Madison Bavosa, Emmanuel Pasteur, EJ Gasataya
CSC 315 - Phase 4 - Elaboration: Database Design

1. Demonstrate that all the relations in the relational schema are normalized to Boyce–Codd normal form (BCNF).

- Before determining whether each relation is in BCNF, we need to first identify the *possible FDs (Functional dependencies)* in each relation, and based on our observations, determine whether it is fully normalized. BCNF requires that each relation must also satisfy the prior normalization rules, such as 2NF and 3NF.

1. For each table, specify whether it is in BCNF or not, and explain why.
   a. MUNICIPALITY

MUNICIPALITY						
Total_Miles_Traveled	Municipality_name	MPO_Organization	MPO_Year_Adoption	Zip_Code	Vehicle_GHGs_Emitted	County_Name

- Possible FDs for this relation:
  - Zip_Code -> Municipality_name
  - {Municipality_name, Zip_Code} -> Vehicle_GHGs_Emitted
  - {Municipality_name, Zip_Code} -> Total_Miles_Traveled
  - {Municipality_name, Zip_Code} -> County_Name
  - {MPO_Organization} -> MPO_Year_Adoption
  - {Municipality_name, Zip_Code} -> MPO_Organization
- This relation is not in BCNF for the following reasons:
  - Since Municipality_name depends on Zip_Code and Municipality_name in conjunction with Zip_Code determines several attributes, **there exists a transitive dependency which violates 3NF.** Similarly, Municipality_name and Zip_Code determine the attribute MPO_Organization which determines the attribute MPO_Year_Adoption. This is another transitive dependency.
  - **MPO_Year_Adoption is a multivalued attribute which means there exists a nested relation which violates 1NF.** For example, the MPO organization "NJTPA" can either be adopted in 2017 or in 2019. Thus the MPO_Year_Adoption for a given organization is multivalued.

   b. ZIP_CODE

ZIP_CODE	
Zip_Code	Municipality_name

- The ZIP_CODE relation is in BCNF because it only contains the key attributes of Zip_Code and Municipality_name, and both are primary keys at the same time.

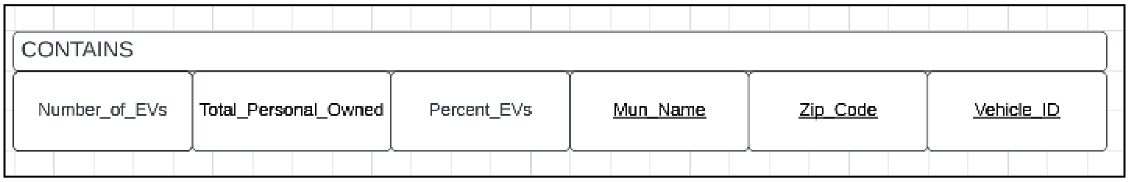
   c. VEHICLE

VEHICLE			
Vehicle_ID	Vehicle_Type	Mun_Name	Zip_Code

- Possible FDs for this relation:
  - Vehicle_ID -> Vehicle_Type, Mun_Name, Zip_Code

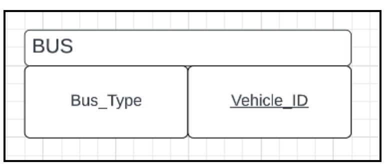
- The VEHICLE relation is in BCNF because Vehicle_Type, Mun_Name, and Zip_Code can all be determined **only by Vehicle_ID,** which is the primary key for this relation.

d. CONTAINS



- Possible FDs for this relation:
    - {Municipality_name, Zip_Code, Vehicle_ID} -> Number_of_EVs
    - {Municipality_name, Zip_Code, Vehicle_ID} -> Total_Personal_Owned
    - {Number_of_EVs, Total_Personal_Owned} -> Percent_EVs
- This relation is not normalized to BCNF, because the following dependency: {Number_of_EVs, Total_Personal_Owned} -> Percent_EVs, does not contain a prime key attribute in the dependency. **We said that the number of electric vehicles, the total number of personal vehicles owned, and the percentage of EVs are not primary key attributes,** and 3NF is violated because of this; thus not satisfying BCNF.
- We also observed that there is a transitive dependency between the percentage of EVs that is **indirectly determined by the municipality name and zip code**.
    - Example, if {Municipality_name, Zip_Code, Vehicle_ID} -> Number_of_EVs and {Municipality_name, Zip_Code, Vehicle_ID} -> Total_Personal_Owned
    - Then {Municipality_name, Zip_Code} should also be able to determine Percent_EVs because of this following FD:
        - {Number_of_EVs, Total_Personal_Owned} -> Percent_EVs.

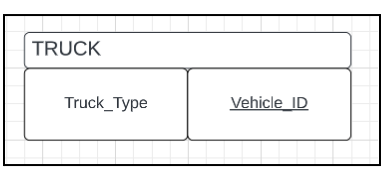
e. BUS



- We suspect there will be only one FD for this relation:
    - {Vehicle_ID} -> Bus_Type

- The BUS relation is in BCNF because it contains only the key attribute which is the Vehicle_ID and Bus_Type which relies on Vehicle_ID.

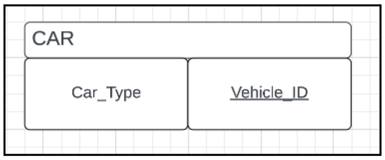
f. TRUCK



- We suspect there will be only one FD for this relation:
    - {Vehicle_ID} -> Truck_Type

- The TRUCK relation is also in BCNF because the Truck_Type attribute is the only attribute (and non-key) that is determined by the Vehicle_ID attribute.

g. CAR



- We suspect there will be only one FD for this relation:
    - {Vehicle_ID} -> Car_Type

- Similarly, the CAR relation is also in BCNF because the Car_Type attribute is the only attribute (and non-key) that is determined by the Vehicle_ID attribute.

For each table that is not in BCNF, show the complete process that normalizes it to BCNF.

- We first need to identify the primary keys that were elicited from the Relational Schema from Iteration III. Based on our schema, the primary keys are {Municipality_Name, Zip_Code, Vehicle_ID}. The Vehicle ID attribute will be randomly generated with unique values for its identity. In other words, the Vehicle ID will be a *surrogate key*. Since surrogate keys are also unique, they do not depend on any other relation.

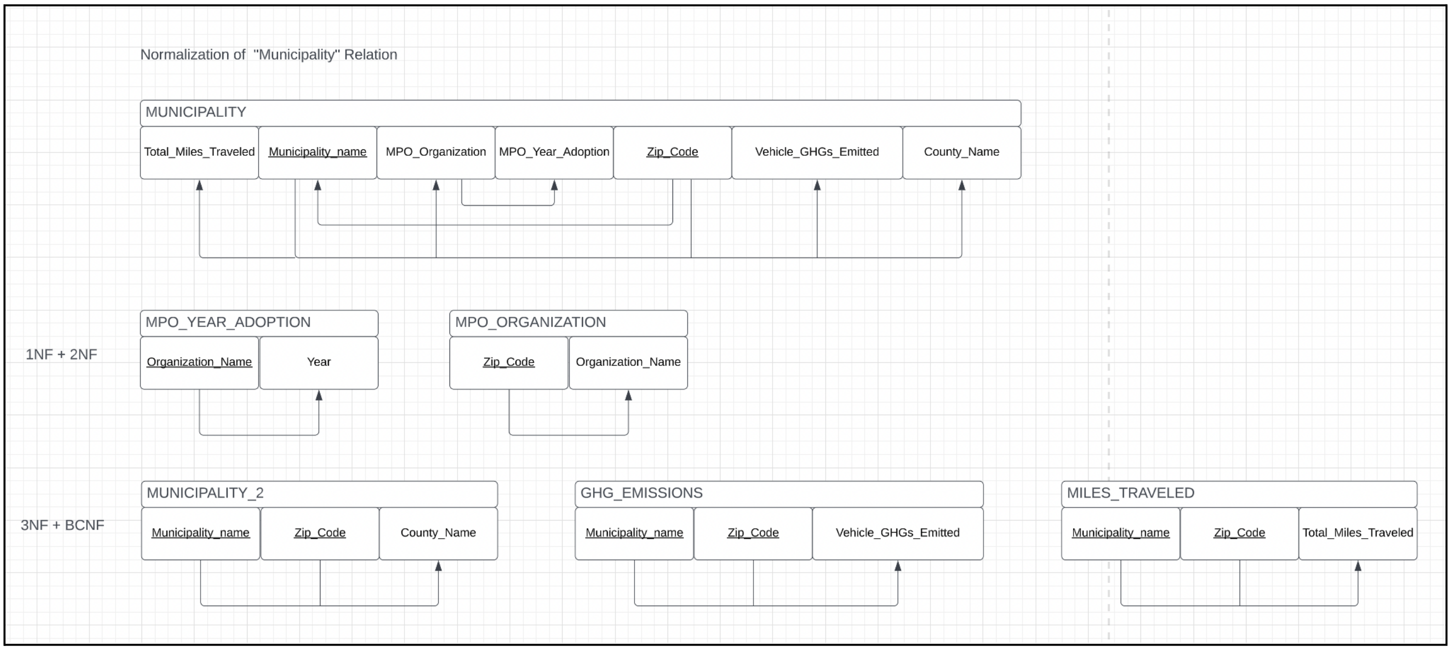
    - **Process for determining Municipality_Name and Zip_Code as the primary key for the MUNICIPALITY relation**
        - Attributes: Municipality_Name, Zip_Code, Vehicle_GHGs_Emitted, Total_Miles_Traveled, County_Name, MPO_Organization, MPO_Year_Adopted
        - Functional dependencies
            - {Municipality_Name, Zip_Code} -> Vehicle_GHGs_Emitted
            - {Municipality_Name, Zip_Code} -> Total_Miles_Traveled
            - {Municipality_Name, Zip_Code} -> County_Name
            - {Municipality_Name, Zip_Code} -> MPO_Organization
            - MPO_Organization -> MPO_Year_Adopted
                - This is a transitive dependency, meaning that MPO_Year_Adopted is dependent on {Municipality_Name, Zip_Code}
        - Functional dependency closures
            - {Municipality_Name, Zip_Code}+ -> {Vehicle_GHGs_Emitted, Total_Miles_Traveled, County_Name, MPO_Organization, MPO_Year_Adopted}

    - **Process for determining Municipality_Name, Zip_Code, and Vehicle_ID as the primary key for the CONTAINS relation**
        - Attributes: Mun_Name, Zip_Code, Vehicle_ID, Number_of_EVs, Total_Personal_Owned, Percent_EVs
        - Functional dependencies
            - {Municipality_name, Zip_Code, Vehicle_ID} -> Number_of_EVs
            - {Municipality_name, Zip_Code, Vehicle_ID} -> Total_Personal_Owned
            - {Number_of_EVs, Total_Personal_Owned} -> Percent_EVs
                - Since Number_of_EVs and Total_Personal_Owned both rely on {Municipality_name, Zip_Code, Vehicle_ID}, this is a transitive dependency
        - Functional dependency closures

- {Municipality_name, Zip_Code, Vehicle_ID}+ -> (Number_of_EVs, Total_Personal_Owned, Percent_EVs)

- We need to further normalize CONTAINS and MUNICIPALITY relations. We drew the observed functional dependencies as a starting point, and broke down the relation based on normalization requirements for 1NF, 2NF, 3NF, and BCNF.

MUNICIPALITY

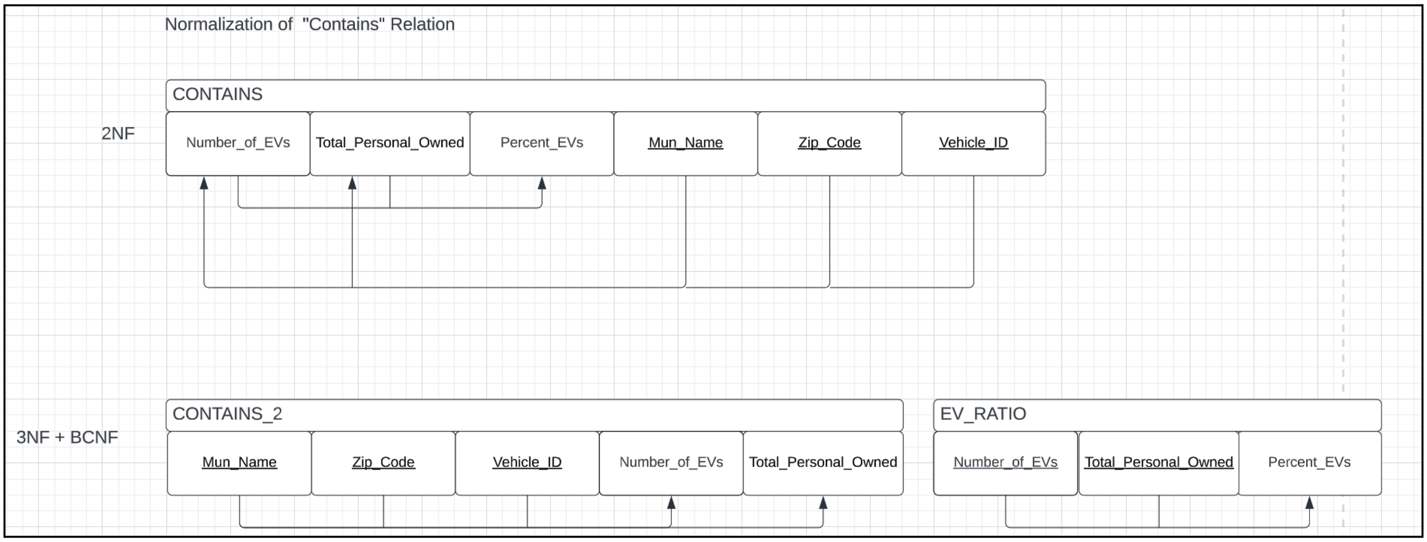


- MPO_YEAR_ADOPTION and MPO_ORGANZATION relations were created to satisfy 1NF and 2NF. Requirements.
  - The Year attributes refer to the Year adoption from a given organization. Based on our Excel data, an organization name may contain **multiple years.** For example, the MPO organization “NJTPA” can either be adopted in 2017 or in 2019. This causes the year attribute to be multivalued in the original MUNICIPALITY relation, which violated the 1NF requirement. Thus the MPO_YEAR_ADOPTION was created.

- Another reason why MPO_ORGANZATION and MPO_YEAR_ADOPTION was made is because in the original MUNICIPALITY relation, there is a **transitive functional dependency between those two attributes.** This violated the rule of 3NF and BCNF.

- MUNICIPALITY_2, GHG_EMISSIONS, MILES_TRAVELED also satisfy 3NF and BCNF because:
  - In MUNICIPALITY_2, the County_Name attribute is fully functionally dependent on the Municipality name and zip code, the same FD goes for Vehicle_GHGs_Emitted, and Total_Miles_Traveled in the GHG_EMISSIONS and MILES_TRAVELED relations, respectively.

CONTAINS



- We saw that the CONTAINS relation was already in 2NF to begin with, as all non-key attributes in this relation have some form of dependency with the **whole primary key.**

- But we have examined where this functional dependency - {Number_of_EVs, Total_Personal_Owned} -> Percent_EVs violates 3NF because it is transitive based on the CONTAINS diagram. Thus we needed to break down the relation into two more relations, CONTAINS_2 and EV_RATIO.

- The EV_RATIO relation removes the transitive functional dependency since the percentage of EVs (Perecent_EVs attribute) will now **only** be determined by the number of EVs, and the total number of personal owned vehicles (became primary keys in its relation); thus also satisfying the BCNF requirement.

- CONTAINS_2 will also be in BCNF because Number_of_EVs & Total_Personal_Owned attributes in this relation will **fully** depend on **all** attributes consisting of the primary key combination of Mun_Name, Zip_Code, and Vehicle_ID.

---

2. Define the different views (virtual tables) required. For each view list the data and transaction requirements. Give a few examples of queries, in English, to illustrate.

- In order to determine the views required, we need to take the use cases into consideration.
    1. Entering GHG emissions
    2. Entering zip code
    3. Entering County and Range of mileages

Example Views to use:

<u>View #1</u>
CREATE VIEW CONTAINS_VEHICLE **AS**
SELECT VEHICLE_2.*, CONTAINS_2.Mun_Name, CONTAINS_2.Zip_code, CONTAINS_2.Number_of_EVs, CONTAINS_2.Total_Personal_Owned
FROM VEHICLE_2 **JOIN** CONTAINS_2
ON VEHICLE_2.Vehicle_ID = CONTAINS_2.Vehicle_ID

<u>View #2</u>
CREATE VIEW MUN_EMISSIONS **AS**
SELECT MUNICIPALITY_2.Zip_Code, MUNICIPALITY_2.Municipality_name, GHG_EMISSIONS.Vehicle_GHGs_Emitted
FROM MUNICIPALITY_2 **JOIN** GHG_EMISSIONS
ON MUNICIPALITY_2.Zip_Code = GHG_EMISSIONS.Zip_Code

<u>View #3 (Utilizes View #2 for another Join operation)</u>
CREATE VIEW MILES_MUN_EMISSIONS **AS**
SELECT MUN_EMISSIONS.*, MILES_TRAVELED.Total_Miles_Traveled
FROM MUN_EMISSIONS **JOIN** MILES_TRAVELED
ON MUN_EMISSIONS.Zip_Code = MILES_TRAVELED.Zip_Code

---

3. Design a complete set of SQL queries to satisfy the transaction requirements identified in the previous stages, using the relational schema and views defined in tasks 2 and 3 above.

- We do not use transactions in our database, since when the database is initially loaded with our data, we will only be making queries to the database (thus read-only operations will be done).
- Therefore, we will not define transaction requirements.