Charles Hardes CS 496 Fall 2016

Assignment 3, Part 1

Deliverables:

1.) Describe the data you will be modeling

The data I will be modeling is voter data, as in actual votes in a hypothetical presidential election. This is a theoretical and very rudimentary implementation of an online voting system where voters will be able to log in and cast their ballots online and have their vote immediately counted and logged into the voting database. Clearly, such a system could be implemented via tools that exist to include the NoSQL databasing tools being learned in this course. The primary barrier, of course, to the implementation of such a system is the extremely high level of security and transparency that would need to be ingrained in such a system, in order for there to be little to no doubt to its accuracy and authenticity. As this is a very immense problem that hasn't been solved or even been attempted to be solved yet, I will not tackle security/ transparency/ authenticity issues in this project, just a very basic implementation of how such a system might work.

Voter data will be broken down thusly as the five following NDB entity groups:

States	Districts	Votes	Voters	Electoral_College
Abbr	Number	Voter_Key	Voter_ID	EVoter_ID
Quant_Districts	State_Key	Candidate	Party	State_Key
Quant_Electors	Quant_Registered	Issues	Age	Candidate
Dist_Key_List	Vote_Key_List	Dist_Key	Sex	
Elector_Key_List		State_Key	Income_Lvl	
			Ethnicity	
			Education_Lvl	

I. Fields:

A. States: All 50 states of the United States

1. Abbr: The two-letter abbreviation ('OR')
2. Quant_Districts: The number of voting districts in the state
    a. Sum of Districts records in the given state
3. Quant_Electors: Number of members of electoral college in the state
    a. Sum of 'Electoral_College' records in the given State
    b. Tallied when electoral college member submits their ballot.
4. Dist_Key_List: A list of all 'Dist_Key's in the given state
    a. References the 'Districts' entity.
    b. One-to-many relationship.

5. Elector_Key_List: List of all Electoral_College keys (members of electoral college) for that state
    a. References the 'Electoral_College' entity
    b. One-to-many relationship

B. Districts: The districts dividing up each state
1. Number: The standard number that the district is represented by
2. State_Key:
    a. reference to "States" entity group
    b. One-to-one relationship
3. Quant_Registered: The number of registered voters in district
4. Vote_Key_List: Contains the full list of ordered 'Votes' keys for that district
    a. reference to Votes entity
    b. One-to-many relationship

C. Votes: The entity that represents actual votes, one for every voter
1. Voter_Key: Reference to 'Voters' entity (One-to-one relationship)
2. Candidate: A number from 1 to 5 representing the available candidates selected by this vote
3. Issues: A list of numbers representing a series of political issues the voter has selected as being important to them
4. Dist_Key:
    a. references the 'District' entity
    b. one-to-one relationship
5. State_Key:
    a. references the 'States' entity
    b. one-to-one relationship

D. Voters: Personal voter information is stored here for analysis of voter demographics. Like the Voter_ID, this information would theoretically be collected/ updated at voter registration by the government.
1. Voter_ID: Unique index assigned to every voter by registration.
2. Party: A number representing a selection of party choices for the voter to choose from
3. Age: A number from a selection of age groups selected by the voter
4. Sex: 'M' or 'F' or 'O' (other)
5. Income_Lvl: A number representing an income range selected by the voter
6. Ethnicity: A list of numbers representing one or more ethnicities chosen by the voter
7. Education_Lvl: A number representing an education level selected by the voter

E. Electoral_College: Stores very basic data for each member of the electoral college
1. EVoter_ID: Unique number assigned by government to each member of electoral college.
2. State_Key:
    a. Reference to the 'State_ID' property of the States entity.
    b. One-to-one relationship
3. Candidate: A number from 1 to 5 representing the available candidates selected by this vote (Identical to the same property in 'Votes' entity)

II. Rationale for this model: There are two primary use cases for this database, which would both occur largely independent of one another. That is to say, a very heavy amount of writing to the database would occur (during election day) with little to no reading from the database. Then, once polls close, the writing would stop altogether (except for small exceptions like absentee ballots being counted) and there would be a heavy amount of reading occurring from government officials as well as political analysts, and the public at large.

Because of the magnitude of voter data that would be hypothetically written to and read from this database, it was necessary to index voters in the Districts entity as a list. This necessitates writing each Vote_Key property to the District ID entity. I believe this is entirely necessary since we don't have the luxury of joining the Votes and Districts entities. With this method, one does not have to search through every single Votes record to find the corresponding state/district ID. One can simply get every reference to every vote by a given district in the Districts entity itself. This same method was applied to listing every district in each States entity as well as listing every electoral college member entity in the States entity.

III. Method of data collection:

A. Theoretical: This data would theoretically be entered via something like the cloud app I designed for Assignment 1 which can be viewed here. That would cover the Votes information, which is the meat and potatoes of the entire database. A similar app could be used to enter and collect the electoral college votes. A separate form would be necessary to enter the 'Voters' information, which would be theoretically collected by the government at the time of voter registration and linked to 'Votes' by the Voter_Key property when the ballot is cast. The list items in 'States' and 'Districts' would be populated dynamically by the casting of both voter and electoral college ballots in the server programing. However, all other fields in the States and Districts entities would have to be preexisting and based on / pulled from the Census Bureau data, etc.

B. Actual: For the purposes of this project, I will write a program to fill these entities with a whole lot of random values to be assigned to each field, will write ten or so votes to each district at a time. Data limit will be a factor in how much data I actually write to this database and will therefore do it incrementally.

IV. Other database considered: MongoDB

Looking into MongoDB, I decided to go with NDB due simply to the fact that MongoDB is primarily document based. Handling the lists of votes and districts I'm using in the States and Districts entities would be problematic with JSON objects and would lead to more replication of data than using keys as I am using in this method with NDB.