Names: Anushka Krishnakumar, Lisa Walker, Gavin Bowman, Katrina Gutierrez, Kevin McGrath

Stage IV – Elaboration: Design

Each team will review the Database Model document with stakeholders, and update the model as needed.

- JOURNALIST: (JournalID) → JName(FName, LName), Publication
- COMMUNITY_MEMBER: (MemberID) → CName(FName, LName), Township
- COMMUNITY_FORM: (FormID) → CName, MemberID, Township, Questions (Category, Summary, Importance, Who's Affected by Issue, Possible Solutions)

*UPDATE***: Separate databases are now a single database*

Demonstrate that all relations in the relational schema are normalized to BCNF.

- For each table, specify whether it is BCNF or not, and explain why.
    - All of our tables are in BCNF. Given that each table has only one primary key, they cannot be derived any further. More specifically, in the JOURNALIST table, JournalID determines JName and publication; in COMMUNITY_MEMBER determines CName and township; in COMMUNITY_FORM, FormID determines CName, MemberID, Township, and Questions. Since each table determines these attributes independently from the others, we know that they are all in Boyce-Codd Normal Form.
- For each table that is not in BCNF, show the complete process that normalizes it to BCNF
    - All of our tables are BCNF.

Define the different views required. For each view list the data and transaction requirements.

- In a database, a view is the result set of a stored query on the data, where the database users can query just as they would in a persistent database collection object.
    - The journalist users will be able to create read-only views with their queries.
        - They can view community forms, whether or not a journalist has been verified, list the names of people who have submitted community forms.
    - Sustainable Media Database Administrators will be able to create updatable views with their queries.
        - They can INSERT, UPDATE, and DELETE community forms, verified journalists, names of people who have submitted forms and other actions/queries.
    - To make our database secure, we will be using query authorization statements such as GRANT and REVOKE, in order to hide certain tuples from the journalist users.
    - Journalists will only be able to view the COMMUNITY_MEMBER database.

- Give a few examples of queries, in English, to illustrate.
    - List community forms from the township "Ewing"
        - $\pi_{\text{Township = "Ewing"}}$ COMMUNITY_FORMS

    - List community forms that categorize under the “Pollution” category
        - $\pi_{\text{Category = “Pollution”}}$COMMUNITY_FORMS
    - List community members that have submitted a form
        - $\pi_{\text{Fname, LName}}$(COMMUNITY_MEMBER $\bowtie_{\text{(MemberID)}}$ COMMUNITY_FORMS)
    - List Journalists that have been verified (Assuming a JournalID is a valid number once they are verified)
        - VERIFIED $\rightarrow \pi_{\text{JournalID}}$ JOURNALIST
        - UNVERIFIED $\rightarrow \pi_{\text{JournalID= NULL}}$ JOURNALIST
        - ALL_VER $\rightarrow$ VERIFIED - UNVERIFIED
        - RESULT $\rightarrow \pi_{\text{FName, LName}}$ (JOURNALIST * ALL_VER)

Design a complete set of queries to satisfy the transaction requirements identified in the previous stages.

- SELECT * FROM JOURNALIST;
    - This is a query in SQL with everything stored in the JOURNALIST table
    - It will have information for all the following attributes:JournalID, JName(FName, LName), Publication
- SELECT * FROM COMMUNITY_FORM;
    - This is a query in SQL with everything stored in the COMMUNITY_FORM table
    - It will have information for all the following attributes: FormID, CName(FName, LName), MemberID, Township, Questions(Category, Summary, Importance, Who’s Affected by Issue, Possible Solutions)
- SELECT * FROM COMMUNITY_MEMBER;
    - This is a query in SQL with everything stored in the COMMUNITY_MEMBER table
    - It will have information for all the following attributes: MemberID, CName(FName, LName), Township