

Part C

Logical Design

In the logical design phase, I mapped the conceptual model from the ER diagram into a set of relational tables. Each table corresponds to an entity or relationship, with attributes defined as columns and primary keys ensuring unique identification. Relationships between entities, such as voters and their registrations or ballots and votes, were implemented using foreign keys.

Normalization was applied to organize the tables effectively and avoid redundancy. For instance, the Places table separates location details, which are then referenced by both Folks and Voting Centers. This design ensures data consistency and integrity while making updates easier, as changes in one table automatically propagate to related data.

Overall, the table structure supports all user requirements, providing a clear and efficient way to manage election data while minimizing storage issues and ensuring data accuracy.

Part D

Physical Design

The physical design phase translates the logical design into an operational MySQL database. Following the structure developed in the conceptual design, the database was implemented to reflect all entities, attributes, and relationships, ensuring data integrity and meeting the user requirements.

Database Design

The database was implemented in MySQL based on the structure defined in Section B. Each entity from the ER diagram was translated into a table, with primary keys for unique identification and foreign keys to establish relationships. Constraints like NOT NULL, UNIQUE, and ENUM were used to enforce data validity, such as limiting possible answers in the **Casted Vote** table to 'YES', 'NO', 'ABSTAIN', or 'NULL'.

Scripts for Table Management (Also Shown in README)

- **Creation Script:**
 - The createAll.sql script sets up all tables, attributes, primary keys, foreign keys, and constraints. This ensures that the database structure is consistent with the conceptual design.
- **Deletion Script:**
 - The dropAll.sql script provides a clean slate by removing all tables and related objects, allowing for re-creation when needed.

Data Loading

A loadAll.sql script was created to populate the database with sample data:

- 6 staff members
- 12 folks
- 6 places (distributed across 2 cities and 2 states)
- 3 voting centers (each with 4 operating periods)
- 4 ballots (each with 3 possible answers)
- 24 voting registrations (distributed across 3 ballots, 3 centers, and 2 months)
- 18 cast votes (distributed across 2 ballots)

Ensuring User Requirements

The design satisfies all user requirements by:

1. Enforcing accurate relationships between entities.
2. Avoiding redundancy by normalizing tables (up to 3NF).
3. Allowing scalability for future data additions, such as new ballots or voting centers.

Justification for Design Choices

The use of primary keys, foreign keys, and constraints ensures data consistency and supports efficient querying. The separation of entities into distinct tables (e.g., **Folks**, **Ballots**, **Casted Vote**) minimizes redundancy and simplifies data updates. Relationships are clear and easy to maintain, supporting the overall functionality of the elections system.