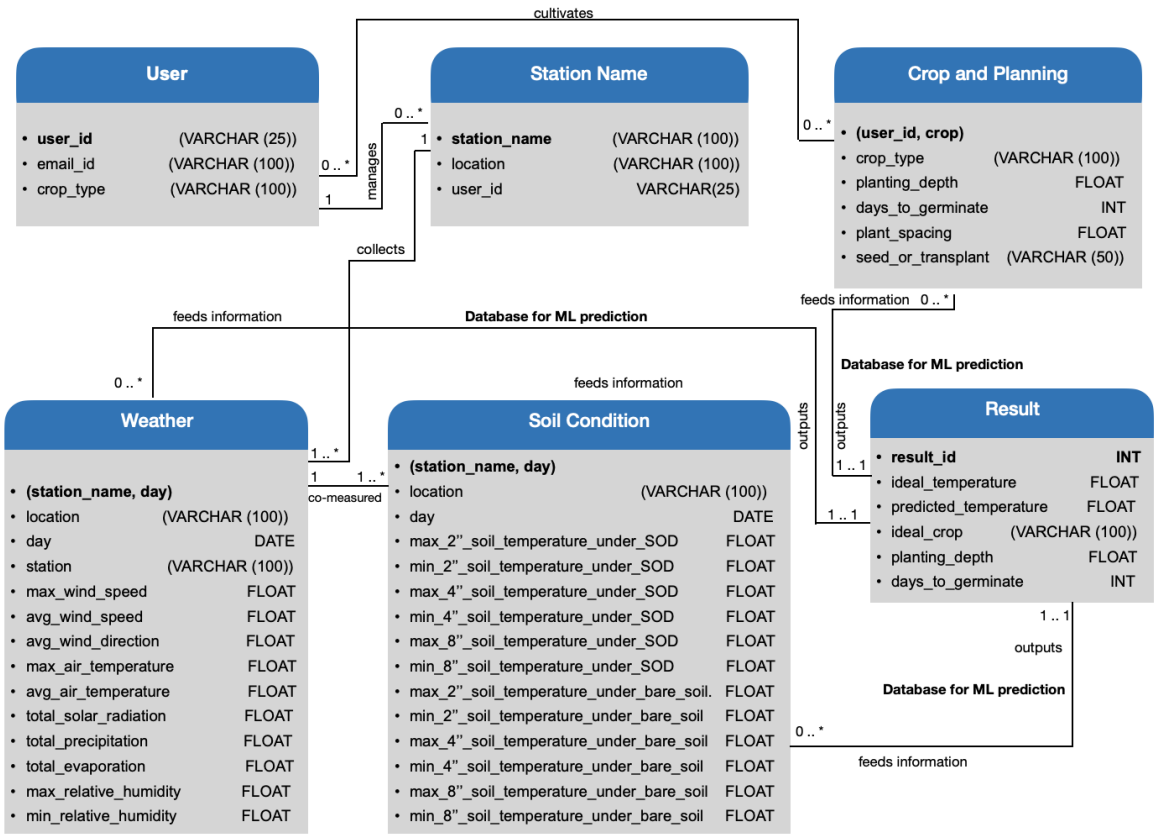
CS 411 Stage 2

March 2025

1 Introduction

This document outlines the data model and database structure for the Smart Agriculture Management Platform for Illinois Farmers. The database schema is designed to store information related to weather, soil conditions, and crop recommendations. We have 6 entities and 1 to many, many to many and many to 1 relationships.

2 UML Diagram



3 Entity Description

User

Attribute	Type
user_id	VARCHAR (25)
email_id	VARCHAR (100)
crop_type	VARCHAR (100)

Station Name

Attribute	Type
station_name	VARCHAR (100)
location	VARCHAR (100)
user_id	VARCHAR (25)

Crop and Planning

Attribute	Type
(user_id, crop)	Primary Key
crop_type	VARCHAR (100)
planting_depth	FLOAT
days_to_germinate	INT
plant_spacing	FLOAT
seed_or_transplant	VARCHAR (50)

Weather

Attribute	Type
(station_name, day)	Primary Key
location	VARCHAR (100)
day	DATE
station	VARCHAR (100)
max_wind_speed	FLOAT
avg_wind_speed	FLOAT
avg_wind_direction	FLOAT
max_air_temperature	FLOAT
avg_air_temperature	FLOAT
total_solar_radiation	FLOAT
total_precipitation	FLOAT
total_evaporation	FLOAT
max_relative_humidity	FLOAT
min_relative_humidity	FLOAT

Soil Condition

Attribute	Type
(station_name, day)	Primary Key
location	VARCHAR (100)
day	DATE
max_2" _soil_temperature_under_SOD	FLOAT
min_2" _soil_temperature_under_SOD	FLOAT
max_4" _soil_temperature_under_SOD	FLOAT
min_4" _soil_temperature_under_SOD	FLOAT
max_8" _soil_temperature_under_SOD	FLOAT
min_8" _soil_temperature_under_SOD	FLOAT
max_2" _soil_temperature_under_bare_soil	FLOAT
min_2" _soil_temperature_under_bare_soil	FLOAT
max_4" _soil_temperature_under_bare_soil	FLOAT
min_4" _soil_temperature_under_bare_soil	FLOAT
max_8" _soil_temperature_under_bare_soil	FLOAT
min_8" _soil_temperature_under_bare_soil	FLOAT

Result

Attribute	Type
result_id	INT
ideal_temperature	FLOAT
predicted_temperature	FLOAT
ideal_crop	VARCHAR (100)
planting_depth	FLOAT
days_to_germinate	INT

4 Cardinality of Relationships

4.1 One-to-Many (1–M) Relationships

- **User – Station:** A `User` can be associated with zero or many `Stations`, but each `Station` belongs to exactly one `User`.
- **Station – Weather:** A `Station` can generate multiple `Weather` records (e.g., for different days), but each `Weather` record is linked to exactly one `Station`.
- **Weather – Soil Condition:** A single `Weather` record may correspond to multiple `Soil Condition` entries, but each `Soil Condition` record is tied to exactly one `Weather` record.

4.2 Many-to-Many (M–M) Relationship

- **User – Crop:** A `User` can plan or grow multiple `Crops`, and each `Crop` can be grown by multiple `Users`.

4.3 Many-to-One (M–1) Relationship

- **Weather, Soil Condition, and Crop Planning – Result (Prediction-Based Relationship):** Multiple weather observations, soil condition records, and crop planning attributes serve as inputs to the ML prediction model. The model analyzes historical and real-time data to generate an optimal planting strategy and predict agricultural outcomes. Each prediction run produces exactly one result, but multiple results can exist over time.

5 Assumptions and Justifications

This section explains the assumptions and design choices for the UML diagram used in our system. We describe why certain concepts were modeled as entities rather than attributes and clarify the cardinality of relationships.

5.0.1 User

Assumption: A user represents an individual interacting with the system, providing input regarding crop planning and monitoring weather conditions.

Justification: The `User` is modeled as an entity because multiple users may interact with the platform, each having unique preferences and associated stations. The table holds key attributes such as `user_id` and `email_id`. Storing users in a dedicated entity allows for clear ownership of `Station` records and potential many-to-many relationships with `Crop`.

5.0.2 Station

Assumption: A station collects meteorological data, serving as a central point for weather measurements.

Justification: The `Station` is an entity rather than an attribute because multiple stations exist across different locations, each uniquely identified (e.g., by `station_name` or an ID). This design supports normalization and makes it easy to link to `Weather` data.

5.0.3 Weather

Assumption: Weather conditions (e.g., temperature, wind speed, humidity) influence crop growth and should be tracked per station and time (or day).

Justification: `Weather` is separated into its own entity because it has multiple attributes that vary independently (such as `max_wind_speed`, `avg_wind_speed`, `temperature`, and `humidity`). Storing these attributes in a dedicated table prevents data redundancy and simplifies querying by time or station.

5.0.4 Soil Condition

Assumption: Soil conditions, including various temperature or moisture readings, are critical for predicting crop health and must be monitored alongside weather data.

Justification: `SoilCondition` is modeled as its own entity because soil data may fluctuate over time and is distinct from meteorological measurements. In this design, each `SoilCondition` record is associated with a single `Weather` record, reflecting the conditions observed at the time of that weather reading.

5.0.5 Crop and Planning

Assumption: Different crops require distinct parameters like planting depth, germination days, and spacing.

Justification: `Crop and Planning` is an entity so that each `crop_type` can be stored with its unique cultivation requirements. This structure facilitates efficient querying, supports linking to multiple users, and provides a foundation for generating crop-specific predictions.

5.0.6 Result

Assumption: The system must store outcomes such as `ideal_temperature`, `predicted_temperature`, and crop recommendations.

Justification: The `Result` entity serves as the final output of the predictive process, linking insights derived from `Weather`, `Soil Condition`, and `Crop Planning`. By assigning a unique `result_id` to each prediction, the system can store multiple prediction runs over time and reliably retrieve or compare specific recommendations. Keeping `Result` separate from other tables ensures that predictions remain modular and can be refined or reused as new data becomes available, without altering the underlying data sources.

6 BCNF Normalization

6.1 Functional Dependencies

Based on the entity relationships and cardinality described in the diagram, we can identify the following functional dependencies:

- user_id → email_id, crop_type (User entity)
- station_name → location, user_id (Station entity)
- (station_name, day) → all weather attributes (Weather entity)
- (station_name, day) → all soil condition attributes (Soil Condition entity)
- (user_id, crop) → crop_type, planting_depth, days_to_germinate, plant_spacing, seed_or_transplant (Crop and Planning entity)
- result_id → ideal_temperature, predicted_temperature, ideal_crop, planting_depth, days_to_germinate (Result entity)

6.2 Normalization Analysis

Our database schema is designed to be in Boyce-Codd Normal Form (BCNF) by construction. Let's analyze why:

- **User Entity:** The primary key user_id uniquely determines all other attributes (email_id, crop_type). No non-key attributes determine other attributes.
- **Station Entity:** The primary key station_name uniquely determines the location and user_id. The 1-to-many relationship from User to Station ensures no non-key attributes determine other attributes.
- **Weather Entity:** The composite primary key (station_name, day) uniquely determines all weather measurements. Each weather record is tied to exactly one station on a specific day, preventing any non-key dependencies.
- **Soil Condition Entity:** The composite primary key (station_name, day) uniquely determines all soil measurements. The co-measured relationship with Weather ensures proper normalization.
- **Crop and Planning Entity:** The composite primary key (user_id, crop) uniquely determines all crop planning attributes. The many-to-many relationship between User and "Crop and Planning" is properly represented.
- **Result Entity:** The primary key result_id uniquely determines all result attributes. The relationship with the ML prediction database ensures no non-key dependencies.

6.3 Justification for BCNF

Our schema satisfies BCNF because:

1. Every functional dependency $X \rightarrow Y$ has X as a superkey
2. No non-key attributes determine other attributes
3. All relationships are properly normalized with appropriate cardinality constraints:
   - One-to-many relationships (User to Station)
   - Many-to-many relationships (User to Crop and Planning)
   - Co-measured relationships (Weather and Soil Condition)
4. Each entity has a clear primary key (single or composite) that uniquely identifies all its attributes

The design avoids common normalization issues such as:

- Transitive dependencies (no $A \to B \to C$ where A is not a key)
- Partial dependencies (all non-key attributes depend on the entire primary key)
- Multi-valued dependencies (handled through proper relationship modeling)

The database schema properly represents the data collection and prediction system, with clear separation of concerns between user management, station data, environmental measurements, crop planning, and prediction results.

7 Relational Schema

Below is the relational schema derived from the UML diagram. For composite primary keys, each attribute participating in the key is labeled "[PK]." For foreign keys, each column includes "[FK to Table.Column]."

```
User(
    user_id  : VARCHAR(25)   [PK],
    email_id : VARCHAR(100),
    crop_type: VARCHAR(100)
)

Station(
    station_name : VARCHAR(100) [PK],
    location     : VARCHAR(100),
    user_id      : VARCHAR(25)  [FK to User.user_id]
)

CropAndPlanning(
    user_id          : VARCHAR(25)   [FK to User.user_id, PK],
    crop             : VARCHAR(100)  [PK],
    crop_type        : VARCHAR(100),
    planting_depth   : FLOAT,
    days_to_germinate: INT,
    plant_spacing    : FLOAT,
    seed_or_transplant: VARCHAR(50)
)

Weather(
    station_name       : VARCHAR(100) [FK to Station.station_name, PK],
    day                : DATE         [PK],
    location           : VARCHAR(100),
    station            : VARCHAR(100),
    max_wind_speed     : FLOAT,
    avg_wind_speed     : FLOAT,
    avg_wind_direction : FLOAT,
    max_air_temperature: FLOAT,
    avg_air_temperature: FLOAT,
    total_solar_radiation : FLOAT,
    total_precipitation   : FLOAT,
    total_evaporation     : FLOAT,
    max_relative_humidity : FLOAT,
    min_relative_humidity : FLOAT
)

SoilCondition(
    station_name : VARCHAR(100) [FK to Weather.station_name, PK],
    day          : DATE         [FK to Weather.day, PK],
    location     : VARCHAR(100),
```

```
    max_2in_soil_temperature_under_SOD         : FLOAT,
    min_2in_soil_temperature_under_SOD         : FLOAT,
    max_4in_soil_temperature_under_SOD         : FLOAT,
    min_4in_soil_temperature_under_SOD         : FLOAT,
    max_8in_soil_temperature_under_SOD         : FLOAT,
    min_8in_soil_temperature_under_SOD         : FLOAT,
    max_2in_soil_temperature_under_bare_soil   : FLOAT,
    min_2in_soil_temperature_under_bare_soil   : FLOAT,
    max_4in_soil_temperature_under_bare_soil   : FLOAT,
    min_4in_soil_temperature_under_bare_soil   : FLOAT,
    max_8in_soil_temperature_under_bare_soil   : FLOAT,
    min_8in_soil_temperature_under_bare_soil   : FLOAT
)

Result(
    result_id            : INT         [PK],
    ideal_temperature    : FLOAT,
    predicted_temperature: FLOAT,
    ideal_crop           : VARCHAR(100),
    planting_depth       : FLOAT,
    days_to_germinate    : INT
)
```

Notes:

- For the `Weather` and `SoilCondition` tables, the primary key is the composite (`station_name`, `day`).
- `Station.station_name` is a primary key in `Station` and a foreign key for `Weather` and `SoilCondition`.
- `Result` has a synthetic key (`result_id`) for uniqueness.