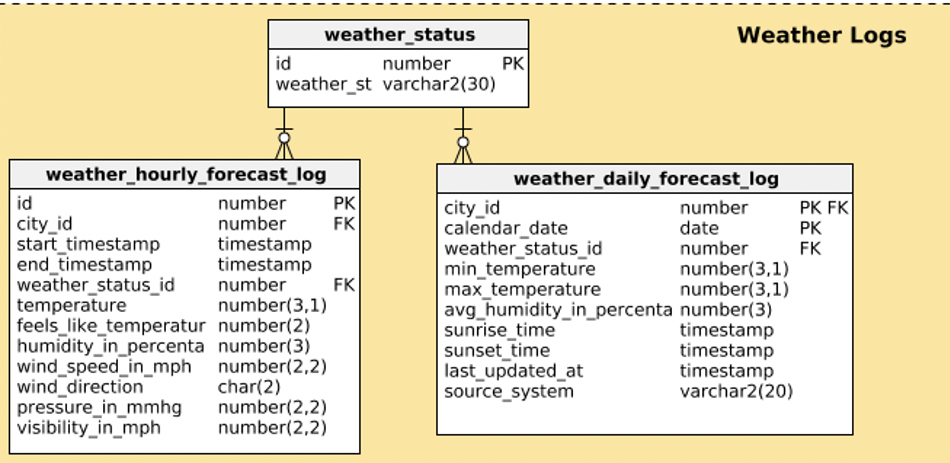


I’ve split the model into three subject areas:

1. Weather Logs
2. User Preferences
3. User Profiles

Weather Logs



This is the most important subject area. Any weather app should capture these basic details:

* Current actual temperature
* Current “feels like” temperature, which may be different than the actual temperature because of additional weather factors (e.g. high humidity can make a hot day feel hotter or a cold day feel colder).
* Daily high and low temperatures
* Dew point and/or relative humidity data
* Wind speed
* Wind direction
* Barometric pressure
* Visibility (i.e. a foggy day will have lower visibility than a clear day)
* Sunrise and sunset times

Together, these give a holistic view of current weather condition. This is the information that will be presented to users, usually through one or more intuitive screens.

There are two kinds of attributes to any weather forecast: ones that change each day and ones that change *throughout* each day. Attributes like sunrise and sunset times are based on events that happen once a day, so this information is captured once for each day. When it comes to long-range forecasts (from 7 to 15 days in advance) users should have enough info if you include each day’s high and low temperatures, humidity level, and overall weather conditions (i.e. sunny, cloudy, etc.).

Attributes like current temperature, “feels like” temperature, wind speed and direction, barometric pressure, and visibility range can change throughout a day. These should be captured for a specific time interval, say every hour or every three hours. For the purposes of this model, we’ll assume a one-hour timeframe.

Because we have two types of attributes, I’ve put two tables in this subject area. The first, weather\_daily\_forecast\_log, holds the daily attributes. It contains these columns:

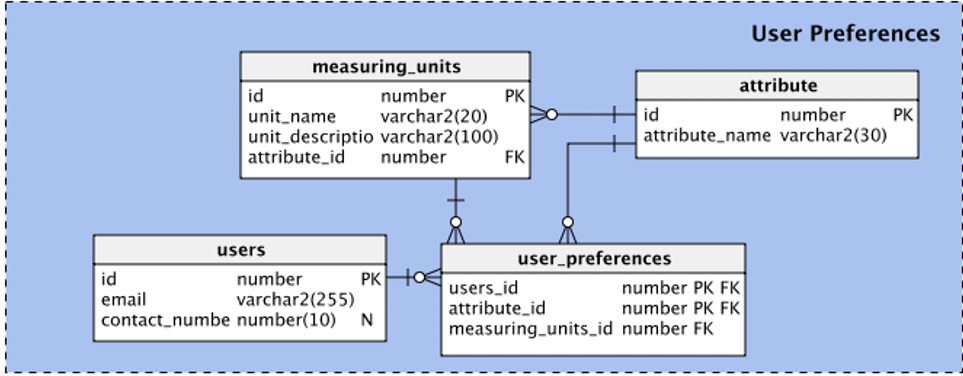
* city\_id – References the city table and signifies the city to which this data applies.
* calendar\_date – The calendar date for this data. Since this table holds one record per city per date, these columns (city\_id and calendar\_date) form the composite primary key for this table.
* weather\_status\_id – References the weather\_status table and denotes the weather condition (i.e. rainy, cloudy, partly cloudy, or sunny).
* min\_temperature – That day’s minimum (lowest) temperature.
* max\_temperature – That day’s maximum (highest) temperature.
* avg\_humidity\_in\_percentage – The *average* relative humidity in the air that day. (The amount of water that air can hold is relative to its temperature.)
* sunrise\_time – A timestamp column that stores the sunrise time.
* sunset\_time – A timestamp column that stores the sunset time.
* last\_updated\_at – Holds the date and time (as a timestamp) when the record was last updated.
* source\_system – The name of the source of our weather forecast. These last two columns are kept for audit purposes.

The weather\_hourly\_forecast\_log table holds all the attributes that can change throughout the day. We consider these attributes as one record for a specific timeframe. The columns are:

* id – The surrogate key for the table.
* city\_id – The relevant city.
* start\_timestamp – A timestamp column that signifies when this timeframe started.
* end\_timestamp – A timestamp column that signifies when this timeframe ended.
* weather\_status\_id – The overall weather status for the timeframe.
* temperature – The current temperature for the timeframe.
* feels\_like\_temperature – The “feels-like” temperature for the timeframe. This can be influenced by many factors, including wind, rain, and high or low humidity. This information gives a more realistic impression of current weather conditions.
* humidity\_in\_percentage –This column holds the amount (as a percentage) of humidity in the air.
* wind\_speed\_in\_mph – Holds the wind speed in mph (miles per hour).
* wind\_direction – This text column stores one or two characters that denote wind direction (N, NW, NE, S, W, SW, etc.)
* pressure\_in\_mmhg – Stores air pressure values, in mmHg.
* visibility\_in\_mph – Stores visibility range values, in miles.

These tables will hold the latest data for a particular timeframe. Occasionally, a future forecast can be issued and then later changed. In such cases, the existing record for the relevant day or timeframe will be overwritten by the latest one. Also, you’ll notice that we’ve only stored attributes in a one measurement unit (e.g. mph) per attribute. To save on storage, we will only store one record for each attribute and let the front end convert these to the user’s preferred units when necessary.

User Preferences



This subject area mainly handles user preferences for measuring units. Most of the columns are self-explanatory, so we will just briefly explain the purpose of each table.

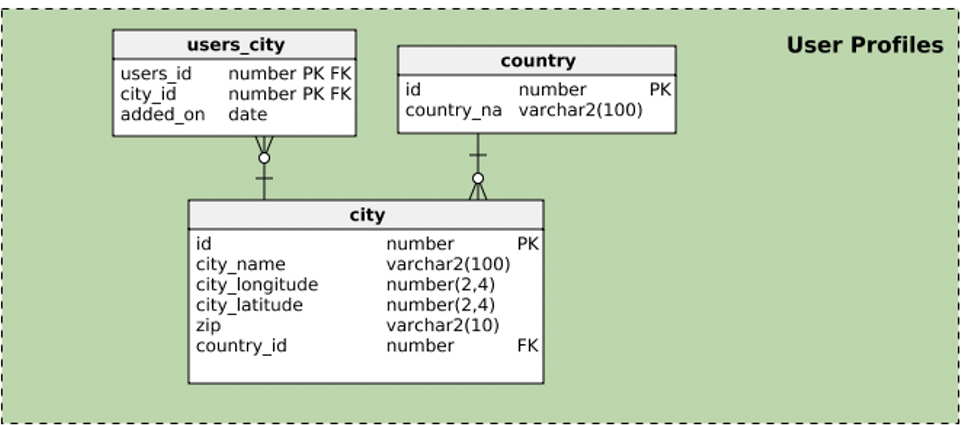
The users table holds basic info about users, like email address and phone number. The id column assigns a unique number to every user who registers with the application.

The attribute table stores a list of attributes, like temperature, wind speed, wind direction, barometric pressure, etc.

The measuring\_units table stores a list of all measurement units, with their corresponding name, description, and attribute\_id.

The user\_preferences table maps the relationship between users and measuring unit preferences. Note that we can store information about users’ preferences for each individual attribute. Since users can choose any one measuring unit out of the given options for an attribute, we have created a composite primary key using the users\_idand attribute\_id columns.

User Profiles



Since the application allows users to monitor the weather in as many cities as they want, this subject area handles associating one or more cities with each user’s profile.

The city table store a list of cities and their location details (postal code, country, map coordinates). The columns in this table are self-explanatory, but it’s good to realize that the city\_longitude and city\_latitude columns can hold positive or negative values.

The user\_city table associates cities with user profiles. Since users can add a city to their profiles only once, we have created a composite primary key using the users\_idand city\_id columns.