**Re-written Schema of databases and tables aka buckets and measurements as per InfluxDB Standards**

I forgot to mention that InfluxDB is not an RDBMS (Relational Database Management System) and because of that database/table relations, Joints etc are not possible, unlike traditional SQL-like DBMS/RDMS. Also, there are different naming conventions and standards that InfluxDB follows, Here I’ll re-write our existing schema as per those.

Some basic terminologies:

* Schema - How the data are organized in InfluxDB. The fundamentals of the InfluxDB schema are databases, retention policies, series, measurements, tag keys, tag values, and field keys. See Schema Design for more information.
* Retention Policy (RP) - Describes how long InfluxDB keeps data (duration), how many copies of the data to store in the cluster (replication factor), and the time range covered by shard groups (shard group duration). RPs are unique per database and along with the measurement and tag set define a series. When you create a database, InfluxDB creates a retention policy called autogen with an infinite duration, a replication factor set to one, and a shard group duration set to seven days.
* Timestamp - The date and time associated with a point. All time in InfluxDB is UTC but can be changed.
* Measurement - The part of the InfluxDB data structure that describes the data stored in the associated fields. Measurements are strings.
* Tag - The key-value pair in the InfluxDB data structure that records metadata. Tags are an optional part of the data structure, but they are useful for storing commonly-queried metadata; tags are indexed so queries on tags are performant. Query tip: Compare tags to fields; fields are not indexed.
* Tag key - The key part of the key-value pair that makes up a tag. Tag keys are strings and they store metadata. Tag keys are indexed so queries on tag keys are performant.
* Tag set - The collection of tag keys and tag values on a point.
* Tag value - The value part of the key-value pair that makes up a tag. Tag values are strings and they store metadata. Tag values are indexed so queries on tag values are performant.
* Field - The key-value pair in an InfluxDB data structure that records metadata and the actual data value. Fields are required in InfluxDB data structures and they are not indexed - queries on field values scan all points that match the specified time range and, as a result, are not performant relative to tags.
* Field key - The key part of the key-value pair that makes up a field. Field keys are strings and they store metadata.
* Field set - The collection of field keys and field values on a point.
* Field value -The value part of the key-value pair that makes up a field. Field values are the actual data; they can be strings, floats, integers, or Booleans. A field value is always associated with a timestamp. Field values are not indexed - queries on field values scan all points that match the specified time range and, as a result, are not performant.

**InfluxDB Schema for /data/\* and /archive/\* feed records**

For /data we will be having a bucket names cam\_feeds with a retention policy of about 1 week. Under cam\_feeds measurements of raw\_feeds and calc\_feeds will be there. We will be using feed\_id as a tag key so that it can be indexable like a primary key. Field key will be used as key=value pair to store our records like feed\_title, start\_time, end\_time, vid\_path, and from\_cam. After 1 week i.e., our retention policy, all data will be transferred from raw\_feeds > arch\_raw\_feeds and calc\_feeds > arch\_calc\_feeds.

|  |  |
| --- | --- |
| **Buckets** | cam\_feeds  (Retention policy = 1 week) |
| **Measurements** | raw\_feeds, calc\_feeds, arch\_raw\_feeds, arch\_calc\_feeds |
| **Tags Keys** | feed\_id (String, index) |
| **Field Keys** | * feed\_title (String) * start\_time (unixtime) * end\_time (unixtime) * vid\_path (String) * from\_cam (String) |

Let us take a sample data for only 1 raw feed for /data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **\_time** | **\_measurement** | **feed\_id** | **\_field** | **\_value** |
| 2022-01-01T00:00:00Z | raw\_feeds | 3eqenu3e3 | feed\_title | brm\_stand1\_therm\_cam\_1641062400\_1641062700.mp4 |
| 2022-01-01T00:00:00Z | raw\_feeds | 3eqenu3e3 | start\_time | 1641062400 |
| 2022-01-01T00:00:00Z | raw\_feeds | 3eqenu3e3 | end\_time | 1641062700 |
| 2022-01-01T00:00:00Z | raw\_feeds | 3eqenu3e3 | vid\_path | /data/raw/202209/640975400/ |
| 2022-01-01T00:00:00Z | raw\_feeds | 3eqenu3e3 | from\_cam | stand\_1\_thermal\_cam |

Schema:

raw\_feeds, feed\_id=3eqenu3e3, feed\_title=brm\_stand1\_therm\_cam\_1641062400\_1641062700.mp4, start\_time=1641062400, end\_time=1641062700, vid\_path=/data/raw/202209/640975400/, from\_cam=stand\_1\_thermal\_cam

Same for calc\_feeds:

calc\_feeds, feed\_id=3eqenu3e3, feed\_title=brm\_stand1\_therm\_cam\_1641062400\_1641062700.mp4, start\_time=1641062400, end\_time=1641062700, vid\_path=/data/calc/202209/640975400/, from\_cam=stand\_1\_thermal\_cam

**InfluxDB Schema for sensor data**

Here we will be having all the data which are available from the installed sensors. Along with the other fields like \_time, sensor\_name, sensor\_zone, sensor\_type, and sensor\_value.

|  |  |
| --- | --- |
| **Buckets** | snsrs\_data  (Retention policy = 1 week) |
| **Measurements** | data |
| **Tags Keys** | snsr\_id (String, index) |
| **Field Keys** | * snsr\_name (String) * snsr\_zone (String) * snsr\_type (String) * snsr\_value (String) |

Let us take 1 sample data for sensor\_data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **\_time** | **\_measurement** | **snsr\_id** | **\_field** | **\_value** |
| 2022-01-01T00:00:00Z | data | 89d5a82s | snsr\_name | infrared\_sensor\_xyz\_inc\_bsp |
| 2022-01-01T00:00:00Z | data | 89d5a82s | snsr\_zone | brm\_stand\_16 |
| 2022-01-01T00:00:00Z | data | 89d5a82s | snsr\_type | infrared\_snsr |
| 2022-01-01T00:00:00Z | data | 89d5a82s | snsr\_value | 846.694546221949562 |

Schema:

data, snsr\_id=89d5a82s, snsr\_name=infrared\_sensor\_xyz\_inc\_bsp, snsr\_zone=brm\_stand\_16, snsr\_type=infrared\_snsr, snsr\_value=846.694546221949562

As influxDB will automatically input a \_time whenever the record is entered, we will not have a different field for a timestamp and I’m assuming that there is no delay between the actual timestamp when a sensor has taken a value and the value of timestamp when the recorded is inserted, although ill makes some tests.

**InfluxDB Schema design for abnormal pattern records**

We are going to store records which are abnormal or fall under our defined abnormal pattern category. This data will help us in future regarding analysis and study of the data wherever, when and how the cobble or any other incident has happened.

|  |  |
| --- | --- |
| **Buckets** | abnormal\_pattern  (Retention policy = 1 week) |
| **Measurements** | data |
| **Tags Keys** | patt\_id (String, index) |
| **Field Keys** | * patt\_type (String) * patt\_snsr (String) * patt\_zone (String) * patt\_occ\_at (unixtime) * patt\_occ\_till (unixtime) |

Let us take 1 sample data for abnormal\_pattern:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **\_time** | **\_measurement** | **pattern\_id** | **\_field** | **\_value** |
| 2022-01-01T00:00:00Z | data | 5sd5sd68 | patt\_type | cobble\_med\_brm |
| 2022-01-01T00:00:00Z | data | 5sd5sd68 | patt\_snsr | infrared\_snsr |
| 2022-01-01T00:00:00Z | data | 5sd5sd68 | patt\_zone | brm\_stand\_16 |
| 2022-01-01T00:00:00Z | data | 5sd5sd68 | patt\_occ\_at | 1641062400 |
| 2022-01-01T00:00:00Z | data | 5sd5sd68 | patt\_occ\_till | 1641062700 |

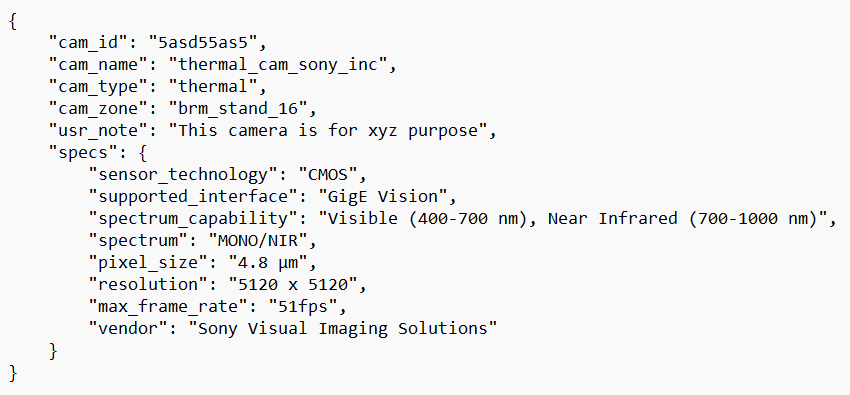
Schema:

data, patt\_id=89d5a82s, patt\_type=cobble\_med\_brm, patt\_zone=brm\_stand\_16, patt\_occ\_at=1641062400, patt\_occ\_till=1641062400

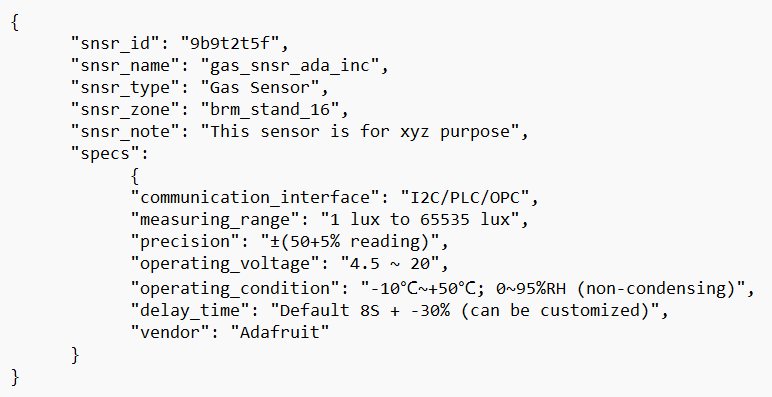
**Workaround to relate records with InfluxDB**

As I have mentioned that influxDB is not an RDBMS and because of that relations aren’t possible with influxDB. I was thinking to make a separate index for our available camera, sensors, and defined patterns (events) in our schema based on JSON. This will help us to have an index of those along with their IDs which are also a tag key in our influxDB records.

JSON schema sample for the camera:



JSON schema sample for our available sensors:



JSON schema sample for our pre-defined events:



**Note: If we need to get more data about the sensor, camera, and more about events we can get it from these indexes using the IDs which are used as tags in our influxDB.**

Security: In influxDB, we have token/password-based authentication for security, but to keep these index data secure we can limit permissions for the user using the UNIX system permission concept also we can encrypt these data using Encryption methods like AES. Encryption, Integrity, Security etc are very important for us and I’ll write and explain more about it later in the other document.

**References:**

* <https://docs.influxdata.com/influxdb/v2.4/reference/key-concepts/data-elements/>
* <https://docs.influxdata.com/influxdb/v2.4/reference/key-concepts/data-schema/>
* <https://www.influxdata.com/blog/data-layout-and-schema-design-best-practices-for-influxdb/>
* <https://docs.influxdata.com/influxdb/cloud/write-data/best-practices/schema-design/>