Time Series Database Design

- We are going to have separate databases and tables for our data. One for video stream records for both /data and /archive where we will have the camera_name along with the timestamp and the calculated fields and other values. This database will be synced with the video records of /data as well as /archive.

- Another one will be sensor data which we will be receiving from various IoT sensors with MQTT (or any) with OPC UA standard. This will have fields like timestamp (in unixtime or if there are other standards defined in OPC UA) along with the sensor_name, sensor_value(s) and then the processed/calculated values and other fields.

- And finally, the Events database. this crucial database will contain the information which was responsible for cobbles and bent billets of any issues that were caused. Along with the other events that are important or that we defined. This will have fields like event_name, event_type, event_location, event_time, and event_status along and the sensor data linked with that event and that specific time.

- For data visualisation, we will use C from tiCk i.e., Chronograf which allows us to quickly see the data that you have stored in InfluxDB. It will be hosted on the server locally. Although I may make a custom dashboard because we can assume that the user will input various query combinations that can't be done in Chronograf I'm guessing also the video database which is stored along with the video file as a reference needs to be browsable, so probably have to use regex also.

- And for email alerts, we have several options like SendGrid, Amazon Simple Email Service (SES), Mailjet, or Mailgun. All of these have good documentation and can easily be configured along with our events and databases. I'll also look for SMS alerts.

- All these are initial blueprints and will change in future as I still need to know a lot about those sensors and the kind of data that we will receive from them, which is OPC UA and probably MQTT or any other IoT protocol. And also, I have to learn more about the TICK Stack.

Some references and links

- https://github.com/influxdata/telegraf/blob/master/plugins/inputs/opcua/README.md
- https://www.influxdata.com/blog/introduction-to-influxdatas-influxdb-and-tick-stack/
- https://opcfoundation.org/about/opc-technologies/opc-ua/
- https://github.com/influxdata/telegraf/tree/master/plugins/inputs/mqtt_consumer

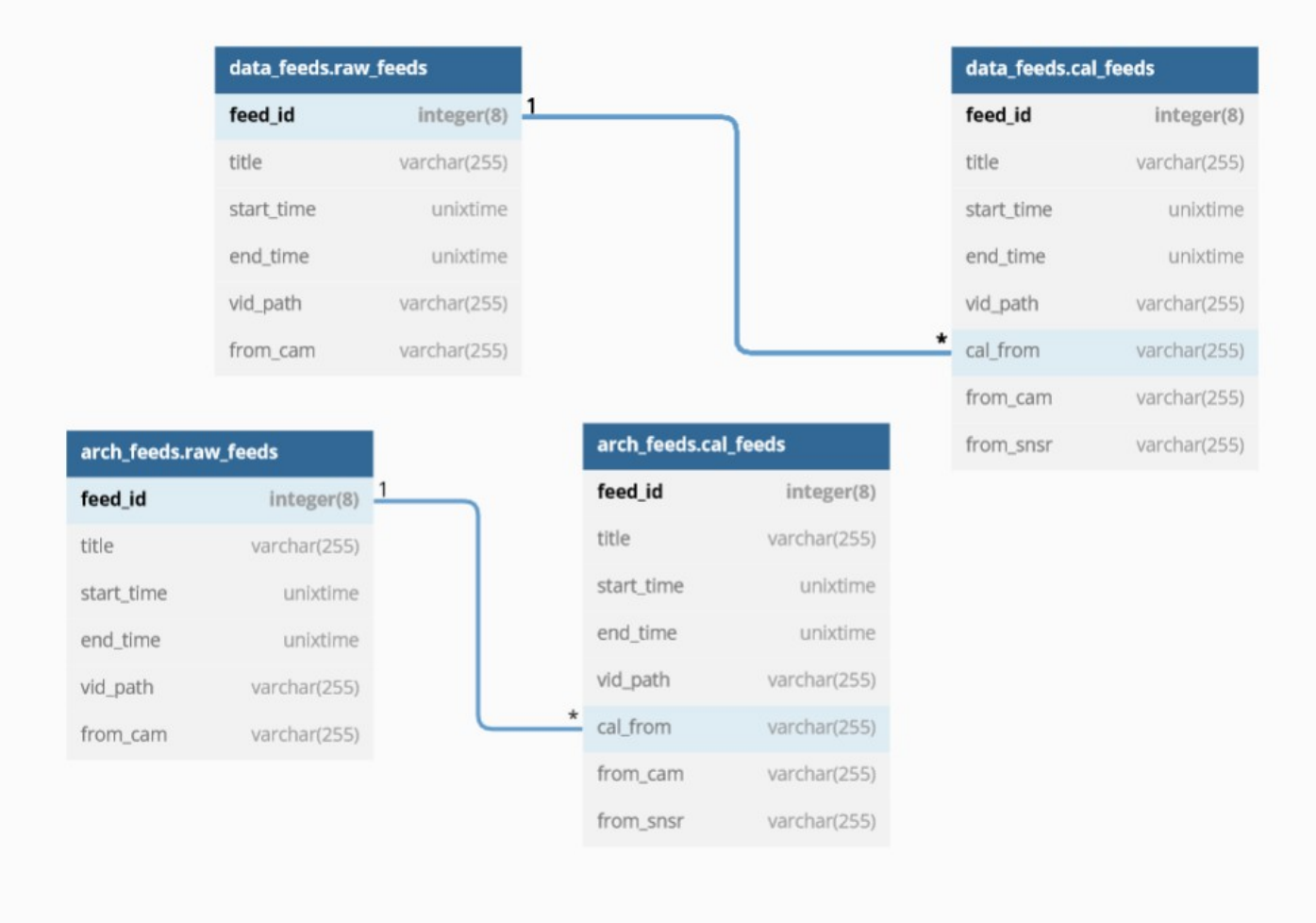
Database and table schema for /data/* and /archive/*

Note: This is the basic SQL-like schema and this is not to be used as influx DB has their standards and scheme structure, please refer to the new schema design documentation.

As we are going to store our video feeds on our storage, we also are storing the database records of those feeds and their attributes along with the camera which it was taken from. When the raw feeds move to archive the data of those will be migrated to arch_feeds from data_feeds. There will be a table for calibrated feeds which will contain other attributes like the raw feed which it is calibrated from as well as the sensor and camera name which it was calibrated from.

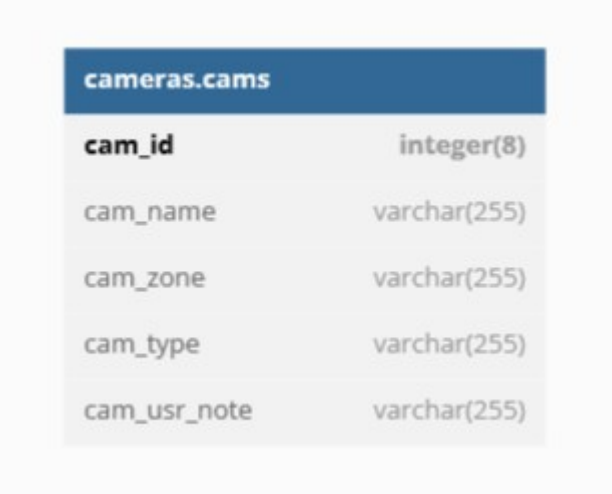
The database data_feeds will have a table raw_feeds for our raw feeds which are not archived and the table cal_feeds which will be a table for calibrated records.

The other database arch_feeds will be the same as the database data_feeds but have records only for the feeds which are archived.



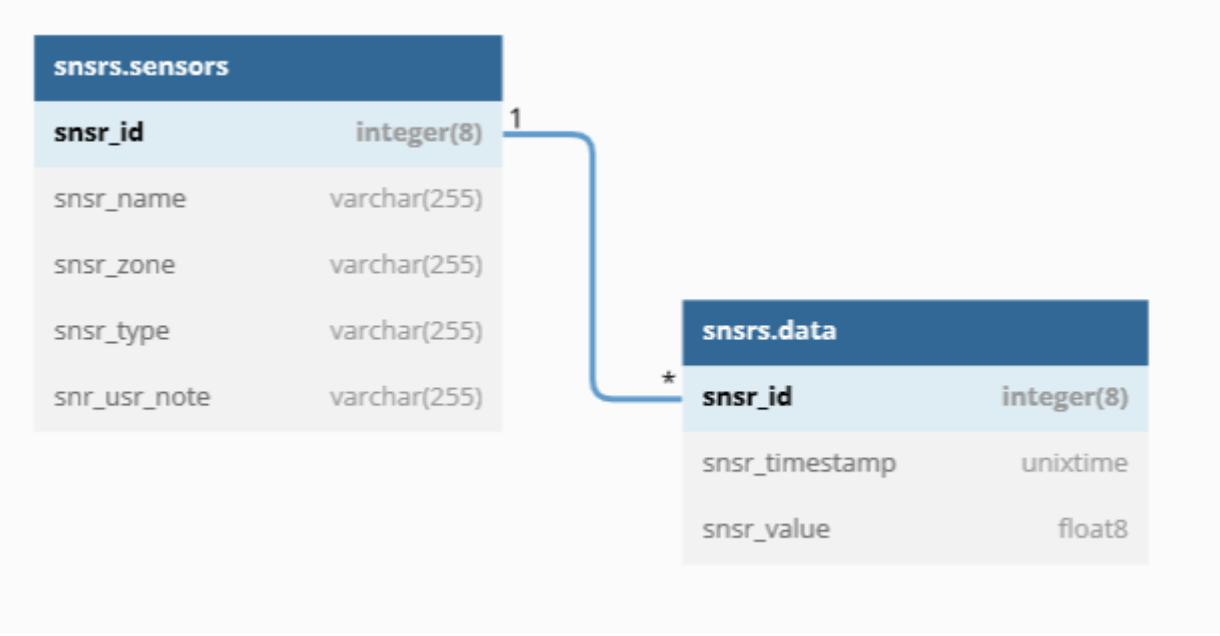
Database and table schema for available cameras

We will be keeping a record of our available camera in the table(cams) with their names, the zone to which it belongs, the type of camera and note/instruction for the user. Having this will help us to make a foreign key with reference to our other data.



Database and table schema for available sensors

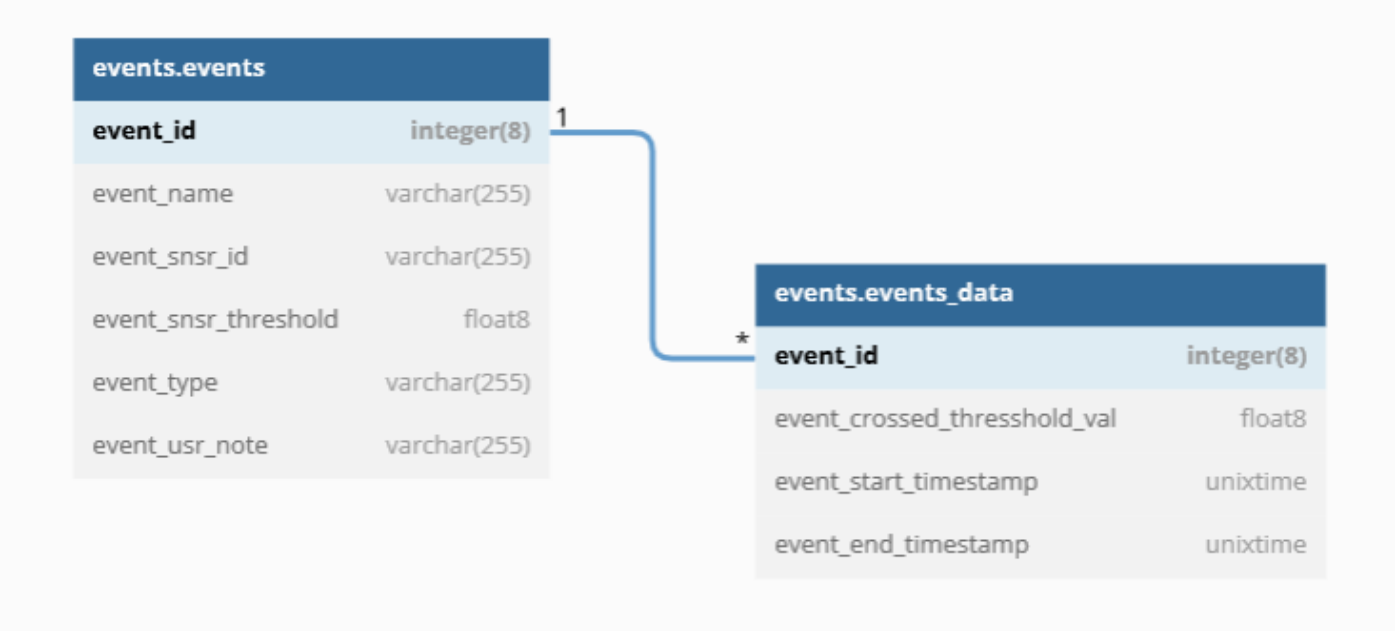
Here we will be having a table (sensors) which will have records of available sensors with their ID along with their name, zone to which it belongs, the type of sensor, and note/instructions for the user. The other table (data) will contain the records of data which is given by the sensor as output and the table will have sensor ID which is a reference to sensor ID from the sensors table, and other fields like the timestamp of when the data was recorded along with the sensor output value.



Database and table schema for events

Here we will be having two tables, the table events will have our pre-defined events that can happen in future and the fields like event ID, name along with the sensor which will detect that kind of event and also a threshold value for that sensor so when the value crossed the limit we can trigger our alert, and the event type and event note/instructions for the user.

The other table (data) will be having records of the events that already happened, fields like crossed threshold value start timestamp and end timestamp. The event ID here will be a reference to the event ID from table events.



Final Schema

The final schema of our time series databases and table and their relations. This is an initial schema and may change in future with needs.