EcoVista: Interactive Environmental Insights Map

Team members: ZiHan Li, YiXuan Li, YaTing Pai, Xuanming Zhang

1. Entity Descriptions and Assumptions

User Profile

- **Description**: Stores user-specific information.
- **Primary Key**: `user_id`
- **Attributes**: `user_id`, `username`, `email`, `county_code`
- **Assumptions**:
    - Each user has a unique `user_id`.
    - Each user belongs to exactly one region, identified by `county_code`.
    - The `county_code` attribute is a foreign key referencing `Location(county_code)`.
- **Discussion**:
    - We use this entity to store user information, especially county_code, which is used to display the user's location data on the website. When a new user registers we update the record, and when a user deletes their account we delete the record from the entity. Users can also change their location.

Location

- **Description**: Contains regional information.
- **Primary Key**: `county_code`
- **Attributes**: `county_code`, `county_name`, `state`
- **Assumptions**:
    - Each `county_code` uniquely identifies a county.
    - Stores general information about the county and state.
- **Discussion**:
    - Location data is modeled as a separate entity rather than an attribute because
        1. Location entities can be reused by multiple other entities to avoid duplicating county_name and state information across multiple different entities.
        2. If county_name or state needs to be updated, we do not need to repeat the change in multiple tables
        3. Modeling location as a separate entity can better normalize the database. If county_name and state are stored as attributes of the Data entities, it will lead to transitive dependencies
    - It is possible that county_name is not unique across states (e.g. there are multiple counties named "Washington County" in different states), which is why county_code uses it as a unique identifier

CO Data

- **Description**: Records Carbon Monoxide (CO) environmental data.
- **Primary Key**: Composite key (`county_code`, `timestamp`)
- **Attributes**: `county_code`, `timestamp`, `co_measurement`
- **Assumptions**:
    - Each record represents CO data for a specific region at a given time.
    - Shares a one-to-one relationship with NO Data, Drought Data, and Air Quality Data based on `county_code` and `timestamp`.
- Discussion:

    - This entity was generated from one of our datasets. Storing CO data separately also helps maintain 3NF.

NO2 Data

- **Description**: Records Nitrogen Oxides (NO) environmental data.
- **Primary Key**: Composite key (`county_code`, `timestamp`)
- **Attributes**: `county_code`, `timestamp`, `no2_measurement`
- **Assumptions**:
    - Each record represents NO data for a specific region at a given time.
    - Shares a one-to-one relationship with CO Data, Drought Data, and Air Quality Data.
- Discussion:
    - This entity was generated from one of our datasets. Storing CO data separately also helps maintain 3NF.

Drought Data

- **Description**: Records drought-related environmental data.
- **Primary Key**: Composite key (`county_code`, `timestamp`)
- **Attributes**: `county_code`, `timestamp`, `drought_level`
- **Assumptions**:
    - Each record represents drought data for a specific region at a given time.
    - Shares a one-to-one relationship with CO Data, NO Data, and Air Quality Data.
- Discussion:
    - This entity was generated from one of our datasets. Storing CO data separately also helps maintain 3NF.

Air Quality Data

- **Description**: Records general air quality data.
- **Primary Key**: Composite key (`county_code`, `timestamp`)
- **Attributes**: `county_code`, `timestamp`, `aqi`
- **Assumptions**:
    - Each record represents air quality data for a specific region at a given time.
    - Shares a one-to-one relationship with CO Data, NO Data, and Drought Data.
- Discussion:
    - This entity was generated from one of our datasets. Storing CO data separately also helps maintain 3NF.

2. Relationships and Cardinalities

UserProfile and Location

- **Type**: One-to-Many
- **Description**: Each user belongs to one location, but a location can have many users.
- **Cardinality**:
    - **UserProfile** (Many) ↔ **Location** (One)

Location and Environmental Data Tables

- **Type**: One-to-Many

- **Description**: Each location can have multiple environmental data records at different timestamps.
- **Cardinality**:
    - **Location** (One) ↔ **COData** (Many)
    - **Location** (One) ↔ **NO2Data** (Many)
    - **Location** (One) ↔ **DroughtData** (Many)
    - **Location** (One) ↔ **AirQualityData** (Many)

Environmental Data Tables Interrelation

- **Type**: One-to-One
- **Description**: For each `county_code` and `timestamp`, the environmental data tables are linked via a one-to-one relationship.
- **Cardinality**:
    - **COData** (One) ↔ **NO2Data** (One)
    - **COData** (One) ↔ **DroughtData** (One)
    - **COData** (One) ↔ **AirQualityData** (One)

3. Normalization Process

For 1NF, 1NF requires that all attributes should have indivisible values and each record should be unique. In our schema, all attributes are indivisible with no repeating groups or multi-valued attributes.

For 2NF, 2NF requires that all non-key attributes must be fully dependent on the primary key and the schema should be 1NF. In our schema, in USER_PROFILE, the primary key is user_id and non-key attributes (username, email, county_code) are fully dependent on user_id. For LOCATION, the primary key is county_code and non-key attributes (country_name, state) are fully dependent on county_code. In CO_DATA, NO2_DATA, DROUGHT_DATA and AIR_DATA,composite primary keys are (county_code, timestamp) and non-key attributes are fully dependent on both county_code and timestamp for each table.

For 3NF, 3NF requires that non-key attributes should not depend on another non-key attribute and the schema should be 2NF. In USER_PROFILE, all non-key attributes directly depend on user_id; In LOCATION, country_name and state are directly depend on county_code; In Environment Data Tables, each table's non-key attributes directly depend on composite primary key (county_code, timestamp).

Thus, we can claim that our database is 3NF.

USER_PROFILE:
user_id -> username, county_code, email
LOCATION:
County_code -> county_name, state
CO_DATA:
County_code, timestamp -> co_measurment
AirQualityData:
County_code, timestamp -> aqi
DROUGHT_DATA:
County_code, timestamp -> drought_level
NO2_DATA:
County_code, timestamp -> no2_measurment

4. Relation Schema

User Profile

UserProfile(user_id: INT [PK], username: VARCHAR(100), email: VARCHAR(100), county_code: INT [FK to Location.county_code])

Location

Location(county_code: INT [PK], county_name: VARCHAR(100), state: VARCHAR(100))

COData

COData(county_code: VARCHAR(10) [PK, FK to Location.county_code], timestamp: VARCHAR(7) [PK], co_measurment: DECIMAL(5,3))

NO2Data

NO2Data(county_code: INT [PK, FK to Location.county_code], timestamp: VARCHAR(7) [PK], no2_measurment: DECIMAL(5,3))

DroughtData

DroughtData(county_code: INT [PK, FK to Location.county_code], timestamp: VARCHAR(7) [PK], drought_level: DECIMAL(5,3))

AirQualityData

AirQualityData(county_code: INT [PK, FK to Location.county_code], timestamp: VARCHAR(7) [PK], aqi: DECIMAL(5,3))