinfo_Wed.zip
  - Pubinfo_Thu.zip
  - Pubinfo_Fri.zip
  - Pubinfo_Sat.zip

-capublic
  - bill_motion_tbl

File: Name_Fixes_Legislator_Migrate.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Used when Legislator names change in capublic to what we have in DDDB2015Apr
- Used in legislator_migrate.py to adjust the names if they are the same Person

File: Person_Name_Fix.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Cleans the all capitalized names in the Person table and revertes them to their proper titling
- Included in Cal-Access-Accessor.py to clean up Lobbyist Names

File: Test-Randomizer.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Used to grab 25% of random values form DDDB2015Apr
- Used for testing reasons on Drupal

File: Vote_Extract.py
Author: Daniel Mangin
Date: 6/11/2015

Description:
- Gets the Vote Data from capublic.bill_summary_vote into DDDB2015Apr.BillVoteSummary and capublic.bill_detail_vote into DDDB2015Apr.BillVoteDetail
- Used in daily update of DDDB2015Apr
- Fills Tables:
    - BillVoteSummary (bid, mid, cid, VoteDate, ayes, naes, abstain, result)
    - BillVoteDetail (pid, voteId, result)

Sources:
- Leginfo (capublic)
    - Pubinfo_2015.zip
    - Pubinfo_Mon.zip

- Pubinfo_Tue.zip
- Pubinfo_Wed.zip
- Pubinfo_Thu.zip
- Pubinfo_Fri.zip
- Pubinfo_Sat.zip

-capublic
- bill_summary_vote_tbl
- bill_detail_vote_tbl

Successful Events in Project

The project has been overall quite a success, with all of the data that was necessary being successfully added into the database.

The biggest successful event in the project was our presentation to Congress about the progress of our project and giving them a tour of the site. During this day, several people from our team went down to Sacramento to present Congress with the current product and give them a tour of it. I was unable to go, but from what the people who went said, it seems that they liked the progress of the project so far. Even better, the worst bug when we presented it to them was a misspelling on the main page of the site.

Another successful moment in the project was when the site went live and became available for the pubic to see. We were especially nervous, because it would have been very embarrassing to have incorrect data showing to the general public. This would not only hurt the project by making it unreliable, it would also hurt the reputation of Cal Poly. However, the site displayed fine, and we have a large number of people who visited the site (we saw through Google Analytics). We even got positive press about the launch of our site from many newspapers. This was extremely encouraging, and gave the whole team good spirits.

Issues and Resolutions in Project

Despite the overall success of the project, there were several set-backs and issues that occurred within the project.

The worst event to occur during the project was when we had to change to using a new database and sync the new one with the old one. At the beginning of the year, we initially were using a database called DDDB2015. This was a new version of an even older database that we used during the first builds of the site and while we were trying to meet the projects February deadline and complete the tasks we had to have done by then. After the deadline goals were met, we had to modify the database to accommodate new information, which meant modifying old tables and creating new ones.

However, we couldn't just remove the old database, as people from the transcription tool were still using it to insert Utterances.

So to do this, we would create a mock version of the database, then on a day the transcription tool owner and I decided, we would cut the transcription tool and the database for a while, move everything over from DDDB2015 to DDDB2015Apr, then turn it back on again. It seemed like the simplest way to do it if I did it right. However, I messed up the script for importing the tables over to DDDB2015Apr, and the tables did not import correctly. This was a huge issue, as we lost a lot of data from this when the transcription tool user swapped over the new database and couldn't find everything.

To solve this, we had to bring it back down during the day, and I had to revisit the scripts and fix the issues that were causing the loss of data. Then we had to delete the new tables, which caused some people to lose whatever data they had inserted into the table after the system was brought back up. After I had run the scripts to re-import all of the data, I had to quickly manually look over all the data transferred to make sure it was all correct. After I was satisfied, the transcription tool owner turned back on access to the server. To make sure that something like this wouldn't happen to the database again, I now have scripts that will run a check over every table and keep a crontab that automatically backs up the database every night.

Future Improvements

Despite all of my work on the project, there are several tasks that weren't completed yet and there are improvements that can be made to the project on the future.

One such improvement is a user-friendly Drupal interface that allows the user to visually act with the updating scripts. The user would be able to update the database at their convenience, and the module would show the users things that the update has changed in the database in the module. Things displayed would be a log of what values were inserted, deleted, or updated and the status of the last update. It would also have access to the database backups and restore the database to one of those backups if necessary.

Another improvement is the addition of analytics to the data that we retrieve. While having the raw data is nice, it would be even better if we were able to have some statistics on the data, such as which Bill was the legislative most divided on.

Documentation is another thing that my part of the project could improve on. Since I was the entirety of the database team, I found it unnecessary to use source control or to comment my files. I was the only one doing anything with them and nobody else needed to see my files. Since I will be leaving, new people will have to use the old code that I have written. Before I leave college, I have to clean my code considerably as well as document it for their ease of use.

However, I cannot make any of these changes, as I will be leaving Digital Democracy at the end of the year when I graduate. So, they have hired two new people, Mandy Chan and Andrew Voorhees to continue work on the project when I have left. Before I leave, I will train them so they fully understand the database and all the relations the data has with each other as well as the scripts used and the updating system that is in place. This will allow them to maintain the database after I am gone, and continue working on improving the current system with the above improvements.

Conclusion

While all of the goals that were set for the project were not completed, I was still able to complete the necessary requirements needed from me to provide the back-end of the Digital Democracy site. Since we were able to present the site to Congress and go live with it successfully, I would still call my part in the project a success.

Appendix

List of Tables Used

The tables used for this project are listed underneath here raw from the DB-create.sql file so anyone who wishes to examine them will have a better understanding of them.

```
-- file: DB-setup.sql
-- author: Daniel Mangin
-- date: 6/11/2015
-- Description: Used to create all of the tables for Digital Democracy
-- note: this will only work on the currently used database

CREATE TABLE IF NOT EXISTS Person (
   pid    INTEGER AUTO_INCREMENT,
   last   VARCHAR(50) NOT NULL,
   first  VARCHAR(50) NOT NULL,
   image VARCHAR(256),
   -- description VARCHAR(1000), deprecated, moved to legislator profile

   PRIMARY KEY (pid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

-- we can probably remove this table and roll into a "role" field in Person
-- alternatively, make it a (pid, jobhistory) and put term for the hearing in
here
CREATE TABLE IF NOT EXISTS Legislator (
   pid         INTEGER AUTO_INCREMENT,
   description VARCHAR(1000),
   twitter_handle VARCHAR(100),
   capitol_phone  VARCHAR(30),
   website_url    VARCHAR(200),
   room_number    INTEGER,
   email_form_link VARCHAR(200),
   OfficialBio TEXT,

   PRIMARY KEY (pid),
   FOREIGN KEY (pid) REFERENCES Person(pid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

-- we can probably remove this table and roll into a "role" field in Person
-- alternatively, we can make it (pid, jobhistory) and put client information
here
```

```
-- CREATE TABLE IF NOT EXISTS Lobbyist (
--   pid    INTEGER,

--   PRIMARY KEY (pid),
--   FOREIGN KEY (pid) REFERENCES Person(pid)
-- )

-- ENGINE = INNODB
-- CHARACTER SET utf8 COLLATE utf8_general_ci;

-- only Legislators have Terms
CREATE TABLE IF NOT EXISTS Term (
   pid      INTEGER,
   year     YEAR,
   district INTEGER(3),
   house    ENUM('Assembly', 'Senate') NOT NULL,
   party    ENUM('Republican', 'Democrat', 'Other') NOT NULL,
   start    DATE,
   end      DATE,

   PRIMARY KEY (pid, year, district, house),
   FOREIGN KEY (pid) REFERENCES Legislator(pid) -- change to Person
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS Committee (
   cid    INTEGER(3),
   house  ENUM('Assembly', 'Senate', 'Joint') NOT NULL,
   name   VARCHAR(200) NOT NULL,
   Type   ENUM('Standing','Select','Budget Subcommittee','Joint'),

   PRIMARY KEY (cid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS servesOn (
   pid      INTEGER,
   year     YEAR,
   district INTEGER(3),
   house    ENUM('Assembly', 'Senate') NOT NULL,
   cid      INTEGER(3),

   PRIMARY KEY (pid, year, district, house, cid),
   FOREIGN KEY (pid, year, district, house) REFERENCES Term(pid, year,
district, house),
   FOREIGN KEY (cid) REFERENCES Committee(cid)
)
ENGINE = INNODB
```

```
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS Bill (
   bid     VARCHAR(20),
   type    VARCHAR(3) NOT NULL,
   number  INTEGER NOT NULL,
   state   ENUM('Chaptered', 'Introduced', 'Amended Assembly', 'Amended
Senate', 'Enrolled',
      'Proposed', 'Amended', 'Vetoed') NOT NULL,
   status  VARCHAR(60),
   house   ENUM('Assembly', 'Senate', 'Secretary of State', 'Governor',
'Legislature'),
   session INTEGER(1),

   PRIMARY KEY (bid),
   INDEX name (type, number)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS Hearing (
   hid    INTEGER AUTO_INCREMENT,
   date   DATE,

   PRIMARY KEY (hid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS CommitteeHearings (
        cid INTEGER,
        hid INTEGER,

        PRIMARY KEY (cid, hid),
        FOREIGN KEY (cid) REFERENCES Committee(cid),
        FOREIGN KEY (hid) REFERENCES Hearing(hid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS JobSnapshot (
   pid   INTEGER,
   hid   INTEGER,
   role  ENUM('Lobbyist', 'General_public', 'Legislative_staff_commitee',
'Legislative_staff_author', 'State_agency_rep', 'Unknown'),
   employer VARCHAR(50), -- employer: lobbyist: lobying firm, union,
corporation. SAR: name of Agency/Department. GP: teacher/etc.
   client   VARCHAR(50), -- client: only for lobbyist

   PRIMARY KEY (pid, hid),
   FOREIGN KEY (pid) REFERENCES Person(pid),
   FOREIGN KEY (hid) REFERENCES Hearing(hid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS Action (
   bid    VARCHAR(20),
   date   DATE,
   text   TEXT,

   FOREIGN KEY (bid) REFERENCES Bill(bid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS Video (
   vid INTEGER AUTO_INCREMENT,
   youtubeId VARCHAR(20),
   hid INTEGER,
   position INTEGER,
   startOffset INTEGER,
   duration INTEGER,

   PRIMARY KEY (vid),
   FOREIGN KEY (hid) REFERENCES Hearing(hid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS Video_ttml (
   vid INTEGER,
   ttml MEDIUMTEXT,

   FOREIGN KEY (vid) REFERENCES Video(vid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

-- examine (without startTime in UNIQUE, duplicate)
CREATE TABLE IF NOT EXISTS BillDiscussion (
   did         INTEGER AUTO_INCREMENT,
   bid         VARCHAR(20),
   hid         INTEGER,
   startVideo  INTEGER,
   startTime   INTEGER,
   endVideo    INTEGER,
   endTime     INTEGER,
   numVideos   INTEGER(4),
   PRIMARY KEY (did),
   UNIQUE KEY (bid, startVideo, startTime),
   FOREIGN KEY (bid) REFERENCES Bill(bid),
   FOREIGN KEY (hid) REFERENCES Hearing(hid),
   FOREIGN KEY (startVideo) REFERENCES Video(vid),
   FOREIGN KEY (endVideo) REFERENCES Video(vid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS Motion (
   mid    INTEGER(20),
   date   DATETIME,
   text   TEXT,

   PRIMARY KEY (mid, date)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS votesOn (
   pid    INTEGER,
   mid    INTEGER(20),
   vote   ENUM('Yea', 'Nay', 'Abstain') NOT NULL,

   PRIMARY KEY (pid, mid),
   FOREIGN KEY (pid) REFERENCES Legislator(pid),
   FOREIGN KEY (mid) REFERENCES Motion(mid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS BillVersion (
   vid                 VARCHAR(30),
   bid                 VARCHAR(20),
   date                DATE,
   state               ENUM('Chaptered', 'Introduced', 'Amended Assembly',
'Amended Senate',
                            'Enrolled', 'Proposed', 'Amended', 'Vetoed') NOT
NULL,
   subject             TEXT,
   appropriation       BOOLEAN,
   substantive_changes BOOLEAN,
   title               TEXT,
   digest              MEDIUMTEXT,
   text                MEDIUMTEXT,

   PRIMARY KEY (vid),
   FOREIGN KEY (bid) REFERENCES Bill(bid)
)
```

```
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS authors (
   pid          INTEGER,
   bid          VARCHAR(20),
   vid          VARCHAR(30),
   contribution ENUM('Lead Author', 'Principal Coauthor', 'Coauthor') DEFAULT
'Coauthor',

   PRIMARY KEY (pid, bid, vid),
   FOREIGN KEY (pid) REFERENCES Legislator(pid), -- change to Person
   FOREIGN KEY (bid, vid) REFERENCES BillVersion(bid, vid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS attends (
   pid    INTEGER,
   hid    INTEGER,

   PRIMARY KEY (pid, hid),
   FOREIGN KEY (pid) REFERENCES Legislator(pid), -- Person
   FOREIGN KEY (hid) REFERENCES Hearing(hid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

-- examine (without endTime in UNIQUE, duplicate)
CREATE TABLE IF NOT EXISTS Utterance (
   uid    INTEGER AUTO_INCREMENT,
   vid    INTEGER,
   pid    INTEGER,
   time   INTEGER,
   endTime INTEGER,
   text   TEXT,
   current BOOLEAN NOT NULL,
   finalized BOOLEAN NOT NULL,
   type   ENUM('Author', 'Testimony', 'Discussion'),
   alignment ENUM('For', 'Against', 'For_if_amend', 'Against_unless_amend',
'Neutral', 'Indeterminate'),
   dataFlag INTEGER DEFAULT 0,

   PRIMARY KEY (uid, current),
   UNIQUE KEY (uid, vid, pid, current, time),
   FOREIGN KEY (pid) REFERENCES Person(pid),
   FOREIGN KEY (vid) REFERENCES Video(vid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;
```

```
CREATE OR REPLACE VIEW currentUtterance
AS SELECT uid, vid, pid, time, endTime, text, type, alignment
FROM Utterance
WHERE current = TRUE AND finalized = TRUE ORDER BY time DESC;

-- tag is a keyword. For example, "education", "war on drugs"
-- can also include abbreviations for locations such as "Cal Poly" for "Cal
Poly SLO"
CREATE TABLE IF NOT EXISTS tag (
   tid INTEGER AUTO_INCREMENT,
   tag VARCHAR(50),

   PRIMARY KEY (tid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

-- join table for Uterrance >>> Tag
CREATE TABLE IF NOT EXISTS join_utrtag (
   uid INTEGER,
   tid INTEGER,

   PRIMARY KEY (uid, tid),
   FOREIGN KEY (tid) REFERENCES tag(tid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

-- an utterance might contain an honorific or a pronoun where it is unclear
who the actual person is
-- this is a "mention" and should be joined against when searching for a
specific person
CREATE TABLE IF NOT EXISTS Mention (
   uid INTEGER,
   pid INTEGER,

   PRIMARY KEY (uid, pid),
   FOREIGN KEY (pid) REFERENCES Person(pid),
   FOREIGN KEY (uid) REFERENCES Utterance(uid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS BillVoteSummary (
        voteId          INTEGER AUTO_INCREMENT,
        bid             VARCHAR(20),
        mid             INTEGER(20),
        cid             INTEGER,
        VoteDate        DATETIME,
        ayes            INTEGER,
        naes            INTEGER,
        abstain         INTEGER,
        result          VARCHAR(20),

        PRIMARY KEY(voteId),
        FOREIGN KEY (mid) REFERENCES Motion(mid),
        FOREIGN KEY (bid) REFERENCES Bill(bid),
        FOREIGN KEY (cid) REFERENCES Committee(cid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS BillVoteDetail (
        pid     INTEGER,
        voteId  INTEGER,
        result  VARCHAR(20),

        PRIMARY KEY(pid, voteId),
        FOREIGN KEY (voteId) REFERENCES BillVoteSummary(voteId)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS Gift (
        RecordId INTEGER AUTO_INCREMENT,
        pid INTEGER,
        schedule ENUM('D', 'E'), -- D is a normal gift whereas E is a travel
gift
        sourceName VARCHAR(50),
        activity VARCHAR(40),
        city VARCHAR(30),
        cityState VARCHAR(10),
        value DOUBLE,
        giftDate DATE,
        reimbursed TINYINT(1),
        giftIncomeFlag TINYINT(1) DEFAULT 0,
        speechFlag TINYINT(1) DEFAULT 0,
        description VARCHAR(80),

        PRIMARY KEY(RecordId),
        FOREIGN KEY (pid) REFERENCES Person(pid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS District (
        state VARCHAR(2),
        house ENUM('lower', 'upper'),
        did INTEGER,
        note VARCHAR(40) DEFAULT '',
        year INTEGER,
        region TEXT,
        geoData MEDIUMTEXT,

        PRIMARY KEY(state, house, did, year)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS Contribution (
        RecordId INTEGER AUTO_INCREMENT,
        pid INTEGER,
        law_eid INTEGER,
        d_id INTEGER,
        year INTEGER,
        house VARCHAR(10),
        contributor VARCHAR(50),
        amount DOUBLE,

        PRIMARY KEY(RecordId),
        FOREIGN KEY (pid) REFERENCES Person(pid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

-- Transcription Tool Tables
CREATE TABLE IF NOT EXISTS TT_Editor (
   id INTEGER AUTO_INCREMENT ,
   username VARCHAR(50) NOT NULL ,
   password VARCHAR(255) NOT NULL ,
   created TIMESTAMP NOT NULL ,
   active BOOLEAN NOT NULL ,
   role VARCHAR(15) NOT NULL ,

   PRIMARY KEY (id),
   UNIQUE KEY (username)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS TT_Task (
   tid INTEGER AUTO_INCREMENT ,
   hid INTEGER ,        -- added
   did INTEGER ,
   editor_id INTEGER ,
   name VARCHAR(255) NOT NULL ,
   vid INTEGER ,
   startTime INTEGER NOT NULL ,
   endTime INTEGER NOT NULL ,
   created DATE,
   assigned DATE,
   completed DATE,

   PRIMARY KEY (tid) ,
   FOREIGN KEY (did) REFERENCES BillDiscussion(did),
   FOREIGN KEY (editor_id) REFERENCES TT_Editor(id),
   FOREIGN KEY (vid) REFERENCES Video(vid),
   FOREIGN KEY (hid) REFERENCES Hearing(hid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS TT_TaskCompletion (
   tcid INTEGER AUTO_INCREMENT ,
   tid INTEGER ,
   completion DATE ,

   PRIMARY KEY (tcid),
   FOREIGN KEY (tid) REFERENCES TT_Task(tid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS user (
   email VARCHAR(255) NOT NULL,
   name VARCHAR(255),
   password VARCHAR(255) NOT NULL,
   new_user INTEGER,

   PRIMARY KEY (email)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS LobbyingFirm(
   filer_naml VARCHAR(200),
   filer_id VARCHAR(9)  PRIMARY KEY,  -- modified  (PK)
   rpt_date DATE,
   ls_beg_yr INTEGER,    -- modified (INT)
   ls_end_yr INTEGER     -- modified (INT)
);

--  ALTER TABLE !!!!!
CREATE TABLE IF NOT EXISTS Lobbyist(
   pid INTEGER,   -- added
   -- FILER_NAML VARCHAR(50),               modified, needs to be same as
Person.last
   -- FILER_NAMF VARCHAR(50),               modified, needs to be same as
Person.first
```

```
   filer_id VARCHAR(9) UNIQUE,         -- modified
   PRIMARY KEY (pid),                    -- added
   FOREIGN KEY (pid) REFERENCES Person(pid)
);

CREATE TABLE IF NOT EXISTS LobbyistEmployer(
   filer_naml VARCHAR(200),
   filer_id VARCHAR(9),  -- modified (PK)
   le_id INTEGER AUTO_INCREMENT,
   coalition TINYINT(1),

   PRIMARY KEY (le_id)
);

-- LOBBYIST_EMPLOYED_BY_LOBBYING_FIRM

CREATE TABLE IF NOT EXISTS LobbyistEmployment(
   pid INT  REFERENCES  Person(pid),                         -- modified (FK)
   sender_id VARCHAR(9) REFERENCES LobbyingFirm(filer_id), -- modified (FK)
   rpt_date DATE,
   ls_beg_yr INTEGER,    -- modified (INT)
   ls_end_yr INTEGER,    -- modified (INT)

   PRIMARY KEY (pid, sender_id, rpt_date, ls_end_yr), -- modified (May 21)
   FOREIGN KEY (sender_id) REFERENCES LobbyingFirm(filer_id)
);

-- NEW TABLE: Lobbyist Employed Directly by Lobbyist Employers
-- Structure same as LOBBYIST_EMPLOYED_BY_LOBBYING_FIRM,
-- but the SENDER_ID is a Foreign Key onto LOBBYIST_EMPLOYER
--  LOBBYIST_EMPLOYED_BY_LOBBYIST_EMPLOYER

CREATE TABLE IF NOT EXISTS LobbyistDirectEmployment(
   pid INT  REFERENCES  Person(pid),
   sender_id VARCHAR(9) REFERENCES LobbyistEmployer(filer_id),
   rpt_date DATE,
   ls_beg_yr INTEGER,    -- modified (INT)
   ls_end_yr INTEGER,     -- modified (INT)
   PRIMARY KEY (pid, sender_id, rpt_date, ls_end_yr) -- modified (May 21)
);

-- end new table


CREATE TABLE IF NOT EXISTS LobbyingContracts(
   filer_id VARCHAR(9) REFERENCES LobbyingFirm(filer_id),     -- modified
(FK)
   sender_id VARCHAR(9) REFERENCES LobbyistEmployer(filer_id), -- modified
(FK)
   rpt_date DATE,
   ls_beg_yr INTEGER,    -- modified (INT)
   ls_end_yr INTEGER,     -- modified (INT)
   PRIMARY KEY (filer_id, sender_id, rpt_date) -- modified (May 21)
);

CREATE TABLE IF NOT EXISTS LobbyistRepresentation(
   pid INTEGER REFERENCES Person(pid),                  -- modified
   le_id INTEGER, -- modified (renamed)
   hearing_date DATE,                                       -- modified
(renamed)
   hid INTEGER,              -- added

   PRIMARY KEY(pid, le_id, hid),                -- added
   FOREIGN KEY (hid) REFERENCES Hearing(hid),
   FOREIGN KEY (le_id) REFERENCES LobbyistEmployer(le_id)
);

CREATE TABLE IF NOT EXISTS GeneralPublic(
   pid INTEGER,   -- added
   affiliation VARCHAR(256),
   position VARCHAR(100),
   RecordId INTEGER AUTO_INCREMENT,
   hid   INTEGER,                                              -- added

   PRIMARY KEY (RecordId),
   FOREIGN KEY (pid) REFERENCES Person(pid),
   FOREIGN KEY (hid) REFERENCES Hearing(hid)
);

CREATE TABLE IF NOT EXISTS LegislativeStaff(
   pid INTEGER,   -- added
   flag TINYINT(1),  -- if flag is 0, there must be a legislator; if flag is
1, there must be a committee
   legislator INTEGER, -- this is the legislator
   committee INTEGER,

   PRIMARY KEY (pid),                  -- added
   FOREIGN KEY (pid) REFERENCES Person(pid),
   FOREIGN KEY (legislator) REFERENCES Person(pid),
   FOREIGN KEY (committee) REFERENCES Committee(cid),
   CHECK (Legislator IS NOT NULL AND flag = 0 OR committee IS NOT NULL AND
flag = 1)
);

CREATE TABLE IF NOT EXISTS LegislativeStaffRepresentation(
   pid INTEGER,   -- added
   flag TINYINT(1),  -- if flag is 0, there must be a legislator; if flag is
1, there must be a committee
   legislator INTEGER, -- this is the legislator
   committee INTEGER,
   hid   INTEGER,                                              -- added

   PRIMARY KEY (pid, hid),                    -- added
   FOREIGN KEY (pid) REFERENCES Person(pid),
   FOREIGN KEY (legislator) REFERENCES Person(pid),
   FOREIGN KEY (hid) REFERENCES Hearing(hid),
   FOREIGN KEY (committee) REFERENCES Committee(cid),
   CHECK (Legislator IS NOT NULL AND flag = 0 OR committee IS NOT NULL AND
flag = 1)
);

CREATE TABLE IF NOT EXISTS LegAnalystOffice(
   pid INTEGER REFERENCES Person(pid),

   PRIMARY KEY (pid),                    -- added
   FOREIGN KEY (pid) REFERENCES Person(pid)
);

CREATE TABLE IF NOT EXISTS LegAnalystOfficeRepresentation(
   pid INTEGER REFERENCES Person(pid),   -- added
   hid   INTEGER,                                      -- added

   PRIMARY KEY (pid, hid),                    -- added
   FOREIGN KEY (pid) REFERENCES Person(pid),
   FOREIGN KEY (hid) REFERENCES Hearing(hid)
);

CREATE TABLE IF NOT EXISTS StateAgencyRep(
   pid INTEGER,   -- added
   employer VARCHAR(256),
   position VARCHAR(100),

   PRIMARY KEY (pid),                    -- added
   FOREIGN KEY (pid) REFERENCES Person(pid)
);

CREATE TABLE IF NOT EXISTS StateAgencyRepRepresentation(
   pid INTEGER,   -- added
   employer VARCHAR(256),
   position VARCHAR(100),
   hid   INTEGER,                                              -- added

   PRIMARY KEY (pid, hid),                    -- added
   FOREIGN KEY (pid) REFERENCES Person(pid),
   FOREIGN KEY (hid) REFERENCES Hearing(hid)
);

CREATE TABLE IF NOT EXISTS StateConstOffice(
   pid INTEGER,
   office VARCHAR(200),
   position VARCHAR(200),

   PRIMARY KEY (pid),                    -- added
   FOREIGN KEY (pid) REFERENCES Person(pid)
);

CREATE TABLE IF NOT EXISTS StateConstOfficeRepresentation(
   pid INTEGER,
   office VARCHAR(200),
   position VARCHAR(200),
   hid INTEGER,

   PRIMARY KEY (pid, hid),                    -- added
   FOREIGN KEY (pid) REFERENCES Person(pid),
   FOREIGN KEY (hid) REFERENCES Hearing(hid)
);

CREATE TABLE IF NOT EXISTS BillDisRepresentation(
        did INTEGER,
        pid INTEGER,
        le_id INTEGER,
        hid INTEGER,

        PRIMARY KEY (did, pid, le_id, hid),
        FOREIGN KEY (did) REFERENCES BillDiscussion(did),
        FOREIGN KEY (pid) REFERENCES Person(pid),
        FOREIGN KEY (le_id) REFERENCES LobbyistEmployer(le_id),
        FOREIGN KEY (hid) REFERENCES Hearing(hid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;

CREATE TABLE IF NOT EXISTS CommitteeAuthors(
        cid INTEGER,
        bid VARCHAR(20),
        vid VARCHAR(30),

        PRIMARY KEY(cid, bid, vid),
        FOREIGN KEY (bid) REFERENCES Bill(bid),
        FOREIGN KEY (cid) REFERENCES Committee(cid),
        FOREIGN KEY (vid) REFERENCES BillVersion(vid)
)
ENGINE = INNODB
CHARACTER SET utf8 COLLATE utf8_general_ci;
```

Sources Websites

Leginfo: ftp://www.leginfo.ca.gov/pub/bill/

Cal-Access:

- Documentation: http://campaignfinance.cdn.sos.ca.gov/calaccess-documentation.zip

- Files: http://campaignfinance.cdn.sos.ca.gov/dbwebexport.zip

Assembly Website: assembly.ca.gov/

Senate Website: senate.ca.gov/

Openstate: https://sunlightlabs.github.io/openstates-api/districts.html

Maplight: http://maplight.org/data/get/california-money-and-politics-dataset