Group 27 Documentation

Christian Cook (cjc3ne) Patrick Hinson (pjh2dz)
Katie Martins (kem5en) Maria Parnell (map5ed)
George Taliaferro (wgt7xp)

May 3, 2021

1 Project Information

1.1 Introduction

Our application is a civics-based information system designed to model historical and contemporary relationships among U.S. congress members, as well as the laws they create. Our relational database contains information about how federal Representatives and Senators have historically voted, and allows us to explore how the makeup of Congress may impact a vote.

Specifically, our database's design allows us not only to query party information with regard to a particular vote, but also investigate how other components such as geography and committee membership can impact voting. The United States Congress is a complex system with many bylaws and traditions that make legislating complicated. Our model aims to help users gain a better understanding of why their representatives vote the way they do and to increase civic participation.

1.2 Requirements Document

One of our application's most basic functions is to inform users about their representatives in Congress. This can include details like their name, party, and which district they represent if they are in the House. We achieve this through retrieving data from the House Member and Senate Member tables in our database. This allows voters to stay informed about how well their representatives reflect their interests, and may help convince them to vote for or against those representatives in future elections.

More informed voters may take an interest in active committees in Congress. Select Committees are special purpose committees that change frequently, so it may be informative to learn what Select Committees are currently active. We can again retrieve information from our Committee table searching for Select committees in the current session of Congress. This may provide insight into what issues Congress currently deems the most important, and help voters stay up to date on current events.

Since our primary purpose is to help inform voters about their representatives, which is a significant task, we implement security measures to maintain the integrity of the database. Our application enforces two separate login user types. We have general users, who have query permissions only, and admin users, who have insert and update permissions. The general voting population should have viewing access to their representatives' voting records and the data we are helping them connect with it. They can achieve this through query access alone, therefore they do not need insert or update permissions. However, new data must be added to the database over time, and in the future we may want to expand the abilities of our database. Admin users thus have permission to insert, update, and delete in order to maintain the database over time.

2 The Design Process

2.1 Design Decisions

Our application is web-based, running primarily on HTML, CSS, and PHP. The languages we used are well-suited to developing web applications with HTML and CSS being used for design purposes and PHP for functionality. We felt that a web-based project would be the most accessible to voters trying to use our application to gain civic insight. Every U.S. voter should have access to this information, and through our application, they can access it anywhere with an internet connection.

Our ER diagram is shown in Figure 1 below, and will be referenced throughout the rest of this section. The design of our relational database centers around Congress Members and their division into House and Senate Members. These entity sets are the hub of the complex set of relationships in the U.S. legislature. One decision we had to make early on was how to differentiate between styles of representation for each chamber. Since Senators represent an entire state, we were able to simply give Senate Members a State attribute. House members were more difficult to design since they must represent a specific district within a state. We decided that it would be best to maintain a separate entity set for a District so that we could incorporate other relevant data such as population as well as the state. This design also gives us flexibility for future development, allowing us to eventually incorporate demographic data as well.

Designing the relationship between Congress Members and the Votes they cast on Roll Calls was also a challenge. We discussed multiple methods of representing these relationships, such as a ternary relationship between all three. We decided that the most accurate method would be to have Congress Members cast a Vote, and have the Vote be on a specific Roll Call. Since members are required to cast a vote (abstentions count as votes) and votes are always on a roll call, a ternary relationship only adds unnecessary complexity. This allows us to connect specific representatives to the laws they vote on with minimal extraneous relationships.

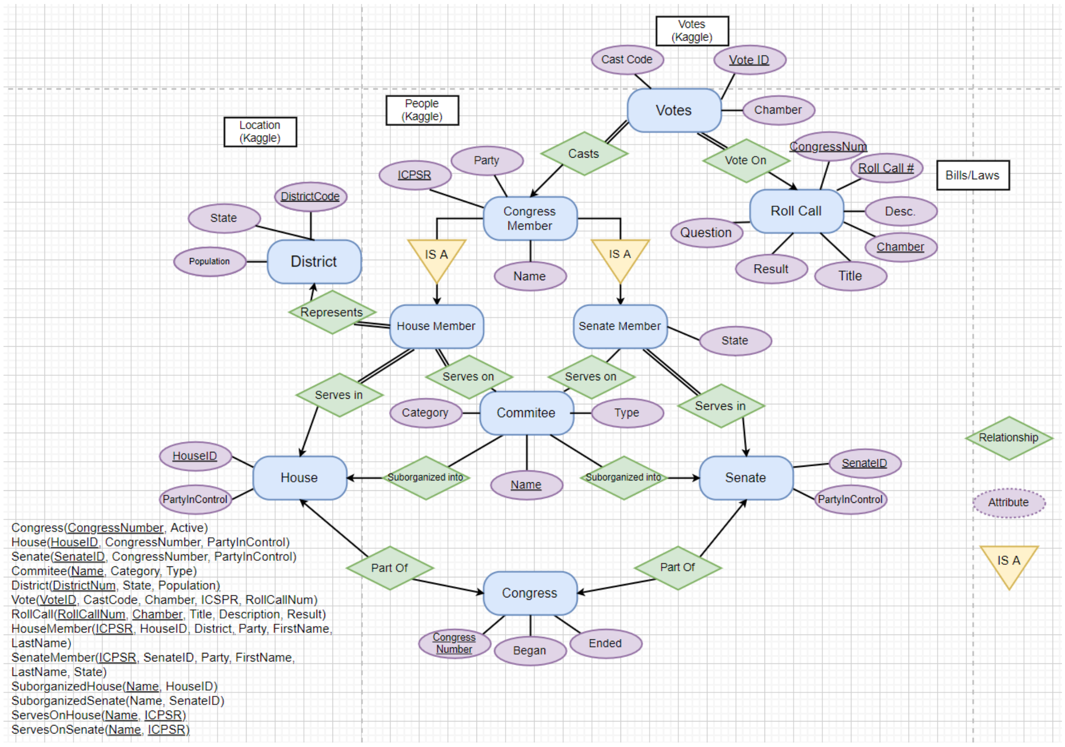
Translating our diagram into schema statements was a relatively straightforward process. Entity sets were directly translated into tables, while most relationships led to one entity storing the primary key of the other. One example of this is House Members storing the ID of the House session they belong to. Other relationships are represented by individual tables storing both primary keys of the two entities. Normalizing the tables was also a simple undertaking, as we had very few functional dependencies. This process is outlined in Appendix A, and the resulting table schemas are listed in Appendix B.

Our database should be secure to prevent misinformation being spread to voters seeking knowledge about their representatives. We implement some security measures to prevent this. We have a hidden, separate user whose only purpose is to read the login table for any other user trying to log in. This user has no permissions to access the database in any way other than to verify login information. When the information is verified, the user type is changed appropriately. This adds a layer of security between the database and potential hackers.

We also have a basic level of Role-Based Access Control, separating permissions among general users, who only need to query the data, and administrators, who maintain the underlying database. This ensures that users only have permissions that are appropriate for their use cases, helping prevent malicious manipulation of the database.

We also use Triggers to help maintain the integrity of our database. If either a House Member or a Senate Member is added to a committee, we check that they are in the current session of Congress. Congress members cannot be retroactively added to a committee, so any insertions or updates that place a Congress Member on a new committee ensure that they are a current member of Congress.

2.2 ER Diagram



Congress(CongressNumber, Active)
House(HouseID, CongressNumber, PartyInControl)
Senate(SenateID, CongressNumber, PartyInControl)
Commitee(Name, Category, Type)
District(DistrictNum, State, Population)
Vote(VoteID, CastCode, Chamber, ICSPR, RollCallNum)
RollCall(RollCallNum, Chamber, Title, Description, Result)
HouseMember(ICPSR, HouseID, District, Party, FirstName, LastName)
SenateMember(ICPSR, SenateID, Party, FirstName, LastName, State)
SuborganizedHouse(Name, HouseID)
SuborganizedSenate(Name, SenateID)
ServesOnHouse(Name, ICPSR)
ServesOnSenate(Name, ICPSR)

Figure 1: ER Diagram

3 Evaluation of Product

3.1 Testing Procedures

Since our application includes both user and admin login options, we divided up the permissions for each of these groups. We tested that users were not able to insert, update, or delete any data in the database. The purpose of this application for users is to query for existing information, so we restrict their permissions. Admins, however, have full access to query and change the database, as they will be maintaining it over time.

In the creation stage, we were able to verify that our application could both create/connect to our database and create the tables for it. We tested querying through various combinations of possible searches based on different motivations seen in our example queries. We tested insertion, deletion, and updating both as general users and as admins. General users are denied access, but admins can successfully insert new data into, delete data from, and update data in tables.

3.2 Sample Data and Queries

If an uninformed voter from Virginia would like more details on who their Senators in the House are, they can search by their state, and our application would submit a query like this

```
SELECT *
FROM HouseMember
WHERE District LIKE 'VA%' AND HouseNumber = 117
```

A voter can check what political party is currently in charge of the Senate after a query like this

```
SELECT *
FROM Senate
WHERE CongressNumber = 117
```

A voter can see details on votes about the environment after our application runs a query like this

```
SELECT *
FROM RollCallInfo
WHERE Description LIKE '%environment%'
```

An admin will want to insert and update info on committees as new Select and other committees are created/disbanded. The application will them submit this query

```
INSERT INTO Committee
VALUES ("Committee on Database Systems", "Education", "Select");
```

4 Appendix A: Functional Dependencies and Normalization

House(<u>HouseID</u>, CongressNumber, PartyInControl) - Normalized
FDs: HouseID $\rightarrow$ CongressNumber, PartyInControl

Senate(<u>SenateID</u>, CongressNumber, PartyInControl) - Normalized
FDs: SenateID $\rightarrow$ CongressNumber, PartyInControl

Committee(<u>Name</u>, Category, Type) - Normalized
FDs: Name $\rightarrow$ Category, Type

District(DistrictNum, State, Population) - Normalized
FDs: DistrictNum → State, Population

Vote(VoteID, CastCode, Chamber, ICSPR, RollCallNum) - Normalized
FDs: VoteID → CastCode, Chamber, ICSPR, RollCallNum

RollCall(CongressNumber, RollCallNum, Chamber, Question, Title, Description, Result)
FDs:
CongressNumber, RollCallNum, Chamber → Question, Title, Description, Result
Question, Title → Description

Fc:
RollCallNum, Chamber → Result
Question, Title → Description

Normalized:
RollCallResults(CongressNumber, RollCallNum, Chamber, Result) - Normalized
RollCallInfo(Question, Title, Description) - Normalized

HouseMember(ICPSR, HouseID, District, Party, FirstName, LastName) - Normalized
FDs: ICPSR → HouseID, District, Party, FirstName, LastName

SenateMember(ICPSR, SenateID, State, Party, FirstName, LastName) - Normalized
FDs: ICPSR → SenateID, State, Party, FirstName, LastName

Congress(CongressNumber, Began, Ended) - Normalized

SuborganizedHouse(Name, HouseID) - Normalized

SuborganizedSenate(Name, SenateID) - Normalized

ServesOnHouse(Name, ICPSR) - Normalized

ServesOnSenate(Name, ICPSR) - Normalized

5 Appendix B: Database Schemas

House(HouseID, CongressNumber, PartyInControl)
Senate(SenateID, CongressNumber, PartyInControl)
Committee(Name, Category, Type)
District(DistrictNum, State, Population)
Vote(VoteID, CastCode, Chamber, ICSPR, RollCallNum)
RollCallResults(CongressNumber, RollCallNum, Chamber, Result)
RollCallInfo(Question, Title, Description)
HouseMember(ICPSR, HouseID, District, Party, FirstName, LastName)
SenateMember(ICPSR, SenateID, State, Party, FirstName, LastName)

Congress(CongressNumber, Began, Ended)
SuborganizedHouse(Name, HouseID)
SuborganizedSenate(Name, SenateID)
ServesOnHouse(Name, ICPSR)
ServesOnSenate(Name, ICPSR)