CPSC 471 - Normalization Document

Sam Safe - Aniket Bulusu - Sultan Alzoghaibi

December 1, 2025

Overview

We did not originally consider much normalization in our tables. After obtaining our data, reviewing it, and revisiting it after learning about normalization, we realized some normalization was required in order to ease our use of a database. In terns of simple data cleaning - a large portion of our data was cleaned manually by deleting columns that simply were not relevant to our project. Some other standardization techniques like using naming conventions, formatting spelling, spaces, capitalization, and other simple changes were performed. Outside of these changes, no significant data cleaning was required, since the raw data was immediately usable and relevant for our use-case. That said, we will go into further detail regarding normalization specifically. We will note that not all details of the normalization process can be covered in this document, but our general approach will be made clear. For specific details, we have chosen to include the original CSV's as well as the schema document which will show exactly the differences made to help the data conform to our needs.

Achieving 1NF

Several tables needed to be normalized to improve data consistency by removing repeated groupings of data. Household income, for instance, originally had columns that listed groupings of income category. We converted this into a long format, but associating each ward number thus creating multiple rows per ward. The benefit here is that we no longer have repeating column groupings, and simplifies the query process. Recreation facilities had 17 categories of recreation facility type, leading to numerous zero values, and information that was hard to access and too spread out. We reduced this by converting to a table with facility type and count numbers instead, significantly reducing complexity. Community services' original CSV document suffered from the same problem, and was fixed in the same way. Transportation had mode-specific columns with counts and percentages, and was normalized to a table with count and percent columns, which makes comparisons far easier. In almost all cases at this level of normalization, tables were converted from wide to long in order to simplify and reduce redundancy. This step reduced the amount of null values in our tables, let us do aggregation of data easier (especially since our entire project revolves around aggregated data), and stopped needlessly repeating columns for the same category of information.

Achieving 2NF & 3NF

The most significant (and only) table required to be altered at this level is ward election data. Looking at the CSV, we have a massive amount of re-

dundancy, a massive amount of data, and huge problems with important values being null. Furthermore, every single row repeats information like candidate names, station names, station ID's, and ward information. We broke this down originally to 6 tables in order to distinguish races and candidacies temporally. These tables are:

- Election - stores meta data about an election event
- Race - specifies mayoral or councilor sub-events
- Candidate - uniquely identifies candidates without having them repeat hundreds of times
- Candidacy - a relationship that connects candidates to races providing us access to event-specific candidate information. This itself can handle candidates running in multiple races, and each race including numerous candidates
- Voting Station - to identify stations around the city
- Result - vote counts which ties all of this together

In addition to the above, we created a central ward table. Instead of storing ward information needlessly across tables, we recognized that all relevant meta data regarding our project connects effortlessly to this table, and thus this became our center-piece. This enforces referential integrity through foreign keys, and makes connects all the necessary pieces to give us a full picture of data across wards.

So What

The added referential integrity constraints mean that all ward related tables are referencing ward as a foreign key. This means that no we have went from CSV's with no specific relationships to a database that enforces data consistency. Tables without data relevant to our visualization process now have data readily available, like ward population which now includes data like density, households, total count, whereas it did not include this originally. Overall, we pulled out independent entities from aggregated tables, reduced redundancy, created a reference table in ward, and added important relationships between our tables. A lot of tables are maintained at this level because the information they contain is uniquely queried at the same time, and are dependent. For instance, a table like ward population could be broken down by density and household, but this seems almost like over-doing it, because the information is relevant on each query. We also want to avoid missing the forest for the trees. All in all, our normalization efforts create a consistent database that is easier to query and is more performance-focused. We claim that the data-base in its entirety is at least at a 3NF level.