# Database overview:

A screenshot of a computer

Description automatically generated

# Site table:

Simply contains a name and primary key idSite so far. Primary key is an unsigned autoincrement tinyint, which means that currently number of national trust sites are limited to 255.

# Sensor table:

Stores both the soil and tinytag sensors, contains 4 fields:

* **IDsensor** stores the primary key. Autoincremented unsigned tinyint, which means number of sensors limited to 255. Currently there are only 24 active sensor and ~12 more when Trelissick sensors are active so for the immediate future this won’t be an issue, but if there are future plans for even more sensors then I would recommend increasing from tinyInt to MediumInt
* **Name** stores name as a varchar, 45 characters is currently a bit excessive and reducing the size may be a small storage optimization.
* **IdSite** is a foreign key linking sensors to site.
* **serialNumber** stores the serial number of a sensor. This can be empty if the serial number is unknown.

# Data table:

Stores raw data from the TinyTag:

* **ID** is the autoincrement primary key.
* **IdSensor** is a foreign key linking data to sensor. Cannot be null.
* **Time** stores the date and time of the record. Cannot be null and forms a composite unique key with idSensor.
* **Temperature, Humidity, and DewPoint** store the raw data from the tiny tag sensors. They are all decimals with the same precision as the data in the csv files.
* **TemperatureValidity, HumidityValidity, and DewPointValidity** store a Boolean 1 or 0 indicating whether they are errors or outliers. Currently they are generated stored columns based on simple metrics.

# Sensor Location History:

Stores the coordinates of sensors and the period of time they were in that position

* **idsensor\_location\_history** is the autoincrement primary key.
* **idSensor** is a foreign key linking to the sensors table
* **StartDate** store date and time sensor moved to this location
* **EndDate** store date and time sensor moved away from location
* **Longitude and Latitude** stores the coordinates.

# Site Sensor Data table:

Stores raw data from the soil sensors:

* **idSiteSensorData** is the autoincrement primary key.
* **Time** stores the date and time of the record. Cannot be null and forms a composite unique key with idSensor.
* **idSensor** is a foreign key linking data to sensor. Cannot be null.
* **theta, temperature, rainfall, power** are columns corresponding to the raw data sent. Each row is a decimal with the exact precision in the csv file.

# Future database improvements:

Both alternate “clean data” tables (time-match and aggregate) are both inactive and redundant, so a possible next step would be to update the tables and get them running again or remove them from the database completely. Time Match clean data used to work when validity used to be 1 column but no longer does so after each column got their own validity columns, and aggregate was shelved early in the project.

Site Sensor Data could potentially have validity columns added.

Validity for data is based on quite simple metrics but they could be expanded to use statistical methods. If they are not going to be expanded reducing validities to be virtual may be a massive storage optimization.

Site Sensor Data stores the power levels of the sensor and a new feature that emails someone when the power level gets too low.

There are a few optimizations with the size of fields which could be adjusted. I’m not sure any of the name fields need to be 45 characters, and the date times in location history could potentially just be dates.

Not necessarily a database change, but finding a sensible way to make use of era5 and other wide area climate data would be a good addition.

# Reflections:

So, as I’ve said before I really enjoyed learning from the gardeners when visiting the site. Before this internship I’ve only had an extremely casual experience with gardening, and it was interesting to hear how they managed disease and risk and all sorts of other problems I didn’t realise they had to consider. Hearing how they intended to use the data also influenced me to make change some of how the database was structured, and more definitely understand the overarching point of the project. To that end, I do think visiting the sites much earlier would’ve been extremely helpful to me. As much as the zoom meetings did provide some context for the project, it became much easier visualise how the data would be used after seeing the site. It also helped me understand what I wanted to know more about as I early on I was stuck in a bit of a didn’t know enough to find out what I needed to know situation.

I know this wasn’t an intentional but some the sensors having issues was another really interesting bit. The combined act of noticing there was something weird with the data, graphing in order to confirm there was a definitive issue and attempting to understand/solve the problem was a practical real world experience you can’t get from the lecture hall.