Student Number: 15334636
Course: CS3041 – Information Management II
Topic: Elect_Gov Database Report
Date: 27-11-2018

Application Description

The database I have chosen to model is a database for political science. The idea is to build a system that contains information on political parties, elections and cabinets for most democracies that are part of the EU (European Union) or the OECD (Organization for Economic Co-operation and Development) and provide extensive analysis to the datasets. However, for this assignment, the scope will be narrowed down to only just one country, i.e. France, to provide a preview of the basis of the intended system and how it can be adjusted to store and unify political science data for every democratic country. So, for the data structure, we need Party, Elections, Election Results, Cabinet, Cabinet Party, President, Party Family and Party Position.

Every party consists of a unique Id stored in database, a party name along with its associated abbreviation, dates for its foundation and dissolve and further descriptions which is optional. A party that has changed its name or merged into bigger organizations is deemed as dissolving current party and establishing a new one in the context of the system. In other words, we create another instance of entries for the table. Every party must have a date it is founded upon but doesn't necessarily need to have a date for its dissolve, as that means that particular party still persists.

A party can also be classified into families based on the ideologies and aspirations it is founded upon. A party can have multiple ideologies or mixed ones as well, such as conservative liberalism and social liberalism, We can then use these ideologies to determine or deduce a marking for their position in the parliament, scaling from 0.0 to 10.0, namely left to center to right. The closer the marks to the boundaries, i.e. 0 and 10, the deeper the party beliefs on that thinking.

Next, we move on to the Election, which also has its own unique ID. The schema can record elections of two kinds: A parliamentary election is an election within a country to choose a national government, whereas a presidential election is an election to choose the head of state. We can add more elections to the system such European parliamentary election. Every election must have a date, total seats contested, electorate (or registered voters), number of vote casts and valid votes and a previous election Id. For presidential elections, total seats contested is 1, which is the position. One election can only elect 1 president or one cabinet.

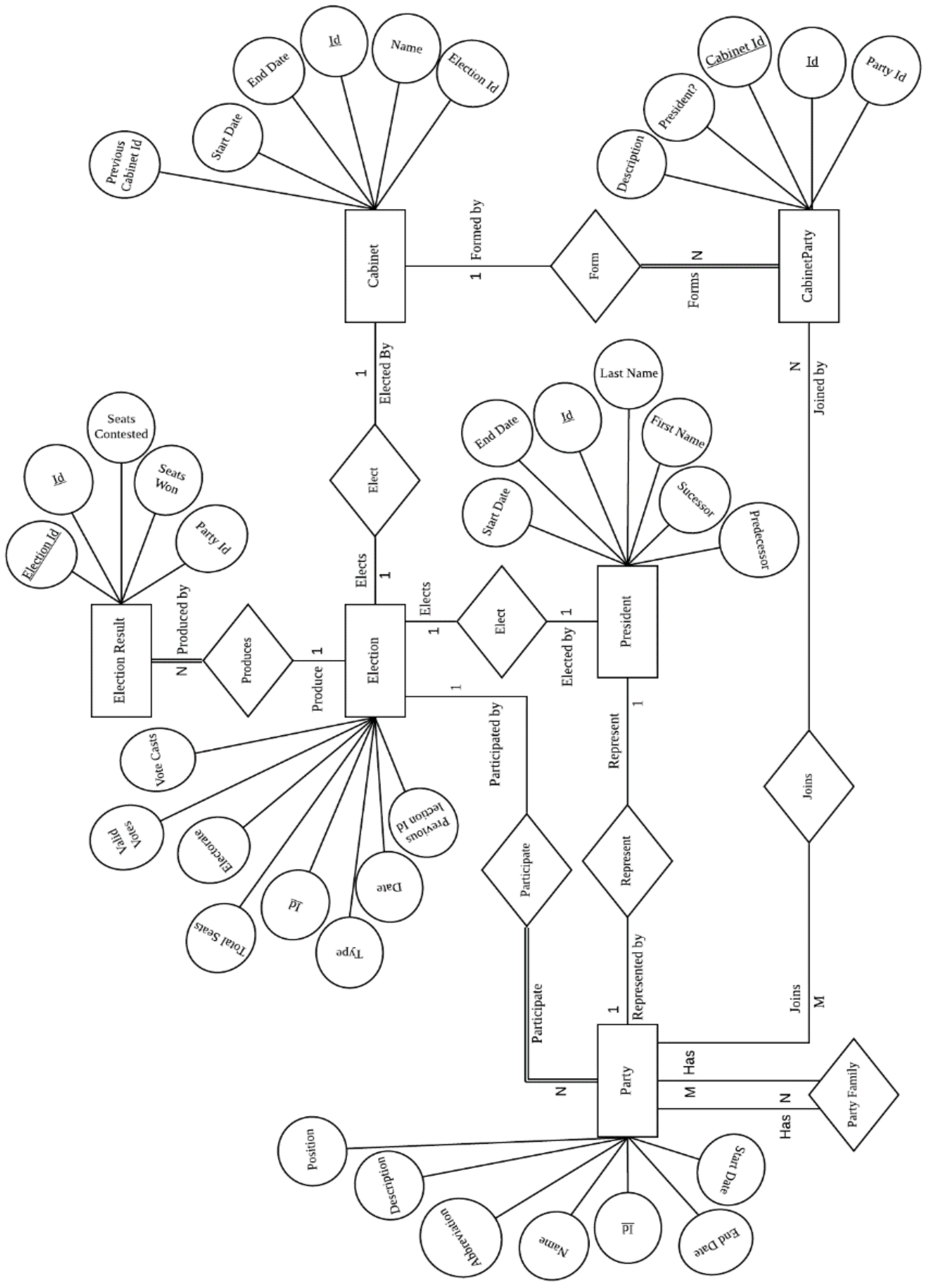
If there is an election, then there must be results for it. For Election Result, we have to take into account that there might be a lot of parties taking part in one election. To better identify each result entries we can use the election id and tag it with index for every entries, i.e. create a composite primary key. Every row will then consist of an Election Id, Id, Party Id, number of seats contested and won in that election.

So after the election, we are concerned with cabinets formed. One thing to take note is that a cabinet can only be formed by one election and also that the date it is formed can never be earlier than it's respective election. A cabinet can then contain its own unique Id, a name which consists of names of the Prime Minister or any personnel elected to oversee the new government, a term start and end date and a reference to the last cabinet.

A cabinet formed normally comprises of party or even parties like a coalition or alliance. Every cabinet party must be a valid party in the system and the entries for this section can be identified by using Cabinet Id and Id associated for it like Election Results. Then it also records if that party fills the position of the president and further descriptions.

Lastly, a president elected must have his own unique Id, typically his placement in the long list of presidents of the nation. The entries for this table comes along with his last and first name, term start and end date, his affiliated party, and his predecessor and successor. A president who is still serving will have no successor. A president can only represent one party, his own.

Entity Relational Diagram



Mapping to Relational Schema

In the diagram, to normalize the diagram, family is removed from the main party table, as it contains multiple instances, i.e. a party can have many party families or ideologies. Hence, a new table, party family is built with party Id and family, where partyId and family are composite primary keys. There's a 2 way relationship between Party Family and Party, as a party family can have many parties in it, while party can also be in multiple families. Secondly. I join Election Id and Id in Election Results as a composite primary key in Election Results to determine each party results in different elections. Thirdly, I combined cabinet Id and Id in Cabinet Party to form another composite primary key to index the parties in different cabinets formed. As for Party Position, I want to simulate the model in real world situation where positions will keep changing based on the beliefs and aspirations of a party. So, in order to cut down access to other tables, we can limit this frequent changes to just one, thus preventing data loss due to mistakes while constantly updating. Now we map it to our relational schema and table before constructing Functional Dependency Diagram.

Party(Id, Name, Abbreviations, Start_Date, End_Date, Description)

Election(Id, Type, Date, Total Seats, Electorate, Vote Cast, Valid Votes, Previous Election Id)

Election Result(**Election Id**, Id, **Party Id**, Seats Contested, Seats Won)

Cabinet(Id, Name, Start_Date, End_Date, **Election Id**, Previous Cabinet Id)

Cabinet Party(**Cabinet Id**, Id, Party Id, President?)

President(Id, Last Name, First Name, Start_Date, End_Date, **Party Id**, Predecessor, Successor)

Party Position(**Party Id**, Position)

Party Family(**Party Id**, Family)

--THIS is foreign key, ps I can't do double underline.

Creating the tables along with the implicit constraints:

CREATE TABLE party(Id INT UNSIGNED NOT NULL, Name VARCHAR(100) NOT NULL, Abbreviation VARCHAR(10) NOT NULL, Start_Date DATE NOT NULL, End_Date DATE NULL, Description VARCHAR(1000), PRIMARY KEY(Id));

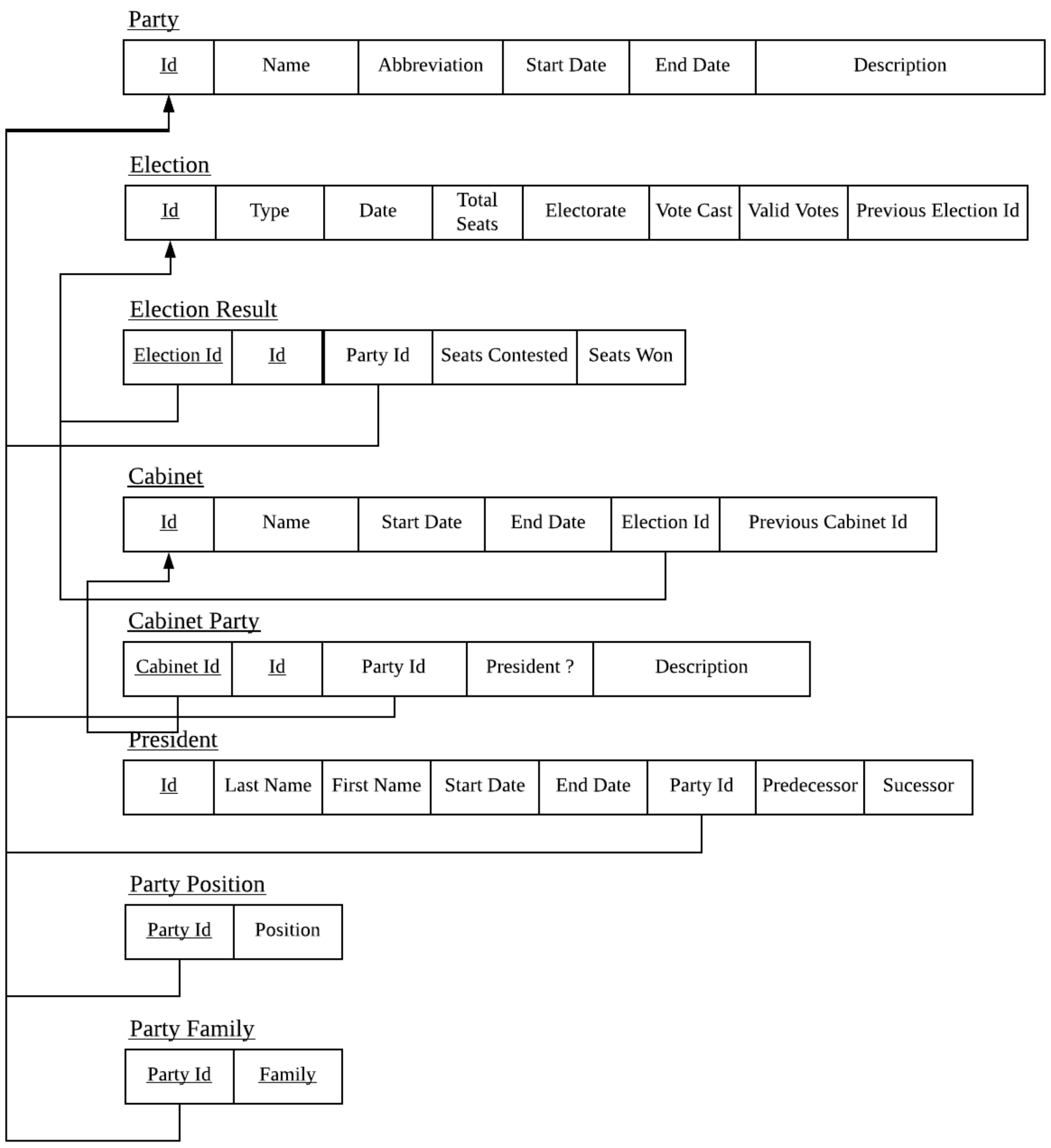
CREATE TABLE Cabinet(Id INT UNSIGNED NOT NULL, Name VARCHAR(100) NOT NULL, Start_Date DATE NOT NULL, End_Date DATE NULL, Election_Id INT UNSIGNED NOT NULL UNIQUE, Previous_Cabinet_Id INT UNSIGNED NOT NULL UNIQUE REFERENCES Cabinet(Id), PRIMARY KEY(Id));

CREATE TABLE Cabinet_Party(Cabinet_Id INT UNSIGNED NOT NULL, Id INT UNSIGNED NOT NULL, Party_Id INT UNSIGNED NOT NULL, President BOOLEAN NOT NULL, DESCRIPTION VARCHAR(1000) NULL, PRIMARY KEY(Cabinet_Id,Id));

```
CREATE TABLE Election(Id INT UNSIGNED NOT NULL, Type ENUM ('European Parliament',
'Parliamentary election', 'Presidential Election'), Date DATE NOT NULL, Total_Seats INT
UNSIGNED NOT NULL, Electorate INT UNSIGNED NOT NULL, Votes_Cast INT UNSIGNED
NOT NULL, Valid_Votes INT UNSIGNED NOT NULL, Previous_Election_Id INT UNSIGNED
NOT NULL UNIQUE REFERENCES Election(Id), PRIMARY KEY(Id));

CREATE TABLE Election_Result(Election_Id INT UNSIGNED NOT NULL, Id INT
UNSIGNED NOT NULL, Party_Id INT UNSIGNED NOT NULL, Seats_Contested INT
UNSIGNED NOT NULL, Seats_Won INT UNSIGNED NOT NULL, PRIMARY
KEY(Election_Id, Id));

CREATE TABLE President(Id INT UNSIGNED NOT NULL, Last_Name VARCHAR(50) NOT
NULL, First_Name VARCHAR(50) NOT NULL, Start_Date DATE NOT NULL, End_Date
DATE NULL, Party_Id INT UNSIGNED NOT NULL, Predecessor INT UNSIGNED NOT NULL
REFERENCES President(Id), Successor INT UNSIGNED NULL REFERENCES President(Id),
PRIMARY KEY(Id));

CREATE TABLE Party_Family(Party_Id INT UNSIGNED NOT NULL REFERENCES party(Id),
Family VARCHAR(50) NOT NULL, PRIMARY KEY(Party_Id, Family));

CREATE TABLE Party_Position(Party_Id INT UNSIGNED NOT NULL REFERENCES
party(Id), Position REAL UNSIGNED NOT NULL, PRIMARY KEY(Party_Id));

ALTER TABLE party ALTER Description SET DEFAULT "";
ALTER TABLE party ADD CHECK(End_Date>Start_Date AND End_Date IS NOT NULL);
ALTER TABLE Cabinet ADD CONSTRAINT cabinet_election FOREIGN KEY (Election_Id)
REFERENCES Election(Id), ADD CHECK(End_Date>Start_Date);
ALTER TABLE Cabinet_Party ADD FOREIGN KEY (Party_Id) REFERENCES Party(Id), ADD
FOREIGN KEY (Cabinet_Id) REFERENCES Cabinet(Id);

ALTER TABLE Election ADD CHECK(Electorate>=Votes_Cast), ADD
CHECK(Votes_Cast>=Valid_Votes), ADD CHECK(Date <= CURDATE());
ALTER TABLE Election_Result ADD FOREIGN KEY (Election_Id) REFERENCES
Election(Id), ADD FOREIGN KEY (Party_Id) REFERENCES Party(Id), ADD
CHECK(Seats_Contested >= Seats_Won);
ALTER TABLE President ADD FOREIGN KEY (Party_Id) REFERENCES Party(Id), ADD
CHECK(End_Date>Start_Date AND End_Date IS NOT NULL);
ALTER TABLE Party_Position ADD CHECK(Position >= 0.0 AND Position <= 10.0);
```

Functional Dependency Diagram

Functional Dependancy Diagram



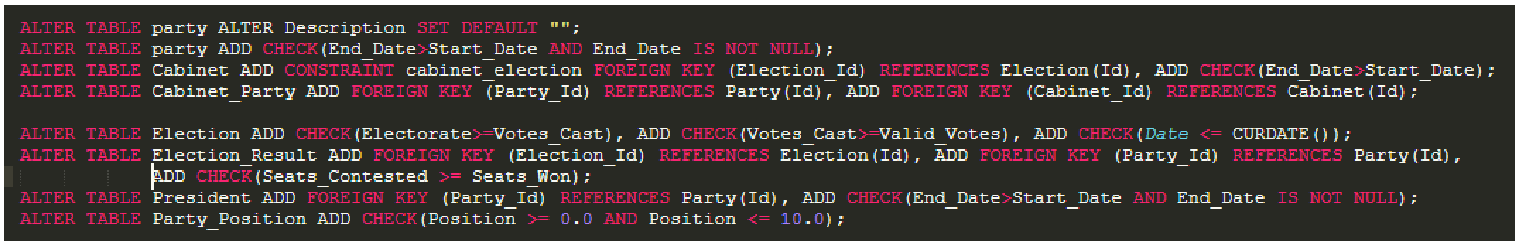
Semantic Constraints

1. The start date of every entry party can never be later than its end date. It is possible for parties to have null end dates, as it means that they are still functional, however start date can never be null as it would mean that the party didn’t exist.
2. In an election, the relationship between number of electorates, vote cast and valid votes will always be as follows:

$$\text{electorates} \geq \text{vote cast} \geq \text{valid votes}$$

3. In an election result, seats contested will always be greater or equal to seats won.
4. Same case as in Party, start date of cabinet can never be later than its end date. It is possible for cabinets to have null end dates, as it means that they are still functional, however start date can never be null as it would mean that the cabinet didn’t exist. All cabinets can only be formed by one election, therefore election Id has to be unique.
5. For President, his term start date can never be later than its end date. End date can be null, as that means he is still serving. A president must have a predecessor but doesn’t necessarily require a successor while he’s still serving.
6. Party position is ranked on a scale that ranges from 0.0 to 10.0, i.e. far left to centre to far right.
7. Cabinet can never be created before its respective election. In other words, election data will always be inserted before cabinet, to ensure that cabinet can reference its election id. We can solve this problem by implementing a trigger.

These constraints can be dealt with using checks and preliminary declarations when creating the tables.



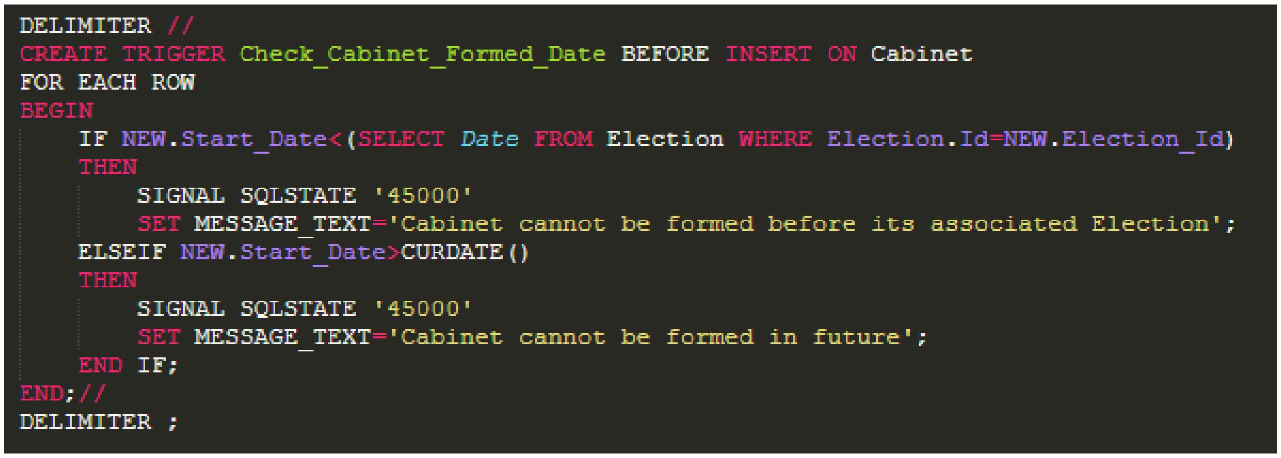
```sql
ALTER TABLE party ALTER Description SET DEFAULT "";
ALTER TABLE party ADD CHECK(End_Date>Start_Date AND End_Date IS NOT NULL);
ALTER TABLE Cabinet ADD CONSTRAINT cabinet_election FOREIGN KEY (Election_Id) REFERENCES Election(Id), ADD CHECK(End_Date>Start_Date);
ALTER TABLE Cabinet_Party ADD FOREIGN KEY (Party_Id) REFERENCES Party(Id), ADD FOREIGN KEY (Cabinet_Id) REFERENCES Cabinet(Id);

ALTER TABLE Election ADD CHECK(Electorate>=Votes_Cast), ADD CHECK(Votes_Cast>=Valid_Votes), ADD CHECK(Date <= CURDATE());
ALTER TABLE Election_Result ADD FOREIGN KEY (Election_Id) REFERENCES Election(Id), ADD FOREIGN KEY (Party_Id) REFERENCES Party(Id),
            ADD CHECK(Seats_Contested >= Seats_Won);
ALTER TABLE President ADD FOREIGN KEY (Party_Id) REFERENCES Party(Id), ADD CHECK(End_Date>Start_Date AND End_Date IS NOT NULL);
ALTER TABLE Party_Position ADD CHECK(Position >= 0.0 AND Position <= 10.0);
```

<u>**Triggers**</u>

As mentioned in the semantic constraint section that cabinet can never be created before its respective election. To prevent this, we create a trigger that fires every time before a new cabinet entry is inserted into the table. This trigger compares start date of the new entry with its respective election date, and will decide if the entry can be validated. There are two possibilities that could cause errors in these scenario.

1. Cabinet is in fact created before its associated election.
2. Cabinet is created after election, but the date is after current date. In other words, its start date is in the future, which is not possible or shouldn't be practiced in database management.

The 2 possible scenarios can be averted by doing 2 if statements in our trigger as follows:



```
DELIMITER //
CREATE TRIGGER Check_Cabinet_Formed_Date BEFORE INSERT ON Cabinet
FOR EACH ROW
BEGIN
    IF NEW.Start_Date<(SELECT Date FROM Election WHERE Election.Id=NEW.Election_Id)
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT='Cabinet cannot be formed before its associated Election';
    ELSEIF NEW.Start_Date>CURDATE()
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT='Cabinet cannot be formed in future';
    END IF;
END;//
DELIMITER ;
```

An alternative problem that might raise the attention of using a trigger is when a new president or cabinet is inserted, we want to end the president's or cabinet's term and set the end date from null to current date using CURDATE(). However, this is not a feasible solution as elections are always held between a president stepping down and a new president elected or a cabinet dissolving and a new cabinet is formed. It is difficult for triggers to decide when to perform these operations As a result, it is still more efficient to resort to manually updating the end date in both president and cabinet table. Same situation applies to party.

Database Security:

Before proceeding into updates, it crucial to implement limited access to political science database users. In real world scenario, entries shouldn't be destroyed or deleted to be inserted again once they are in the system, unless they are inserted in the beginning. In fact, changes to existing data entries should be small with its frequency set to the bare minimum. Moreover, any entries operations should only be executed by authorized personnel and normal users can only be granted READ-ONLY access to maintain the integrity and reliability of the data.

We can do this by using GRANT command. For an authorized personnel, we can do

```
 GRANT ALL PRIVILEGES ON database.*TO 'user'@'localhost';
```

where user is the username of the personnel and localhost is the database server.

As for a normal user, we can perform the following:

```
GRANT SELECT ON database.* TO 'user'@'localhost';
```

For further data protection, we can implement triggers that fire whenever any data modification operations is executed, to record the personnel responsible and time for the operation.

Updates:

As mentioned above, some data entries are meant to be alter once after their insertion, e.g. End Date of Party, Cabinet and President.

To update the corresponding values we can perform the following simple command:

UPDATE party SET end_date = '2010-09-08' WHERE Id = 12345;

Or

UPDATE president SET end_date = '2010-09-08' WHERE Id = 32173;

We specify the conditions to be the id of the target entity rather than its End Date due to the further improvements to convert the system to accommodate data from other countries as well. For instance in the case of president, if we do WHERE End_Date = NULL, then every current serving president will have to resign, causing unwanted, detrimental errors to the whole column. As Id is unique due to its primary key nature, we can perform the operations properly as long as we know our target.

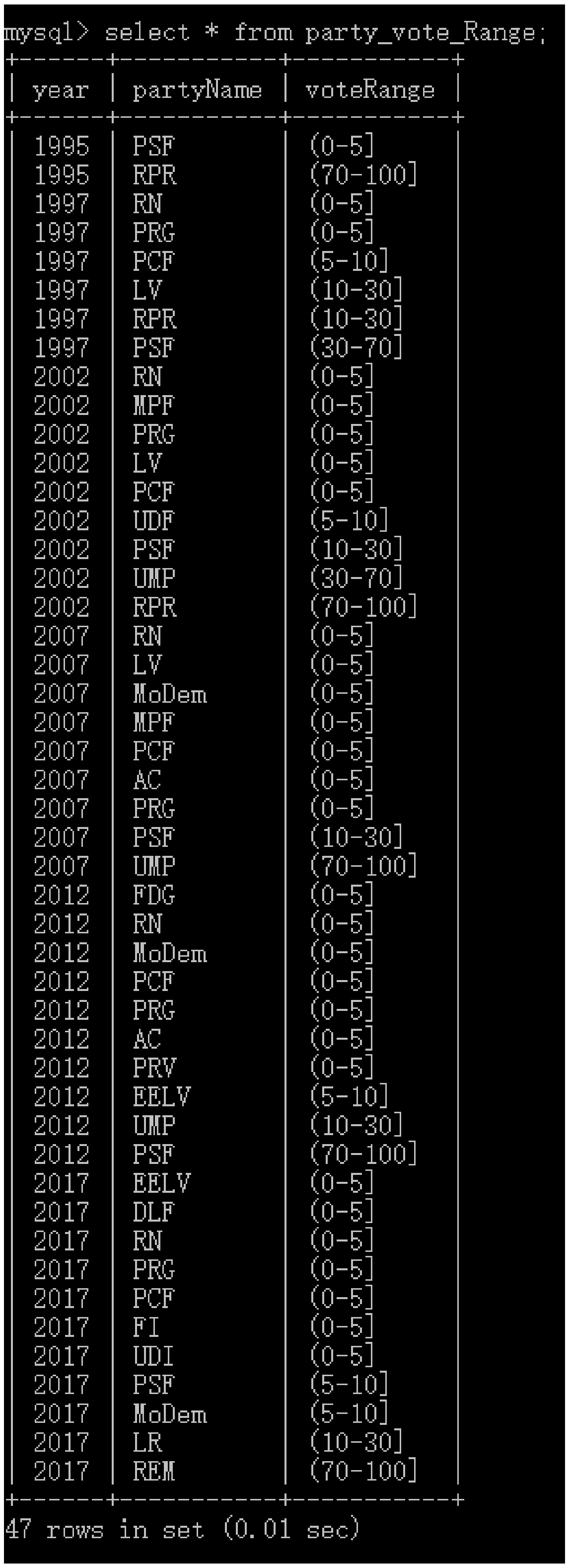
As preliminary checks and precautionary warning are being integrated into the system while implementing the constraints and triggers, wrong data entries should be prevented from being inserted into respective table.

VIEWs using SELECT, ORDER BY, GROUP BY, JOINs, CASE:

VOTE RANGE: Suppose if we want to report the name of the party and its performance in elections, we can use a description of the range into which the number of valid votes it received falls, in the following format: (lb-ub], where lb is the lower bound of the range and ub is the upper bound of the range (for example, (20-30].) These are the range values to consider: zero and below 5 percent of valid votes inclusive, 5 to 10 percent of valid votes inclusive, 10 to 30 percent of valid votes inclusive, 30 to 70 percent of valid votes inclusive, and above 70 percent of valid votes. The range values are defined only for the parties and elections for which the number of votes are recorded. Note, we don't perform it on presidential elections, as there are only 2 possible values 0 or 1, which is not interesting.

```
--create a view for the percentage of votes each party received in all elections by dividing seats_won by total_seats and multiplying it by 100.

create view party_percentage_votes as(
        select t.partyName, ceil(avg(t.vote_percentage)) as vote_percentage, t.year
        from (
                select YEAR(e.date) as year, p.Abbreviation as partyName,
Format(((er.seats_won / e.Total_Seats)*100),2) as vote_percentage
                from election e join election_result er on e.id = er.election_id join party p on p.id
= er.party_id
                where e.Type != 'Presidential_Election'
  )t
  where t.vote_percentage is not null
  group by t.year, t.partyName
);

--Then, classify them using CASE statements

create view party_vote_range as(
        select year, partyName,
        case    when vote_percentage >= 0 and vote_percentage <= 5 then '(0-5]'
                        when vote_percentage > 5 and vote_percentage <= 10 then '(5-10]'
                        when vote_percentage > 10  and vote_percentage <= 30   then '(10-30]'
                        when vote_percentage > 30   and vote_percentage <= 70   then '(30-70]'
                        when vote_percentage > 70 and vote_percentage <= 100 then '(70-100]'
        end as voteRange
        from party_percentage_votes
        order by year, vote_percentage
);
```

Result:

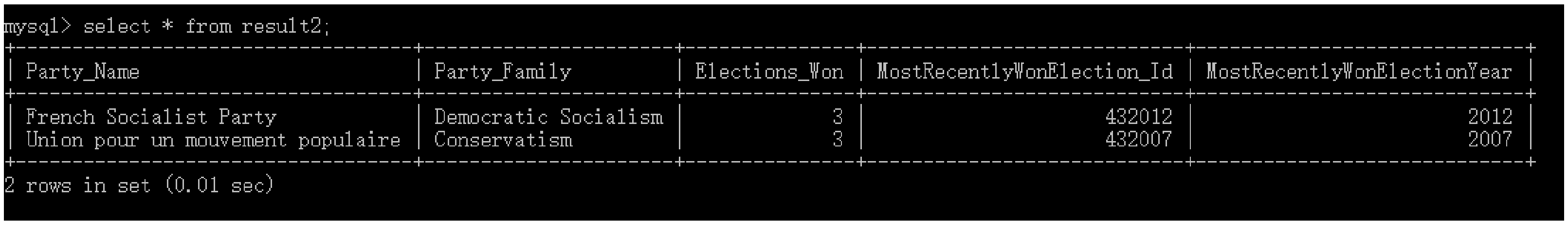


```
mysql> select * from party_vote_Range;
+------+-----------+-----------+
| year | partyName | voteRange |
+------+-----------+-----------+
| 1995 | PSF       | (0-5]     |
| 1995 | RPR       | (70-100]  |
| 1997 | RN        | (0-5]     |
| 1997 | PRG       | (0-5]     |
| 1997 | PCF       | (5-10]    |
| 1997 | LV        | (10-30]   |
| 1997 | RPR       | (10-30]   |
| 1997 | PSF       | (30-70]   |
| 2002 | RN        | (0-5]     |
| 2002 | MPF       | (0-5]     |
| 2002 | PRG       | (0-5]     |
| 2002 | LV        | (0-5]     |
| 2002 | PCF       | (0-5]     |
| 2002 | UDF       | (5-10]    |
| 2002 | PSF       | (10-30]   |
| 2002 | UMP       | (30-70]   |
| 2002 | RPR       | (70-100]  |
| 2007 | RN        | (0-5]     |
| 2007 | LV        | (0-5]     |
| 2007 | MoDem     | (0-5]     |
| 2007 | MPF       | (0-5]     |
| 2007 | PCF       | (0-5]     |
| 2007 | AC        | (0-5]     |
| 2007 | PRG       | (0-5]     |
| 2007 | PSF       | (10-30]   |
| 2007 | UMP       | (70-100]  |
| 2012 | FDG       | (0-5]     |
| 2012 | RN        | (0-5]     |
| 2012 | MoDem     | (0-5]     |
| 2012 | PCF       | (0-5]     |
| 2012 | PRG       | (0-5]     |
| 2012 | AC        | (0-5]     |
| 2012 | PRV       | (0-5]     |
| 2012 | EELV      | (5-10]    |
| 2012 | UMP       | (10-30]   |
| 2012 | PSF       | (70-100]  |
| 2017 | EELV      | (0-5]     |
| 2017 | DLF       | (0-5]     |
| 2017 | RN        | (0-5]     |
| 2017 | PRG       | (0-5]     |
| 2017 | PCF       | (0-5]     |
| 2017 | FI        | (0-5]     |
| 2017 | UDI       | (0-5]     |
| 2017 | PSF       | (5-10]    |
| 2017 | MoDem     | (5-10]    |
| 2017 | LR        | (10-30]   |
| 2017 | REM       | (70-100]  |
+------+-----------+-----------+
47 rows in set (0.01 sec)
```

FIND WIINERS: Let’s say now we want to find parties that managed to win more than 2 times:

```
--organize all the results in elections
create view partyVotes as
(
        select e.id as election_id, e.date, p.id as party_id, p.name as party_name, er.seats_won
        from election_result er, election e,  party p
        where
                er.election_id = e.id AND
                er.party_id = p.id
);


--find the party with the majority vote
create view partyWins as
(
   select pv.election_id, pv.date, pv.party_id, pv.party_name, pv.seats_won
   from partyVotes pv,
      (
      select election_id, max(seats_won) as max_seats
      from partyVotes
      group by election_id
      ) pv2
   where pv.election_id = pv2.election_id and pv.seats_won = pv2.max_seats
);

--to count average wins of each party by counting their election_id
create view partyWinAvg as
(
        select pw.party_id, max(date) as max_date, election_id, count(distinct pw.election_id) as
num_won_elections
        from partyWins pw
        group by party_id
        order by num_won_elections
);

--now just compare it so its >= 3
create view moreThan2Avg as
(
        select * from
        (
           (
           select party_id, family as party_family
           from party_family pf
           ) ssq1
           natural right join
           (
            select pwa.party_id, max_date as recent_victory, num_won_elections
            from partyWinAvg pwa
            where
            pwa.num_won_elections >= 3
            ) ssq2
        )
);

--merge the more than 3 times winners with its corresponding families using join
create view result2 as
(
select p.name as Party_Name, party_family as Party_Family, num_won_elections as
Elections_Won,
er.election_id as MostRecentlyWonElection_Id, extract(YEAR from recent_victory) as
MostRecentlyWonElectionYear
from moreThan2Avg mt3a, party p, election e, election_result er
where
   mt3a.recent_victory = e.date              and
   mt3a.party_id = p.id                      and
   p.id = er.party_id                        and
   e.id = er.election_id
order by Party_Name asc
);
```

Result:

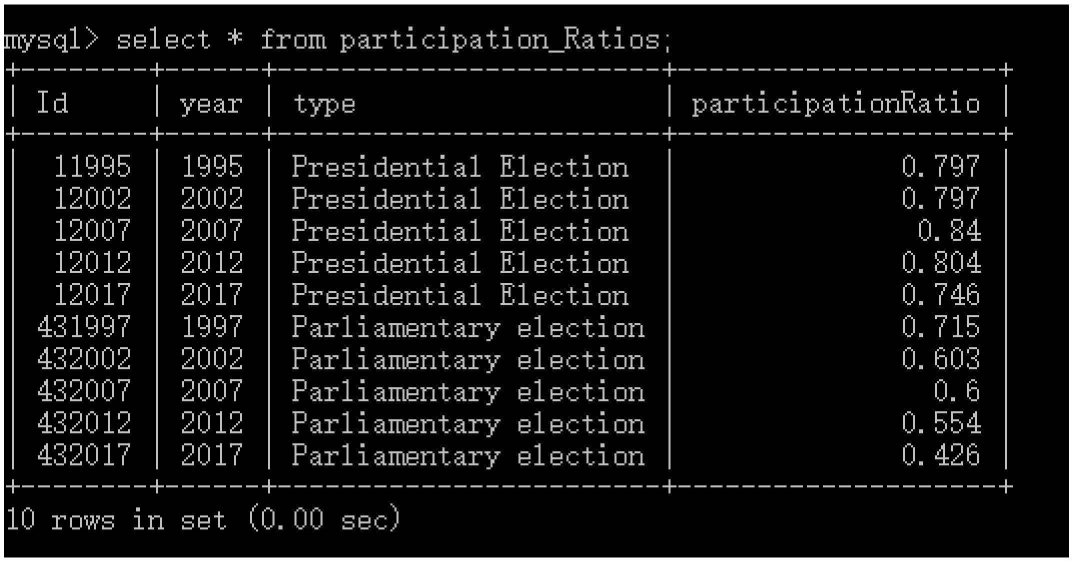


```
mysql> select * from result2;
+------------------------------------+----------------------+---------------+----------------------------+-----------------------------+
| Party_Name                         | Party_Family         | Elections_Won | MostRecentlyWonElection_Id | MostRecentlyWonElectionYear |
+------------------------------------+----------------------+---------------+----------------------------+-----------------------------+
| French Socialist Party             | Democratic Socialism |             3 |                     432012 |                        2012 |
| Union pour un mouvement populaire  | Conservatism         |             3 |                     432007 |                        2007 |
+------------------------------------+----------------------+---------------+----------------------------+-----------------------------+
2 rows in set (0.01 sec)
```

Do citizens participate more? The number of eligible voters/electorate, vote cast, valid votes have been recorded for each election. Analysts would like to know if citizens of countries participate enthusiastically in elections. The participation ratio of an election is the ratio of votes cast to the number of citizens who are eligible to vote, i.e. the electorates. Note that the participation ratio can only be ranging between zero and one. Let's compute the average participation ratio for each elections.

```
create view participation_ratios as(
        select t.Id, year, type, avg(t.p_ratio) as participationRatio
        from(
                select Id, YEAR(e.date) as year, type, FORMAT((e.votes_cast / e.electorate),3)
                as p_ratio
                from election e
        )t
        group by t.Id
);
```

Result:

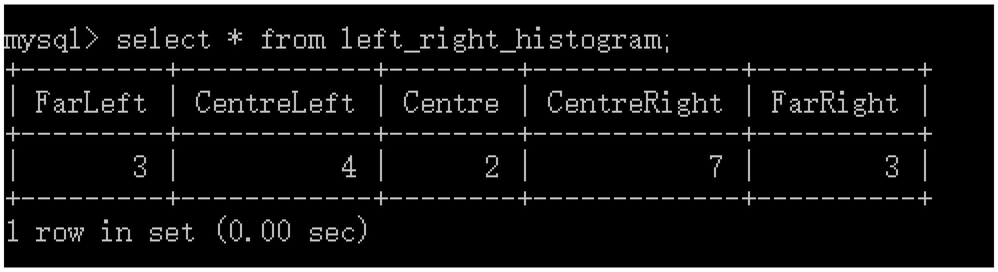


```
mysql> select * from participation_Ratios;
+--------+------+------------------------+--------------------+
| Id     | year | type                   | participationRatio |
+--------+------+------------------------+--------------------+
|  11995 | 1995 | Presidential Election  |              0.797 |
|  12002 | 2002 | Presidential Election  |              0.797 |
|  12007 | 2007 | Presidential Election  |               0.84 |
|  12012 | 2012 | Presidential Election  |              0.804 |
|  12017 | 2017 | Presidential Election  |              0.746 |
| 431997 | 1997 | Parliamentary election |              0.715 |
| 432002 | 2002 | Parliamentary election |              0.603 |
| 432007 | 2007 | Parliamentary election |                0.6 |
| 432012 | 2012 | Parliamentary election |              0.554 |
| 432017 | 2017 | Parliamentary election |              0.426 |
+--------+------+------------------------+--------------------+
10 rows in set (0.00 sec)
```

LEFT-RIGHT Histogram: The database records several policy positions of political parties, including their "left-right dimension". Suppose the left-right range is divided into 5 interval ([0,2.5), [2.5,5), [5), [5,7.5) and [7.5,10]). Create a table that is, essentially, a histogram of parties and their respective class based on left-right position in

```
--view the histogram as columns where every range is an attribute. Extract all the columns separately.
create view far_left as(
        select count(pp.position) as FarLeft
        from party_position pp join party p on pp.party_id = p.id
        where pp.position >= 0 AND pp.position < 2.5
);
create view center_left as(
        select count(pp.position) as CentreLeft
        from party_position pp join party p on pp.party_id = p.id
        where pp.position >= 2.5 AND pp.position < 5
);
create view center as(
        select count(pp.position) as Centre
        from party_position pp join party p on pp.party_id = p.id
        where pp.position = 5
);
create view center_right as(
        select count(pp.position) as CentreRight
        from party_position pp join party p on pp.party_id = p.id
        where pp.position > 5 AND pp.position <= 7.5
);
create view far_right as(
        select count(pp.position) as FarRight
        from party_position pp join party p on pp.party_id = p.id
        where pp.position > 7.5 AND pp.position <= 10
);

--now merge them together.
create view left_right_histogram as (
        select a.FarLeft, b.CentreLeft, c.Centre, d.CentreRight, e.FarRight
        from far_left a join center_left b join center c join center_right d join far_right e
);
```

Result:



```
mysql> select * from left_right_histogram;
+---------+------------+--------+-------------+----------+
| FarLeft | CentreLeft | Centre | CentreRight | FarRight |
+---------+------------+--------+-------------+----------+
|       3 |          4 |      2 |           7 |        3 |
+---------+------------+--------+-------------+----------+
1 row in set (0.00 sec)
```

Appendix:

Creating Tables

```
CREATE TABLE party(Id INT UNSIGNED NOT NULL, Name VARCHAR(100) NOT NULL,
Abbreviation VARCHAR(10) NOT NULL, Start_Date DATE NOT NULL, End_Date DATE
NULL, Description VARCHAR(1000), PRIMARY KEY(Id));

CREATE TABLE Cabinet(Id INT UNSIGNED NOT NULL, Name VARCHAR(100) NOT
NULL, Start_Date DATE NOT NULL, End_Date DATE NULL, Election_Id INT UNSIGNED
NOT NULL UNIQUE, Previous_Cabinet_Id INT UNSIGNED NOT NULL UNIQUE
REFERENCES Cabinet(Id), PRIMARY KEY(Id));

CREATE TABLE Cabinet_Party(Cabinet_Id INT UNSIGNED NOT NULL, Id INT UNSIGNED
NOT NULL, Party_Id INT UNSIGNED NOT NULL, President BOOLEAN NOT NULL,
DESCRIPTION VARCHAR(1000) NULL, PRIMARY KEY(Cabinet_Id,Id));

CREATE TABLE Election(Id INT UNSIGNED NOT NULL, Type ENUM ('European
Parliament', 'Parliamentary election', 'Presidential Election'), Date DATE NOT NULL,
Total_Seats INT UNSIGNED NOT NULL, Electorate INT UNSIGNED NOT NULL,
Votes_Cast INT UNSIGNED NOT NULL, Valid_Votes INT UNSIGNED NOT NULL,
Previous_Election_Id INT UNSIGNED NOT NULL UNIQUE REFERENCES Election(Id),
PRIMARY KEY(Id));

CREATE TABLE Election_Result(Election_Id INT UNSIGNED NOT NULL, Id INT
UNSIGNED NOT NULL, Party_Id INT UNSIGNED NOT NULL, Seats_Contested INT
UNSIGNED NOT NULL, Seats_Won INT UNSIGNED NOT NULL, PRIMARY
KEY(Election_Id, Id));

CREATE TABLE President(Id INT UNSIGNED NOT NULL, Last_Name VARCHAR(50) NOT
NULL, First_Name VARCHAR(50) NOT NULL, Start_Date DATE NOT NULL, End_Date
DATE NULL, Party_Id INT UNSIGNED NOT NULL, Predecessor INT UNSIGNED NOT
NULL REFERENCES President(Id), Successor INT UNSIGNED NULL REFERENCES
President(Id), PRIMARY KEY(Id));

CREATE TABLE Party_Family(Party_Id INT UNSIGNED NOT NULL REFERENCES
party(Id), Family VARCHAR(50) NOT NULL, PRIMARY KEY(Party_Id, Family));

CREATE TABLE Party_Position(Party_Id INT UNSIGNED NOT NULL REFERENCES
party(Id), Position REAL UNSIGNED NOT NULL, PRIMARY KEY(Party_Id));
```

Alter Tables

ALTER TABLE party ALTER Description SET DEFAULT "";

ALTER TABLE party ADD CHECK(End_Date>Start_Date AND End_Date IS NOT NULL);

ALTER TABLE Cabinet ADD CONSTRAINT cabinet_election FOREIGN KEY (Election_Id) REFERENCES Election(Id), ADD CHECK(End_Date>Start_Date AND END_Date IS NOT NULL);

ALTER TABLE Cabinet_Party ADD FOREIGN KEY (Party_Id) REFERENCES Party(Id), ADD FOREIGN KEY (Cabinet_Id) REFERENCES Cabinet(Id);

ALTER TABLE Election ADD CHECK(Electorate>=Votes_Cast), ADD CHECK(Votes_Cast>=Valid_Votes), ADD CHECK(Date <= CURDATE());

ALTER TABLE Election_Result ADD FOREIGN KEY (Election_Id) REFERENCES Election(Id), ADD FOREIGN KEY (Party_Id) REFERENCES Party(Id), ADD CHECK(Seats_Contested >= Seats_Won);

ALTER TABLE President ADD FOREIGN KEY (Party_Id) REFERENCES Party(Id), ADD CHECK(End_Date>Start_Date AND End_Date IS NOT NULL);

ALTER TABLE Party_Position ADD CHECK(Position >= 0.0 AND Position <= 10.0);

Creating Trigger

```
DELIMITER //
CREATE TRIGGER Check_Cabinet_Formed_Date BEFORE INSERT ON Cabinet
FOR EACH ROW
BEGIN
        IF NEW.Start_Date < (SELECT Date FROM Election
                WHERE Election.Id=NEW.Election_Id)
        THEN
                SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT='Cabinet cannot be formed before its associated Election';
        ELSEIF NEW.Start_Date>CURDATE()
        THEN
                SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT='Cabinet cannot be formed in future';
        END IF;
END;//
DELIMITER ;
```

Populating Party Table

```
INSERT INTO Party VALUES(2608,"Debout la France","DLF",'2008-11-23',NULL,DEFAULT);

INSERT INTO Party VALUES(2612,"Union des démocrates et indépendants","UDI",'2012-9-
18',NULL,DEFAULT);

INSERT INTO Party VALUES(2609,"Alliance Centriste","AC",'2009-6-27',NULL,DEFAULT);

INSERT INTO Party VALUES(2628,"Front De Gauche","FDG",'2008-11-18',NULL,DEFAULT);

INSERT INTO Party VALUES(2501,"Party Radical","PRV",'1901-6-23','2017-12-9',"Merge into
Mouvement Radical");

INSERT INTO Party VALUES(2578,"Union pour la Démocratie Française","UDF",'1978-2-
1','2007-11-30',"Suceeded by MoDem");

INSERT INTO Party VALUES(2637,"Rassemblement pour la République","RPR",'1976-12-
5','2002-9-21',"Changed to UMP.");

INSERT INTO Party VALUES(2638,"Union pour un mouvement populaire","UMP",'2002-11-
17','2015-5-30',"Changed to The Republicans.");

INSERT INTO Party VALUES(2594,"Mouvement pour la France","MPF",'1994-1-
1',NULL,DEFAULT);

INSERT INTO Party VALUES(2615,"Les Républicains","LR",'2015-5-30',NULL,DEFAULT);

INSERT INTO Party VALUES(2520,"Parti communiste français","PCF",'1920-12-
30',NULL,DEFAULT);

INSERT INTO Party VALUES(2509,"Parti radical de Gauche","PRG",'1901-6-23','2017-12-
9',DEFAULT);

INSERT INTO Party VALUES(2617,"French Socialist Party","PSF",'1969-5-
4',NULL,DEFAULT);

INSERT INTO Party VALUES(2584,"Les Verts","LV",'1984-1-20','2010-11-13',"Merge into
EELV");

INSERT INTO Party VALUES(2613,"Europe Écologie Les Verts","EELV",'2010-11-
13',NULL,DEFAULT);

INSERT INTO Party VALUES(2643,"La République En Marche!","REM",'2016-4-
6',NULL,DEFAULT);

INSERT INTO Party VALUES(2601,"Mouvement démocrate","MoDem",'2007-12-
1',NULL,DEFAULT);

INSERT INTO Party VALUES(2546,"Rassemblement National","RN",'1972-10-
5',NULL,DEFAULT);

INSERT INTO Party VALUES(2644,"La France Insoumise","FI",'2016-2-10',NULL,DEFAULT);
```

Populating Election Table

```
INSERT INTO Election VALUES(432017,"Parliamentary election",'2017-6-18',577,47293103,20164615,18176066,432012);

INSERT INTO Election VALUES(432012,"Parliamentary election",'2012-6-17',577,43233648,23952486,23029308,432007);

INSERT INTO Election VALUES(432007,"Parliamentary election",'2007-6-17',577,35225248,21129039,20406454,432002);

INSERT INTO Election VALUES(432002,"Parliamentary election",'2002-6-16',577,36783746,22186165,21221026,431997);

INSERT INTO Election VALUES(431997,"Parliamentary election",'1997-6-18',577,37626821,26886073,25189627,431993);

INSERT INTO Election VALUES(12017,"Presidential election",'2017-5-7',1,47568693,35467327,31381603,12012);

INSERT INTO Election VALUES(12012,"Presidential election",'2012-5-7',1,46066307,37016309,34861353,12007);

INSERT INTO Election VALUES(12007,"Presidential election",'2007-4-22',1,44472733,37342004,35773578,12002);

INSERT INTO Election VALUES(12002,"Presidential election",'2002-4-21',1,41191169,32832295,31062988,12095);

INSERT INTO Election VALUES(11995,"Presidential election",'1995-4-23',1,39976944,31845819,29943671,12088);
```

Populating Election Result Table

```
INSERT INTO Election_Result VALUES(432017,10530,2643,430,308);

INSERT INTO Election_Result VALUES(432017,10531,2601,50,42);

INSERT INTO Election_Result VALUES(432017,10532,2615,130,112);

INSERT INTO Election_Result VALUES(432017,10534,2612,40,18);

INSERT INTO Election_Result VALUES(432017,10535,2617,50,30);

INSERT INTO Election_Result VALUES(432017,10536,2509,5,3);

INSERT INTO Election_Result VALUES(432017,10537,2644,60,17);

INSERT INTO Election_Result VALUES(432017,10538,2520,100,10);

INSERT INTO Election_Result VALUES(432017,10539,2546,120,8);
INSERT INTO Election_Result VALUES(432017,10540,2613,30,1);
INSERT INTO Election_Result VALUES(432017,10541,2608,30,1);

INSERT INTO Election_Result VALUES(432012,10430,2617,300,280);
INSERT INTO Election_Result VALUES(432012,10431,2613,150,39);
INSERT INTO Election_Result VALUES(432012,10432,2509,120,12);
INSERT INTO Election_Result VALUES(432012,10433,2638,300,194);
INSERT INTO Election_Result VALUES(432012,10434,2609,70,17);
INSERT INTO Election_Result VALUES(432012,10435,2501,40,21);
INSERT INTO Election_Result VALUES(432012,10436,2520,10,6);
INSERT INTO Election_Result VALUES(432012,10437,2546,20,2);
INSERT INTO Election_Result VALUES(432012,10438,2601,30,2);
INSERT INTO Election_Result VALUES(432012,10439,2628,30,2);

INSERT INTO Election_Result VALUES(432007,10330,2638,360,313);
INSERT INTO Election_Result VALUES(432007,10331,2609,70,22);
INSERT INTO Election_Result VALUES(432007,10332,2594,30,10);
INSERT INTO Election_Result VALUES(432007,10333,2617,300,186);
INSERT INTO Election_Result VALUES(432007,10334,2520,70,15);
INSERT INTO Election_Result VALUES(432007,10335,2509,20,22);
INSERT INTO Election_Result VALUES(432007,10336,2584,20,4);
INSERT INTO Election_Result VALUES(432007,10337,2601,24,1);
INSERT INTO Election_Result VALUES(432007,10338,2546,30,0);

INSERT INTO Election_Result VALUES(432002,10230,2638,400,369);
INSERT INTO Election_Result VALUES(432002,10231,2578,60,29);
INSERT INTO Election_Result VALUES(432002,10232,2594,5,1);
INSERT INTO Election_Result VALUES(432002,10233,2617,300,140);
INSERT INTO Election_Result VALUES(432002,10234,2520,100,21);
INSERT INTO Election_Result VALUES(432002,10235,2509,110,7);
INSERT INTO Election_Result VALUES(432002,10236,2584,30,9);
INSERT INTO Election_Result VALUES(432002,10237,2546,20,0);

INSERT INTO Election_Result VALUES(431997,10130,2617,400,255);
INSERT INTO Election_Result VALUES(431997,10131,2520,100,35);
INSERT INTO Election_Result VALUES(431997,10132,2584,20,7);
INSERT INTO Election_Result VALUES(431997,10133,2509,35,23);
INSERT INTO Election_Result VALUES(431997,10135,2637,300,139);
INSERT INTO Election_Result VALUES(431997,10136,2584,200,114);
INSERT INTO Election_Result VALUES(431997,10137,2546,20,1);

INSERT INTO Election_Result VALUES(12017,10520,2643,1,1);
INSERT INTO Election_Result VALUES(12017,10521,2546,1,0);
INSERT INTO Election_Result VALUES(12012,10420,2617,1,1);
INSERT INTO Election_Result VALUES(12012,10421,2638,1,0);
INSERT INTO Election_Result VALUES(12007,10320,2638,1,1);
INSERT INTO Election_Result VALUES(12007,10321,2617,1,0);
INSERT INTO Election_Result VALUES(12002,10120,2637,1,1);
INSERT INTO Election_Result VALUES(12002,10121,2546,1,0);
INSERT INTO Election_Result VALUES(11995,10020,2637,1,1);
```

<u>Populating Cabinet Table</u>

```
INSERT INTO Cabinet VALUES(100,"Édouard Philippe",'2017-6-20',NULL,432017,97);
INSERT INTO Cabinet VALUES(93,"Lionel Jospin",'1997-6-20','2002-5-7',431997,91);
INSERT INTO Cabinet VALUES(94,"Jean-Dominique",'2002-6-18','2007-5-15',432002,93);
INSERT INTO Cabinet VALUES(96,"François Fillon",'2007-6-19','2012-5-10',432007,94);
INSERT INTO Cabinet VALUES(97,"Jean-Manuel-Bernard",'2012-6-19','2017-5-10',432012,96);
```

Populating Cabinet_Party Table

```
INSERT INTO Cabinet_Party VALUES(100,1,2643,TRUE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(100,2,2601,FALSE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(97,1,2617,TRUE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(97,2,2613,FALSE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(97,3,2509,FALSE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(96,1,2638,TRUE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(96,2,2594,FALSE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(94,1,2638,TRUE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(94,2,2578,FALSE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(94,3,2584,FALSE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(93,1,2617,FALSE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(93,2,2520,FALSE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(93,3,2509,FALSE,DEFAULT);
INSERT INTO Cabinet_Party VALUES(93,4,2509,FALSE,DEFAULT);
```

Populating Party_Position Table

```
INSERT INTO Party_Position VALUES(2501,7.5);
INSERT INTO Party_Position VALUES(2509,2.5);
INSERT INTO Party_Position VALUES(2520,0.0);
INSERT INTO Party_Position VALUES(2546,8.75);
INSERT INTO Party_Position VALUES(2578,7.5);
INSERT INTO Party_Position VALUES(2584,2.5);
INSERT INTO Party_Position VALUES(2594,8.75);
INSERT INTO Party_Position VALUES(2601,6.25);
INSERT INTO Party_Position VALUES(2608,8.75);
INSERT INTO Party_Position VALUES(2609,5.0);
INSERT INTO Party_Position VALUES(2612,6.25);
INSERT INTO Party_Position VALUES(2613,2.5);
INSERT INTO Party_Position VALUES(2615,7.5);
INSERT INTO Party_Position VALUES(2617,2.5);
INSERT INTO Party_Position VALUES(2628,1.25);
INSERT INTO Party_Position VALUES(2637,7.5);
INSERT INTO Party_Position VALUES(2638,7.5);
INSERT INTO Party_Position VALUES(2643,5.0);
INSERT INTO Party_Position VALUES(2644,1.25);
```

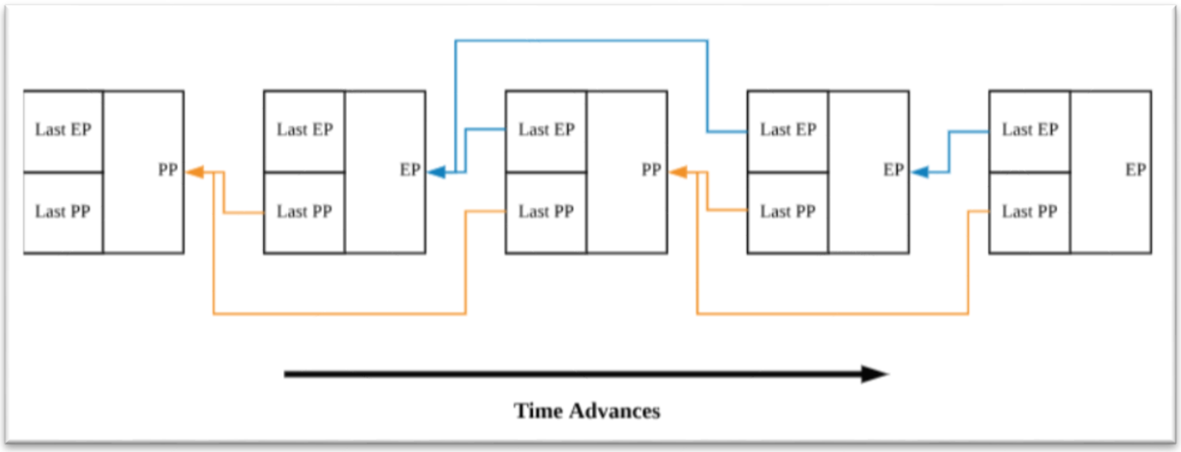
Populating Party_Family Table

```
INSERT INTO Party_Family VALUES(2501,"Liberalism");
INSERT INTO Party_Family VALUES(2509,"Social Liberalism");
INSERT INTO Party_Family VALUES(2520,"Communism");
INSERT INTO Party_Family VALUES(2546,"National Conservatism");
INSERT INTO Party_Family VALUES(2578,"Christian Democracy");
INSERT INTO Party_Family VALUES(2584,"Green Politics");
INSERT INTO Party_Family VALUES(2594,"National Conservatism");
INSERT INTO Party_Family VALUES(2601,"Centrism");
INSERT INTO Party_Family VALUES(2608,"National Conservatism");
INSERT INTO Party_Family VALUES(2609,"Liberalism");
INSERT INTO Party_Family VALUES(2612,"Liberalism");
INSERT INTO Party_Family VALUES(2613,"Green Politics, Regionalism");
INSERT INTO Party_Family VALUES(2615,"Conservatism");
INSERT INTO Party_Family VALUES(2617,"Democratic Socialism");
INSERT INTO Party_Family VALUES(2628,"Socialism");
INSERT INTO Party_Family VALUES(2637,"Conservatism");
INSERT INTO Party_Family VALUES(2638,"Conservatism");
INSERT INTO Party_Family VALUES(2643,"Social Liberalism");
INSERT INTO Party_Family VALUES(2644,"Democratic Socialism");
```

Further Improvements

As mentioned, there is a potential to build a database systems that can handle every political science data on parties, elections and governments of different countries. This expansion can be justified by just adding one more country table that stores all democratic countries. Then, we can merely just assign a country code to every primary key in every table to form a composite primary key. This allows us to distinguish between data between different countries in a concise and systematic manner.

Another interesting or peculiar fact about the data structure of the model is storing previous election Id or previous cabinet Id in every entry set. This actually allow us to construct a linked list of elections or cabinets that can be achieved in embedded SQL. Each row of the election table records information about a single election. The row includes the ID of the previous Parliamentary election and the previous EP election (using attributes previous parliament election id and previous parliament election id). These two attributes essentially create two independent linked lists within the table. However, it’s more complicated than that, because even a Parliamentary election has a reference to the previous EP election, and even an EP election has a reference to the previous Parliamentary election. This diagram may help you understand the structure embedded in the election table:


Last EP
Last PP
PP
Last EP
Last PP
EP
Last EP
Last PP
PP
Last EP
Last PP
EP
Last EP
Last PP
EP
Time Advances