Theodore “TJ” Norred
David McGee
Project Title: Model Managers
Project URL: http://classwork.engr.oregonstate.edu:3797/

Project 4 DRAFT:

1. Feedback Received
    a. From TA: “Appears to mostly be generated from dump. Cleanly imports.” Received from Alex Lovato
    b. From TA: “There are some issues with the .html pages provided here that are possibly cleared up by mentioning that the app you're creating should be one for the administrators of the database which The Model-Managers will use for their app. So The Model-Managers will have their app which people can use to compare Forecasts to Readings and such, but the app you're building should just provide an interface which database admins can use to perform CRUD operations on each table from that database. Most groups accomplish this by dedicating each page to a single table and its associated operations but you could technically not have 1:1 correspondence. What's important is that there be someplace to view the records from each table and that you satisfy requirements for CRUD operations as detailed by the CS 340 Project Guide.” Received from Alex Lovato
    c. From Student Amin Hussien:

“Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.

While the UI does have a table for the sensor and location table. It is missing a select for Models and forecasts.

- *Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?*

Currently, no select utilizes a filter with dynamic properties. Although I can see the intent to implement that on the Reading and Forecast page.

- *Does the UI implement an INSERT for every table in the schema?* In other words, there should be UI input fields that correspond to each table and attribute in that table.

The UI implements an insert for the sensors and location tables but no other tables.

- *Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?* In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection

The sensor insert does add the corresponding fk attributes.

- *Is there at least one DELETE, and does at least one DELETE remove things from a M:M relationship?* In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

Currently no DELETE.

- *Is there at least one UPDATE for any one entity?* In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

Currently no UPDATE.

- *Is at least one relationship NULLable?* In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus, it should be feasible to edit an Order and change the value of Employee to be empty.

None of the current relationships are nullable.

- *Do you have any other suggestions for the team to help with their HTML UI? For example, using AS aliases to replace obscure column names such as fname with First Name.*

I like how well-organized your UI is once you implement INSERT, DELETE and SELECT for the other entities it will be in a very good spot."

2. Fixes based on the received feedback above:
    a. Updated the DDL to mirror the APIs used to generate forecasts and readings data.
    b. Added an individual html web page for each entity:
        i. Forecasts.html
        ii. Readings.html
        iii. Sensors.html
        iv. Models.html
        v. Dates.html
        vi. Locations.html
    c. Updated the base.j2 file to make the nav bar representative of these pages.
    d. Broke out SQL queries for each page/entity versus consolidating them into a library page.
    e. Made Models nullable
3. Changes based on own improvements
    a. DLL Changes:
        i. Modified DLL to better mirror the open-meteo API (removed forecastWindGust)
        ii. Updated the Schema and image below to correspond to the attribute removal.

b. Added add/forecasts and add/readings routes
    i. Connected forecasts openmeteo API
    ii. Connected readings to holfuy api
c. Developed the results.html page and display of data.
d. Added css buttons for edit and delete functionality
e. Added better css formatting via the pico-css library
f. Updated the HTML paragraphs to provide better clarity / included hrefs versus text URLs.
g. Modified the DDL to make the `forecastModelID` foreign key nullable
    i. Modified the Entities list below to correspond

Model-Managers Project Proposal

- **Overview:**

The Model-Managers company enables customers to determine which wind forecast models are most accurate for given locations over specified time intervals. There are a variety of wind forecast models (HRRR, ECMWF, GFS, MBLUE, etc.) which often have different wind forecasts for a given location. Users of these forecast models usually cannot determine what model is the most accurate for their specific location. In other words, they would need to compare the model forecasts to actual data from weather/wind collection devices at specific locations to determine which model is typically more accurate for those locations.

The Model-Managers company solves this problem by storing hourly wind forecasts for a standardized forecast period of 7 days in a database, and then allowing users to interact with a web interface to compare the forecast data against the actual measured data at a location for different time intervals. The 3 criteria used to compare forecast predictions to measured wind are average wind speed, wind direction, and wind gust speed. The Model-Managers company uses the free OpenMeteo API (https://open-meteo.com) to obtain wind forecast data, and a Holfuy (https://api.holfuy.com/) device API to obtain the measured wind forecast data.

The minimal viable product of the Model-Managers company uses 2 weather forecast models (HRRR and ECMWF) and 1 Holfuy device, located at La Bajada Ridge, New Mexico. The Model-Managers interface is expected to be accessible by a minimum of 50 different users over 100 times per day. Model-Managers uses a GitHub repository at https://github.com/skipmcgee/wind-forecast to manage their codebase.

The entities that the Model-Manager's Database uses are Models, Locations, Sensors, Forecasts, Dates, and Readings. There will be more instances of Dates, Readings and Forecasts than any other entity, as we are limiting the MVP to one Sensor and 2 Models. Since there is one Sensor, that Sensor will be at 1 location for the MVP phase of the project. The Dates of the Readings and Forecasts will be taken hourly, so 24 times a day for approximately

the next four months (each). This is anticipated to be around 2100 instances of each of the Dates, Forecasts and Readings entities.

- **Entities:**

☐ Models:
These are the individual weather models that the Model-Managers product supports. These weather models are the model that drives a specific forecast and include HRRR, ECMWF, etc models. This entity's purpose is to capture the weather model specifics.
   a. modelID: int, auto_increment, unique, not NULL, PK
   b. modelName: varchar(100), not NULL

Relationships: there is a one to many relationship between models and forecasts. For every model there are many forecasts. Forecasts are the intersection table(s) of the M:N or many to many relationship between Models and Locations.

☐ Locations:
The locations where customers desire to compare forecasts to measured values. This entity's purpose is to manage the location information such as latitude, longitude, and altitude.
   a. locationID: int, auto_increment, unique, not NULL, PK
   b. locationName: varchar(300), unique, not NULL
   c. locationLatitude: float, not NULL
   d. locationLongitude: float, not NULL
   e. locationAltitude: int, not NULL

Relationships: A one to one relationship exists between locations and sensors. There is only one sensor at a specific location. A one to many relationship exists between locations and forecasts. There are typically greater than one forecast at a single location. Forecasts are the intersection table(s) of the M:N or many to many relationship between Locations and Models.

☐ Sensors:
The sensors that can measure the weather data at a specific location. These are specifically Holfuy sensors for the MVP but the attributes are intentionally generalized enough that these could be expanded to include other makes and models of sensor. The purpose of this entity is to manage the attributes of the specific sensors, This includes specific attributes of a sensor, such as its manufacturing number and the api key used to access it, as well as references to its location.
   a. sensorID: int, auto_increment, unique, not NULL, PK
   b. sensorName: varchar(300), not NULL
   c. sensorLocationID: int, not NULL, FK, -> locationID
   d. sensorAPIKey: varchar(300), not NULL
   e. sensorNumber: int, not NULL

Relationships: there is a one to one relationship between sensors and locations.

☐ Forecasts:

Intersection Table for hourly forecast information that is common across all weather models. The purpose of this entity is to hold the forecast prediction information for the weather at a specific date and time.

a. forecastID: int, auto_increment, unique, not NULL,
b. forecastMadeDateID: int, not NULL, FK -> dateID
c. forecastForDateTime: DATETIME, not NULL
d. forecastTemperature2m: float, not NULL
e. forecastPrecipitation: float, not NULL
f. forecastWeatherCode: varchar(100), not NULL
g. forecastPressureMSL: int, not NULL
h. forecastWindSpeed10m: float, not NULL
i. forecastWindDirection10m: float, not NULL
j. forecastCape: float, not NULL
k. forecastLocationID: int, not NULL, FK -> locationID
l. forecastModelID: int, FK -> modelID

Relationships: Forecasts in an intersection table. There is a many to one relationship between forecasts and locations. There are many forecasts at a single location. There is a many to one relationship between forecasts and models. There are many forecasts for a single model. There is a many to one relationship for forecasts to dates. There are many forecasts at a date.

☐ Readings:

The actual measured sensor readings at a specific location. The purpose of this entity is to hold the reading information that a sensor provides at a specific date and time.

a. readingID: int, auto_increment, unique, not NULL, PK
b. readingSensorID: int, not NULL, FK -> sensorID
c. readingDateTime: int, not NULL, FK -> dateID
d. readingWindSpeed: float, not NULL
e. readingWindGust: float, not NULL
f. readingWindMin: float, not NULL
g. readingDirection: int, not NULL
h. readingTemperature: float, not NULL

Relationships: there is a many to one relationship between readings and sensors. There are many readings at a single sensor. There is a many to one relationship between readings and dates. There are many readings at a single date.

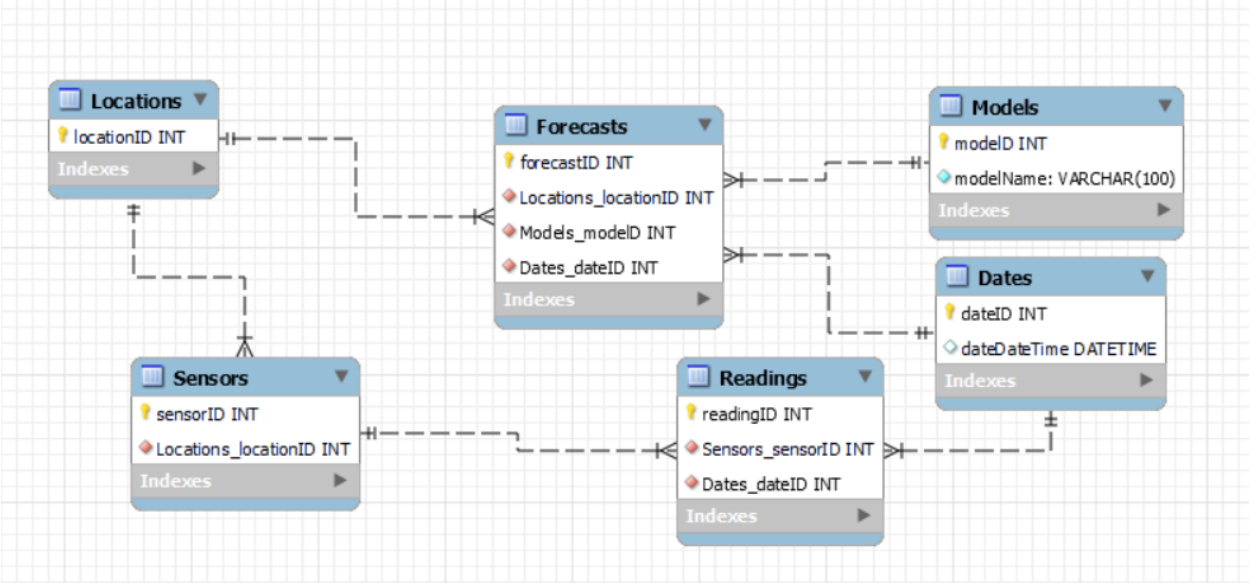
☐ Dates:

The date and time. The purpose of this entity is to hold the date information and to faciliate access to a specific forecast and/or a specific reading at a date and time.

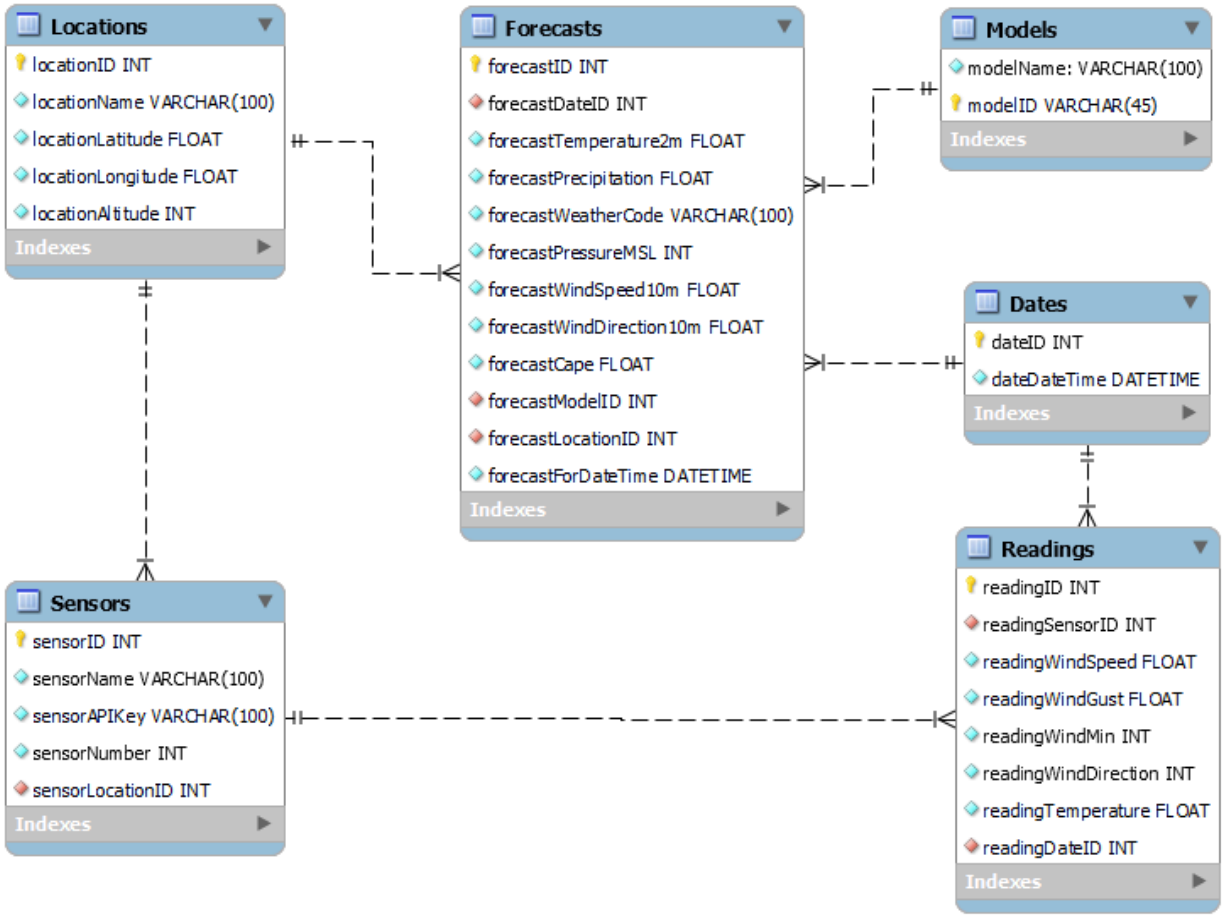
a. dateID: int, auto_increment, unique, not NULL, PK
b. dateDateTime: DateTime, not NULL,

Relationships: there is a many to one relationship between readings and dates. There are many readings at a single datetime. There is a many to one relationship between forecasts and dates. There and many forecasts for a single datetime.

- **Updated Entity Relationship Diagram:**



d) Schema:



e) Example Data:

```
Models Data:
('ECMWF'),
('HRRR'),
('GFS'),
('ICON'),
('MBLUE'),
('NAM');

Locations Data:
```

```
('La Bajada Ridge Launch', 35.56195,
-106.22596, 6135),
('Sandia Peak Launch', 35.196576, -106.434662,
10275),
('Sandia Crest Launch', 35.21342, -106.45026,
10600),
('Blue Springs Launch', 34.44002, -106.51913,
6200);

Sensors Data:
('La Bajada Holfuy', 1, 'mytestapikey1234',
1151),
('Sandia Peak Holfuy', 2, 'mytestapikey23456',
1152),
('Sandia Peak Tempest', 2,
'mytestapikey345670', 2),
('Sandia Crest Holfuy', 3,
'mytestapikey456789', 1153),
('Blue Springs Holfuy', 4, 'mytestapikey56789',
1154);

Dates Data:
('2024-04-01 08:00:00'),
('2024-04-01 09:00:00'),
('2024-04-01 10:00:00'),
```

```
('2024-04-01 11:00:00'),
('2024-04-01 12:00:00'),
('2024-04-01 13:00:00'),
('2024-04-01 14:00:00'),
('2024-04-01 15:00:00'),
('2024-04-01 16:00:00'),
('2024-04-01 17:00:00'),
('2024-04-01 18:00:00'),
('2024-04-01 19:00:00'),
('2024-04-01 20:00:00'),
('2024-04-01 21:00:00');

Readings Data:
(1, 1, 22.0, 28.0, 14.0, 250, 64),
(1, 1, 22.0, 28.0, 14.0, 250, 64),
(1, 2, 20.0, 22.0, 6.0, 231, 66),
(1, 3, 21.0, 24.0, 11.0, 242, 67),
(1, 4, 23.0, 25.0, 8.0, 250, 69),
(1, 5, 18.0, 21.0, 4.0, 256, 71),
(1, 6, 17.0, 22.0, 12.0, 264, 74),
(1, 7, 16.0, 20.0, 15.0, 270, 72),
(1, 8, 18.0, 21.0, 18.0, 265, 77),
(1, 9, 20.0, 24.0, 16.0, 254, 75),
(1, 10, 19.0, 22.0, 17.0, 246, 74);
```

```
Forecasts Data:
('2024-04-01 15:00:00', 1, 58, 0, 'CLEAR', 3,
10.0, 14.7, 220.0, 3.5, 1, 1),
('2024-04-01 16:00:00', 2, 58, 0, 'CLEAR', 3,
10.0, 14.7, 220.0, 4.1, 1, 1),
('2024-04-01 17:00:00', 3, 62, 0, 'CLEAR', 4,
14.0, 24.0, 230.0, 4.2, 1, 1),
('2024-04-01 18:00:00', 4, 64, 0, 'CLEAR', 4,
16.0, 25.2, 232.0, 3.6, 1, 1),
('2024-04-01 19:00:00', 5, 66, 0, 'CLEAR', 4,
13.0, 25.0, 228.0, 3.4, 1, 1),
('2024-04-01 20:00:00', 6, 62, 0, 'CLEAR', 4,
11.0, 24.0, 220.0, 3.3, 1, 1),
('2024-04-01 21:00:00', 7, 58, 0, 'CLEAR', 3,
9.0, 20.0, 218.0, 3.1, 1, 1);
```