Prof. Nick Roussopoulos

CMSC424 Term Project

U.S. Presidential Election Result Database

Yufan Fei, Yufang Feng

Table of Content

1.1 Description of purpose of the document

The purpose of this document is to provide detailed requirement and design specifications as well as to describe the implementation process and result for the Presidential Election Data Project. In this document, there is a description of how we performed the ETL (Extract-Transform-Load) tool and process, a description of the design documents and activities within the project, the function of the design the development phases.

1.2 Purpose of the project

The first purpose of the system is that to learn and practice in the ETL process which included collecting data from the internet (extract), organize and clean the data (transform), and load into to our designed database in an organized manner (load). After establishing a reliable database, we will run queries on the database to get various aspects of information the user needs and an interactive web server to provide limited knowledge to the user with customized variables among queries.

2 The sites and sources being utilized for the system

--- Wikipedia/poll

The U.S poll wiki page synchronize almost all of the U.S presidential result starting from 1936, which is the first year when the presidential poll officially launch.

--- archives.gov

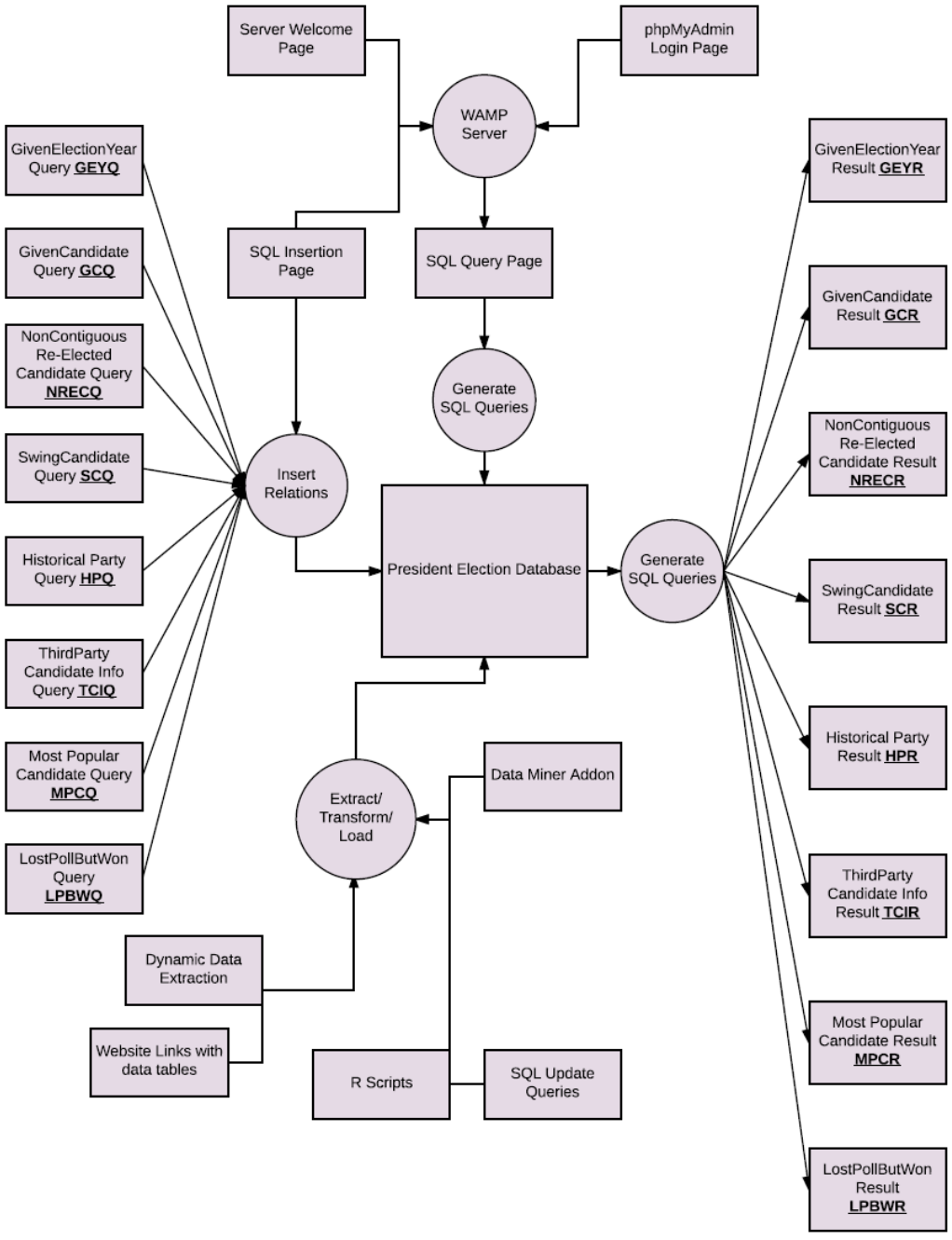
The National Archives and Records Administration preserves U.S. government records, manages the Presidential Libraries system, and publishes laws, regulations, Presidential, and other public documents. It provides vote details per year from state view as well.

3 Assumptions about the system

The major limitation of the system is that as designers, we didn't have any web server experience before, such that loading data into the database became extremely

difficult at first, until Fei figured out using his R background to tackle the data manipulation task.

4 The top-level information flow diagram



5 The list of tasks

5.1 Build a Server

TASK NUMBER: BAS

TASK NAME: Build a Server with MySql installed that allows users to access from anywhere

PERFORMER: EC2 Apache Server

PURPOSE: Create the server for backend and frontend jobs.

ENABLING COND: User accessing the web interface.

DESCRIPTION: find a remote server that hosts our website and the database

FREQUENCY: Once finished the job

DURATION: Very short

IMPORTANCE: Critical

MAXIMUM DELAY: 10 seconds

INPUT: None

OUTPUT: Welcome Page

DOCUMENT USE: WIFWF: Web Interface Welcome Form

OPS PERFORMED: Generation of welcome page, send it to the user and wait for user action.

SUBTASKS: None

ERROR COND: If A/TServer == busy, then Process=TimeOut.

5.2 Web pages Research Task

TASK NUMBER: WPRT

TASK NAME: Web Pages Research

PERFORMER: Presidential Election Database Designers

PURPOSE: Research on the rules of US election, and different statics needed for the database. And filter out what might be useful for us.

ENABLING COND: To populate the Presidental Election Query Generator.

DESCRIPTION: Research the internet

FREQUENCY: As often as necessary

DURATION: Varies

IMPORTANCE: Critical

MAXIMUM DELAY: N/A

INPUT: Web queries

OUTPUT: Index of queried results

DOCUMENT USAGE: Web-based search engines

OPS PERFORMED: Researching and bookmarking websites and/or pages with Historical presidential election data

SUBTASKS: None

ERROR COND: None

5.3 Creating tables and Make sure it fits BCNF

TASK NUMBER: CTAMSIF

TASK NAME: Web Pages Research

PERFORMER: Presidential Election Database Designers

PURPOSE: Research on the rules of US election, and different statics needed for the database. And filter out what might be useful for us.

ENABLING COND: To populate the Presidental Election Query Generator.

DESCRIPTION: Research the internet

FREQUENCY: As often as necessary

DURATION: Varies

IMPORTANCE: Critical

MAXIMUM DELAY: N/A

INPUT: Web queries

OUTPUT: Index of queried results

DOCUMENT USAGE: Web-based search engines

OPS PERFORMED: Researching and bookmarking websites and/or pages with Historical presidential election data

SUBTASKS: None

ERROR COND: None

5.4 ETL Task

TASK NUMBER: ETLT

TASK NAME: Extract, Transform, and Load Task

PERFORMER: R script, SQL update query, DataMiner and PHPMyAdmin

PURPOSE: To extract data, transform or reformat it and load it into the database

ENABLING COND: The creation of the database and any addition of data or updates to the database.

DESCRIPTION: This tool (DataMiner) extracts specific data from a web page, and load it into a CSV table.

FREQUENCY: Once for the creation of the database and during any updates.

DURATION: Varies

IMPORTANCE: Critical

MAXIMUM DELAY: N/A

INPUT: A selected web page

OUTPUT: Data into a relation in the database

DOCUMENT USE: HTML documents and

OPS PERFORMED: Data extraction, data transformation, and data loading.

SUBTASKS: Web pages Research

ERROR COND: None

6 The data documents that carry data between tasks

Query for a given Election Year
Election Year President name Main Opponent Vice President Party Affiliation Poll Results

Query for a given President/Candidate
President name Election Year Main Opponent Vice President Party Affiliation
Candidate Name Election Year Candidate Electoral Vote

Query for Re-elected on non-contiguous times

Prof. Nick Roussopoulos

President name
Term1
Term2

Query for Swing Candidates

Election Year1
Election Year2
President/MainOpponent name
Party Affiliation of Year1
Party Affiliation of Year2
Election Result of Year1
Election Result of Year2

Party Historical Query
Party Name
Electoral Vote
Party Vote
Win Counts

Third Party Candidate Info Query

Election
Name
Party
Election Result

Most Popular Candidate Query
Year
Name
Popular Vote Number
Party Affiliation

Lost Poll But Won Election Query
Year
President Name
President Poll Rate
Opponent Name
Opponent Poll Rate

7 ETL Task

A database system refers to a data management warehouse and hence, requires input data in a formatted way. Data used for this President Election Database are extracted from U.S Archives and Records Administration - U.S Electoral College. We would use PED and USEC afterward for simplicity sake.

7.1 The Web Data Extraction Procedure

USEC, as the major data source of the PED, keeps information of the electoral college box score from 1792 to 1996 completely. From a designer's perspective, it is essential to choose a highly effective and repeatable measure in extracting required information. Thus, the Data Miner Addon provides the best tradeoff between customized data extraction and clarity. The backend users can select the bookmarked sites and set up XPATH traits accordingly, which greatly enhance the accuracy of data being filtered. This is accomplished by a few scripts, which are embedded in javascript codes. The addon will then store the data in .csv format which will be used as input to the next procedure. See appendix for more details.

NOTE1: Data extracted from USEC does not include "V.P" and "Vote For Others" Information such that all columns could be uniformly formatted

NOTE2:

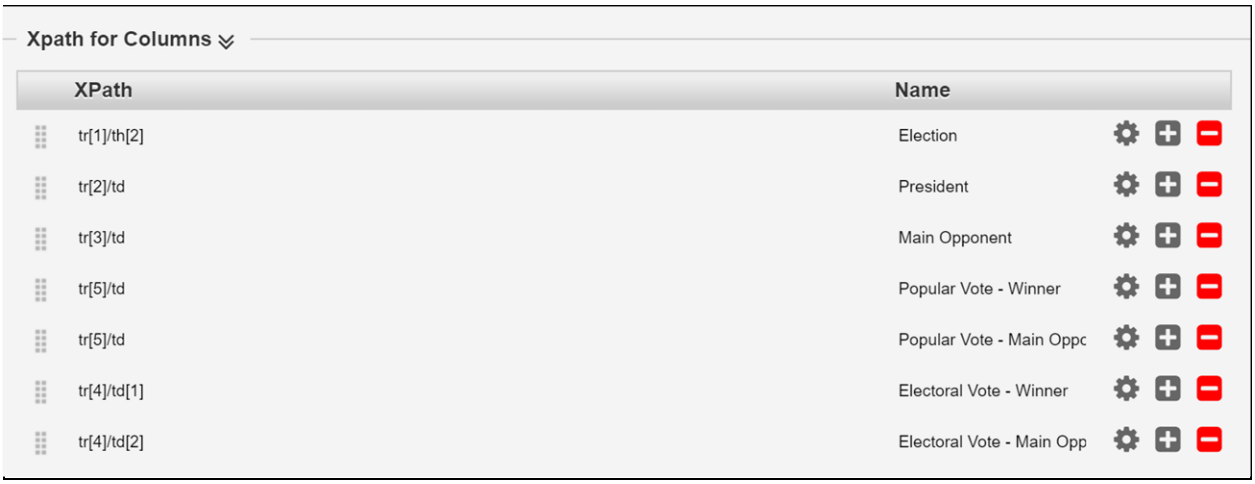


Figure1

```
▼<tr>
    <th>Popular Vote</th>
    <td>Winner:  764,176 </td>
    <td colspan="2">Main Opponent:  550,816 </td> == $0
  </tr>
```

Figure2

Also, notes that XPATH feature for Popular Vote Columns is an incorrect result from ‘colspan=”2”’ attribute. The detailed analysis of the markup structure offers a neat solution, which requires the database designers to change XPATH fields with attributes tr[5]/td as tr[5]/td[1] AND tr[5]/td[2] respectively.

7.2 Localhost Data Wrap up Procedure

Election	President	Main Opponent	Popular Vote - Winner	Popular Vote - Main Opponent	Electoral Vote - Winner	Electoral Vote - Main Opponent
1789	George Washington [F]	John Adams [F]	no record		Winner: Â 69	Main Opponent: Â 34
1792	George Washington [F]	John Adams [F]	no record		Winner: Â 132	Main Opponent: Â 77
1796	John Adams [F]	Thomas Jefferson [D-R]	no record		Winner: Â 71	Main Opponent: Â 68
1800	Thomas Jefferson [D-R]	Aaron Burr [D-R]	no record		Winner: Â 73	Main Opponent: Â 73
1804	Thomas Jefferson [D-R]	Charles C. Pinckney [F]	no record		Winner: Â 162	Main Opponent: Â 14
1808	James Madison [D-R]	Charles C. Pinckney [F]	no record		Winner: Â 122	Main Opponent: Â 47

Figure3

The figure above represents part of the raw data after accomplishing procedure 1. This involves Election by year value, party information... etc. It is worth to note that Party Information is mixed up with candidate name. Hence, further clean up to this dataset is critical. In procedure 2, we applied an R script to this dataset aiming at column splitting and renaming.

Figure4

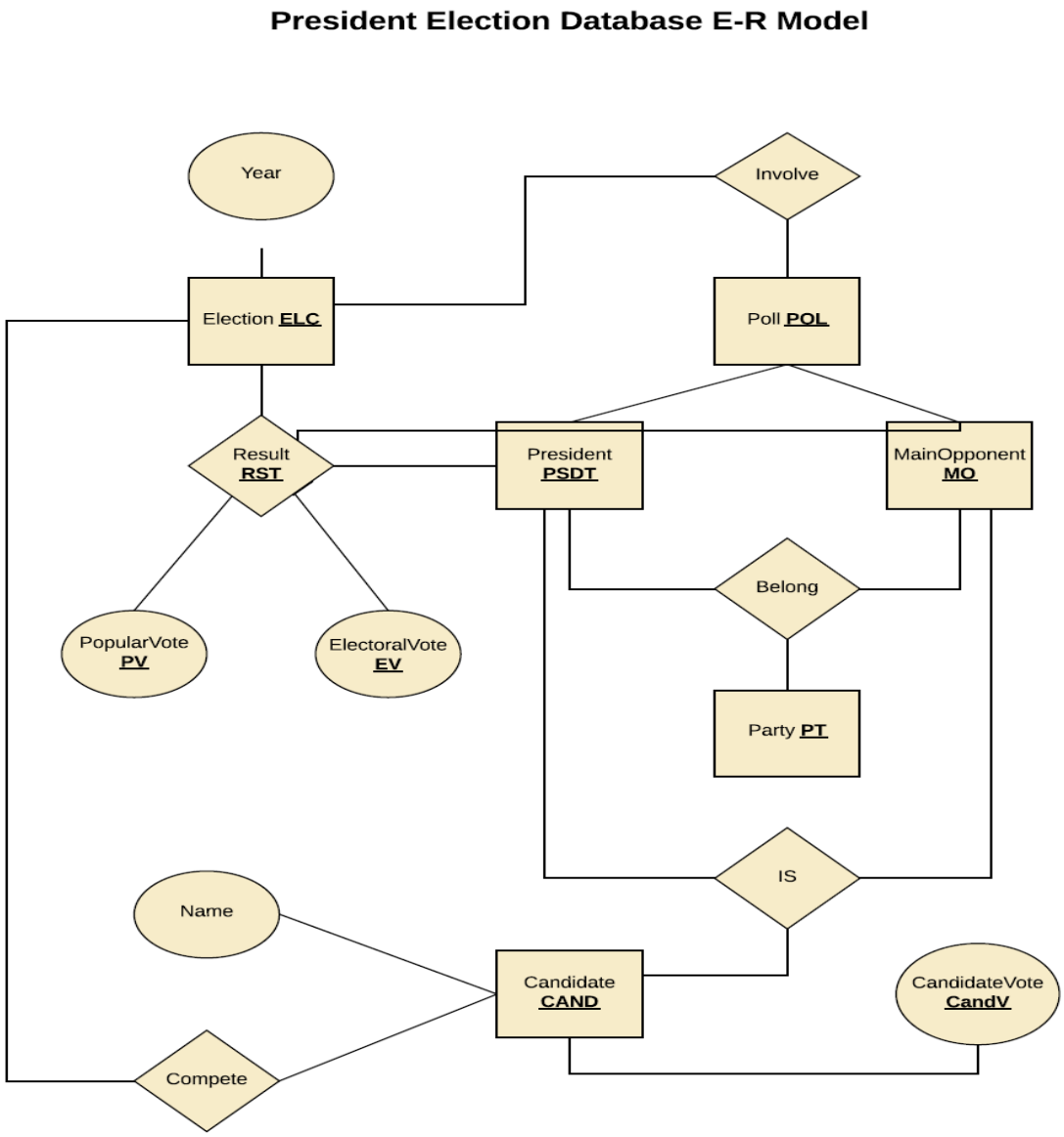
```
1
2  library(stringr)
3  tb=read.csv('C:/Users/Yufan/Downloads/DataMiner.csv')
4  partyPattern = '\\[([^[:digit:]])*\\]'
5  newCol1 = unlist(str_extract_all(tb[2][,],partyPattern))
6  newCol2 = unlist(str_extract_all(tb[3][,],partyPattern))
7
8  tb[2][,] <-gsub(' \\[([^[:digit:]])*\\]','',tb[2][,])
9  tb[3][,] <-gsub(' \\[([^[:digit:]])*\\]','',tb[3][,])
10
11 tb$WinnerParty <- newCol1
12 tb$OpponentParty <- newCol2
13
14 colnames(tb) <- c('Y','P' ,'MO', 'PV_W', 'PV_MO', 'EV_W','EV_MO','WP','OP')
15
16 for (i in 4:7) {
17   tb[i][,] <-gsub('[^[:digit:]]*', '', tb[i][,])
18 }
19
20 write.csv(tb, 'C:/Users/Yufan/Desktop/CMSC424/PE.csv')
```

The selected file will then be processed and matches to the system requirement.

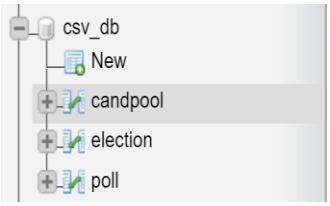
7.3 Load Procedure

A well-formatted CSV file will be used in this step such that the designers can load it to the PED located on the WAMP Server. Users can, therefore, query the relevant data to answer their pre-defined questions through a web interface.

8 ER Model



9 Relation Break down in BCNF Form



The screenshot on the left is the complete relational schema of the PED.

Figure5: CandidatePool Relation

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**Election**	int(4)			Yes	*NULL*			Change Drop Primary Unique Index
2	**CandidateList**	varchar(17)	utf8_general_ci		Yes	*NULL*			Change Drop Primary Unique Index
3	**CandidateVote**	varchar(2)	utf8_general_ci		Yes	*NULL*			Change Drop Primary Unique Index

Figure6: ElectionResult Relation

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**Election**	int(4)			Yes	*NULL*			Change Drop Primary
2	**President**	varchar(22)	utf8_general_ci		Yes	*NULL*			Change Drop Primary
3	**Main Opponent**	varchar(22)	utf8_general_ci		Yes	*NULL*			Change Drop Primary
4	**Electoral Vote -Winner**	int(3)			Yes	*NULL*			Change Drop Primary
5	**Electoral Vote -Opponent**	int(3)			Yes	*NULL*			Change Drop Primary
6	**Popular Vote -Winner**	bigint(8)			Yes	*NULL*			Change Drop Primary
7	**Popular Vote -Opponent**	bigint(8)			Yes	*NULL*			Change Drop Primary
8	**Vice President**	varchar(21)	utf8_general_ci		Yes	*NULL*			Change Drop Primary
9	**WinnerParty**	varchar(9)	utf8_general_ci		Yes	*NULL*			Change Drop Primary
10	**OpponentParty**	varchar(4)	utf8_general_ci		Yes	*NULL*			Change Drop Primary
11	**VicePresidentEV**	varchar(4)	utf8_general_ci		Yes	*NULL*			Change Drop Primary

Figure7: PollResult Relation

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**Election**	int(4)			Yes	*NULL*			Change Drop Primary
2	**cand**	varchar(18)	utf8_general_ci		Yes	*NULL*			Change Drop Primary
3	**pollrate**	int(2)			Yes	*NULL*			Change Drop Primary

Through thorough analysis we decided break down the PED into three pieces. Any relation between these three tables are unrelated to the ones in the other table, and hence, this breakdown remove the redundancy and holds the lossless property.

10 Task Emulation

In this section, we aim to provide the pseudo SQL code that would be available to the users, which would be used later to connect PED with our webserver. Below we list our current available queries and their underlying SQL code respectively.

Query for a given Election Year
Election Year President name Main Opponent Vice President Party Affiliation Poll Results

```
SELECT `Election`,`President`,`MainOpponent`,`VicePresident`,
`WinnerParty`,`OpponentParty`, p1.pollrate, p2.pollrate
FROM `election` e1 join poll p1 using (election) join poll p2 using (election)
WHERE e1.president = p1.cand and e1.`Main Opponent` = p2.cand
LIMIT ".$limit
```

Query for a given President/Candidate
President name Election Year Main Opponent Vice President Party Affiliation
Candidate Name Election Year Candidate Electoral Vote

```
SELECT `Election`,`President`,`Main Opponent`,`VicePresident`,
`WinnerParty`,`OpponentParty`
FROM `election`
WHERE `president` like '%" .$temp."%'
```

Query for Re-elected on non-contiguous times

President name
Term1
Term2

```
SELECT DISTINCT e1.President, e1.Election, e2.Election
FROM election e1, election e2
WHERE e1.President = e2.President and (e1.Election - e2.Election > 4) and
        e1.President like '%".$temp."%'
```

Query for Swing Candidates

Election Year1
Election Year2
President/MainOpponent name
Party Affiliation of Year1
Party Affiliation of Year2
Election Result of Year1
Election Result of Year2

```
SELECT e1.Election, e2.Election, e1.President, e1.WinnerParty, e2.WinnerParty
FROM election e1, election e2
WHERE e1 and e2 are not in the same party AND e1 and e2 have the same name
      AND (e1 is president, e2 is main opponent OR
            e1 is president, e2 is president OR
            e1 is main opponent, e2 is main opponent)
```

Party Historical Query
Party Name
Electoral Vote
Party Vote
Win Counts

```
SELECT WinnerParty as Party, SUM(`Electoral Vote -Winner`) as EV,
            SUM(`Popular Vote -Winner`) as PV, count(WinnerParty)
FROM `election`".$where.
Group by WinnerParty
ORDER BY SUM(`Electoral Vote -Winner`) DESC
```

Third Party Candidate Info Query

Election
Name
Party
Election Result

```
(SELECT election, President as name, WinnerParty as party, TRUE
FROM `election` e1
WHERE e1.WinnerParty != 'D' and e1.WinnerParty != 'R'
) UNION (
SELECT election, President as name, OpponentParty as party, FALSE
FROM `election` e1
WHERE e1.OpponentParty != 'D' and e1.OpponentParty != 'R'
)
```

Most Popular Candidate Query
Year
Name
Popular Vote Number
Party Affiliation

```
SELECT election, President, `Popular Vote -Winner`, WinnerParty
FROM `election`
ORDER BY `Popular Vote -Winner` DESC
LIMIT ".$temp
```

Lost Poll But Won Election Query
Year
President Name
President Poll Rate
Opponent Name
Opponent Poll Rate

```
SELECT p1.Election, e1.President, p1.pollrate, p2.cand, p2.pollrate
        FROM poll p2, election e1 join poll p1 USING (election)
        WHERE e1.President = p1.cand and
        p2.Election = e1.Election and
        p2.cand != e1.President and
        p2.pollrate >= p1.pollrate
```

---

11 Progress Report on Web Server Build up

We used Amazon EC2 instance to set up a remote server such that anyone can access this host from the branch http:/yufangfeng/phpmyadmin.

Mysql and PHPMyAdmin for database management are installed on this server along with the WAMP sever. Currently designers can import the local data files into the server and build up their own queries under the SQL tab. We plan to set up a wrapper for this database server so that users who visit our server can extract information interested in by choosing or entering input without writing MySQL code.

Figure8: Import Tab Screenshot

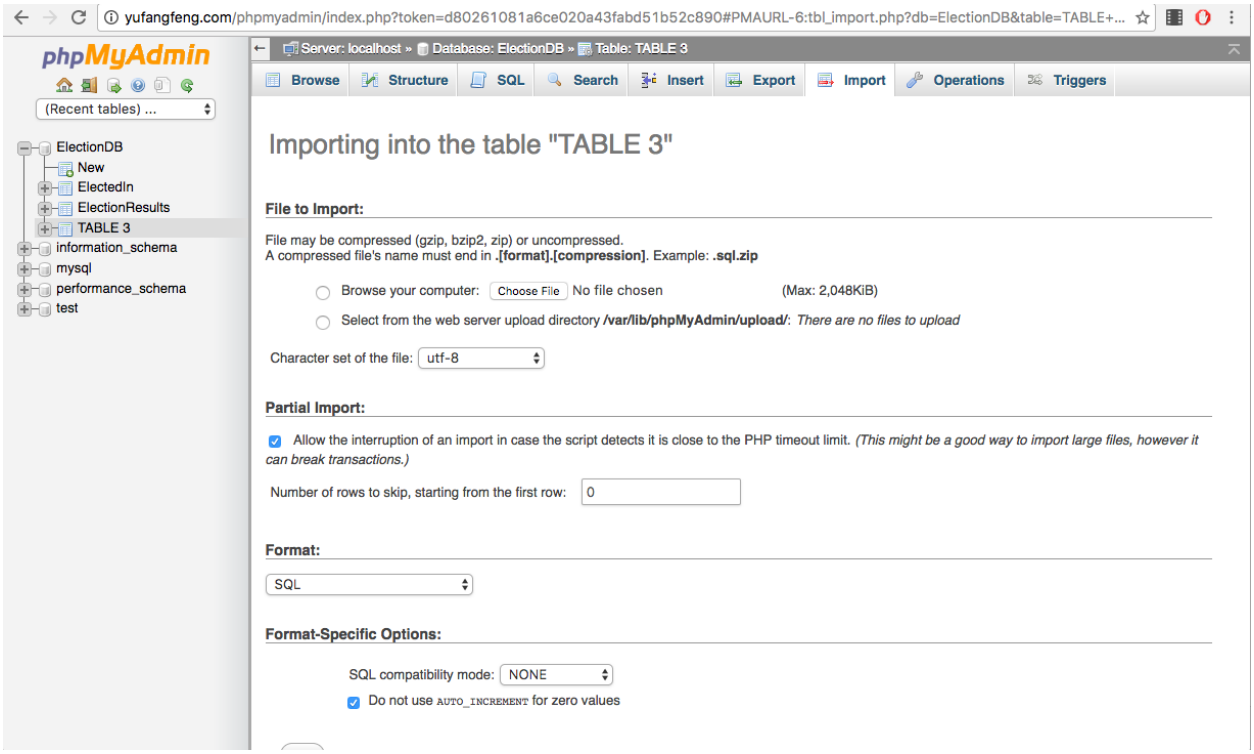
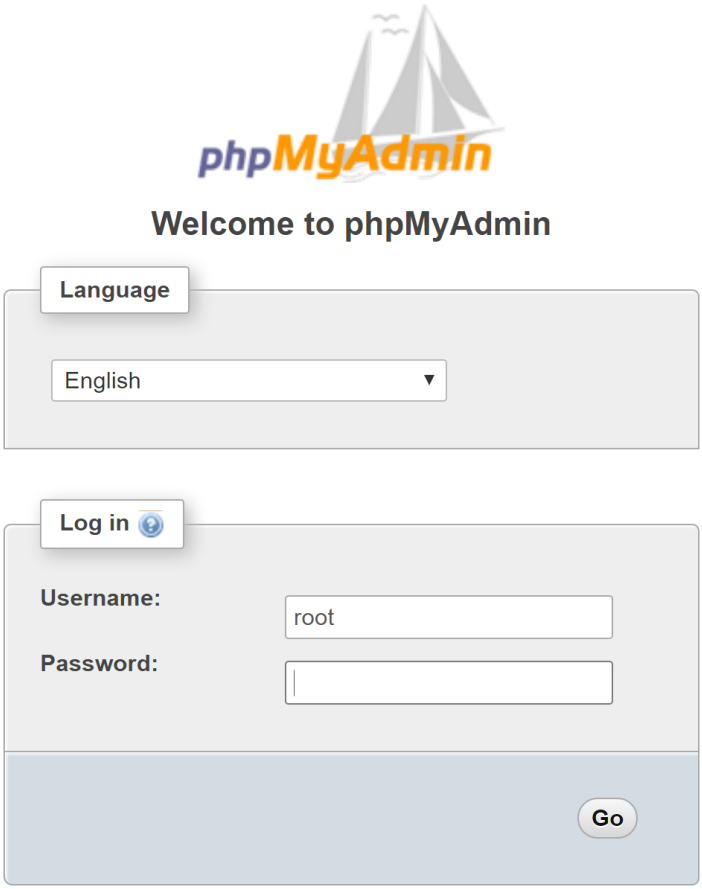


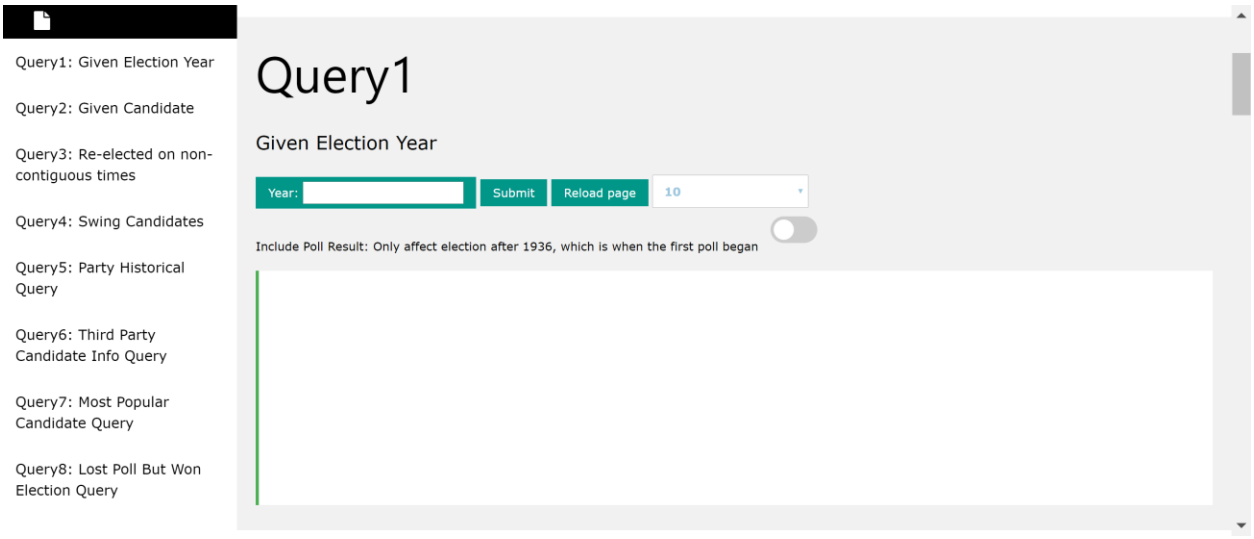
Figure9: PHPmyAdmin Login Page



12 Interactive User Interface

In PhaseIII, I implemented a User Interface build upon PHP, HTML, CSS, and a small portion of JQuery, so that user can search for a limited set of data without modifying the dataset.

Figure10: Screenshot of the user interface



13 User Manual

PAGE LAYOUT:

There is a navigation bar at the left-hand side, users can jump to the location of a certain query by clicking on the html anchors at the side bar.

All queries are listed in one page. There is a title refers to the purpose for each query; a submit button used to activate required search; numerous optional buttons for users to customize their own queries; and a text box below each query used to display corresponding results.

Query1

For Query1, while providing some basic functionalities of this query, I also granted the public users the permission to modify their query in a reasonable and restricted way.

INPUT:

1. The election year to be searched for
2. Dropdown list with value indicating number of maximum results allowed to be shown in the text box
3. A scrolling button to switch between POLL/NO POLL forms. I designed this functionality because of the inconsistency of the dataset. The election relation table contains presidential election data from 1789-2016, while the polling results were not available until 1936. Users can access more detailed results starting from 1936 by choosing the POLL form, or access a less comprehensive dataset ranging from 1789-1936.

OUTPUT:

1. Option info
2. Number of rows returned
3. A table required by the query

Query2

For Query2, I allowed the public users to search for a given candidate/president using his/her partial name.
(e.g 'George' for 'George Bush' and 'George Washington')

INPUT:

1. The candidate/president name to be searched for
2. A scrolling button to switch between PRESIDENT/CANDIDATE forms. I separated all candidates into two parts. The PRESIDENT part represents all major candidates, e.g. President and the Main Opponent of that President. The CANDIDATE part contains limited info about those less competitive candidates.

OUTPUT:

1. Option info
2. Number of rows returned
3. A table required by the query

Query3

For Query2, I allowed the public users to search for a given candidate/president using his/her partial name.
(e.g 'George' for 'George Bush' and 'George Washington')

INPUT:

1. The candidate/president name to be searched for

OUTPUT:

1. Number of rows returned
2. A table required by the query

Query4

INPUT:

NONE

OUTPUT:

1. Number of rows returned
2. A table required by the query

Query5

INPUT:

1. Party Key – A party key legend is provided for users' convenience.

OUTPUT:

1. Number of rows returned
2. A table required by the query

Query6

INPUT:

NONE

OUTPUT:

1. Number of rows returned
2. A table required by the query

Query7

INPUT:

1. A number represents maximum number of results to be listed in the text box

OUTPUT:

3. Number of rows returned
4. A table required by the query

Query8

INPUT:

NONE

OUTPUT:

5. Number of rows returned
6. A table required by the query

Data Insertion

To insert new data to this dataset, users need to have an authorized account provided by the database manager. After being granted, they can sign in to `whateverhostitis/phpmyadmin` to manage the database from a backend perspective.

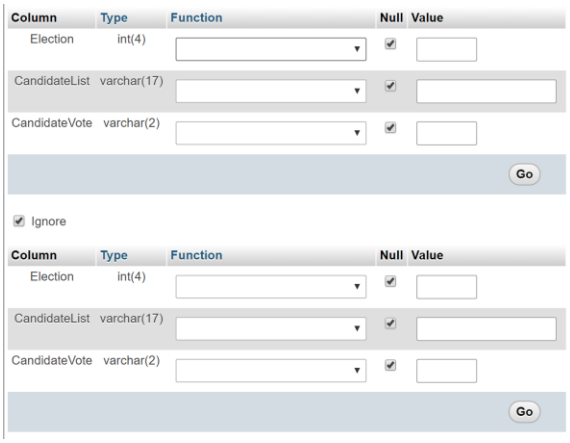


Figure12 on the left shows that users can insert new data to the 'candpool' table.

14 Limitations and possible improvements

Though the PED that I implemented works well in different scenarios, it also has lots of limitations. For example, the data source merely comes from archive.gov and Wikipedia, while the formats of these sites are not in consistent with each other.

Case1: Poll result in Wikipedia includes the party affiliations of some of the candidates, however, it also omits the party affiliations of some candidates. Therefore, I have to delete the PA information from the Poll result dataset.

Case2: The electoral college provides very comprehensive data about historical U.S presidential election result. However, the information of some less competitive candidates is incomplete. Therefore, I have to separate the dataset into two pieces – one for strong candidates with complete information and one for weak candidates with merely electoral votes.

The PED would be more comprehensive and could provide more insights to the users if the data source could be kept in a more consistent and complete manner.