S4L - CLI Queries gegen Influx ausführen

Grundsätzliches

Stats4Lox liefert ein Wrapper-Shellscript mit, mit dem das Influx CLI Programm ohne Eingabe von Host/User/Pass möglich ist. Dieses Wrapperscript meldet sich automatisch an der Influx-Instanz an und übergibt alle Parameter weiter an Influx

```
/opt/loxberry/bin/plugins/stats4lox/s4linflux -database 'stats4lox' -execute
"SELECT value FROM testdata"
```

InfluxQL Syntax

InfluxQL ist eine an SQL angelehnte Querysprache.

Grundsätzliches: https://docs.influxdata.com/influxdb/v1.8/query_language/

Funktionen: https://docs.influxdata.com/influxdb/v1.8/query_language/functions/

Beispielabfragen von Stats4Lox-Daten:

Verschiedene Beispielabfragen

```
# 10 Außentemperaturen
/opt/loxberry/bin/plugins/stats4lox/s4linflux -database 'stats4lox' -execute
'SELECT "Default" FROM "Außentemperatur" LIMIT 10'
time                Default
----                -------
1341153900000000000 27.531
1341154200000000000 27.437
1341154500000000000 27.219
1341154800000000000 26.722
1341155100000000000 25.7
1341155400000000000 24.528
1341155700000000000 23.725
1341156000000000000 23.25
1341156300000000000 23.031
1341156600000000000 22.875

# Außentemperaturen der letzten Stunde
/opt/loxberry/bin/plugins/stats4lox/s4linflux -database 'stats4lox' -execute
'SELECT "Default" FROM "Außentemperatur" WHERE time > now()-60m'
name: Außentemperatur
time                Default
----                -------
```

```
1634652061000000000 13.7
1634652361000000000 13.6
1634652661000000000 13.5
1634652961000000000 13.4
1634653561000000000 13.1
1634653861000000000 13.1

# Erfasste Minimum/Maximum Temperatur seit 2020
/opt/loxberry/bin/plugins/stats4lox/s4linflux -database 'stats4lox' -execute
'SELECT MAX("Default"), MIN("Default") FROM "Außentemperatur" WHERE
"Default" > -128;'
name: Außentemperatur
time max    min
---- ---    ---
0    49.625 -68.688
```

Wegen des Quotings sind Queries mit Zeitstempeln (WHERE time > '2020-01-01' usw.) problematisch bzw. funktionieren nicht oder nicht richtig.

Auffüllen und Gruppieren

Dieses Query macht Folgendes:

- `time >= 6h and time <= now()` Abfrage der letzten 6 Stunden
- `GROUP BY time(1h)` Werte auf 1h Werte gruppieren
- `fill(previous)` Es wird jeweils angenommen, dass der letzte Wert solange beibehalten bleibt, bis der nächste kommt (das, was man in einer Grafik als gerade Linie sehen würde)

```
/opt/loxberry/bin/plugins/stats4lox/s4linflux -database stats4lox -execute
'SELECT mean("Default") FROM "Außentemperatur" WHERE time >= now() - 6h and
time <= now() GROUP BY time(1h) fill(previous)'
```

Beispiel mit Testdaten

Script zum Testdaten erzeugen

Mit diesem Script erzeugst du in Influx die Test-Datenreihe, mit der ich hier gespielt habe:

Script zum Anlegen von Beispieldaten

```
#!/bin/bash
PLUGINNAME=stats4lox
INFLUXUSERNAME=`jq -r '.influx.influxdbuser'
$LBPCONFIG/$PLUGINNAME/cred.json`
```

```
INFLUXPASSWORD=`jq -r '.influx.influxdbpass'
$LBPCONFIG/$PLUGINNAME/cred.json`

curl -k -i -XPOST
"https://$INFLUXUSERNAME:$INFLUXPASSWORD@localhost:8086/write?db=stats4lox"
--data-binary 'testdata value=5 1634475600000000000'
curl -k -i -XPOST
"https://$INFLUXUSERNAME:$INFLUXPASSWORD@localhost:8086/write?db=stats4lox"
--data-binary 'testdata value=10 1634475605000000000'
curl -k -i -XPOST
"https://$INFLUXUSERNAME:$INFLUXPASSWORD@localhost:8086/write?db=stats4lox"
--data-binary 'testdata value=1 1634475780000000000'
curl -k -i -XPOST
"https://$INFLUXUSERNAME:$INFLUXPASSWORD@localhost:8086/write?db=stats4lox"
--data-binary 'testdata value=3 1634476200000000000'
```

Download: inserttestdata.sh

Beim Ausführen sollte 4x `HTTP/1.1 204 No Content` dortstehen - das ist OK!

Zeit-Ausgabe bei Influx Queries

Der Influx Timestamp ist immer Epoch in Nanosekunden

1634475600000000000 ⇒ 1634475600 Sekunden EPOCH ⇒ Sonntag, 17. Oktober 2021 15:00:00 GMT+02:00 DST

```
loxberry@loxberry-dev:/ $ /opt/loxberry/bin/plugins/stats4lox/s4linflux -
database stats4lox -execute 'SELECT value FROM "testdata"'
name: testdata
time                value
----                -----
1634475600000000000 5
1634475605000000000 10
1634475780000000000 1
1634476200000000000 3
```

Mit dem Commandline-Parameter **`-precision <Format>`** kann man die Zeitausgabe verändern:

```
loxberry@loxberry-dev:/ $ /opt/loxberry/bin/plugins/stats4lox/s4linflux -
database stats4lox -precision rfc3339 -execute 'SELECT value FROM "testdata"'
name: testdata
time                 value
----                 -----
2021-10-17T13:00:00Z 5
2021-10-17T13:00:05Z 10
2021-10-17T13:03:00Z 1
2021-10-17T13:10:00Z 3
```

Aggregieren der Beispiel-Werte

Die Beispielwerte könnten z.B. per MQTT Live bei jeder Änderung übertragen worden sein, ohne Intervall für einen zeitbasierten Abruf.

Dabei wurde der erste Wert 5 um 13:00:00 übertragen, dann fünf Sekunden später der Wert 10, dann nach knapp drei Minuten der Wert 1 und nach insgesamt zehn Minuten der Wert 3.

Eine normale Statistik würde die Punkte dieser Werte einfach verbinden, was aber eigentlich falsch ist.

Tatsächlich bleibt der Wert so lange konstant, bis der nächste folgt, grafisch also so:

Ziel der Übung ist es, aus einer Anzahl von zu vielen oder aber auch zu wenigen Datenpunkten einen Durchschnittswert für ein vorgegebenes Zeitintervall zu bekommen.

Ich gehe davon aus, dass ich einen **zeitlich gewichteten Mittelwert über 2 Minuten** haben möchte.

Berechnung mit der Hand

Um die späteren Ergebnisse zu kontrollieren, berechne ich erstmal das erwartete Ergebnis per "Hand"

Zeit	Gewichteter Mittelwert	Anmerkungen
13:00:00	(5)	Noch kein Mittelwert möglich
13:02:00	9,792	Für 5 Sekunden Wert 5, danach Wert 10. Berechnung (5s*5 + 115s*10) / 120
13:04:00	5,5	Für 60 Sekunden Wert 10, danach Wert 1. Berechnung (60s*10 + 60s*1) / 120
13:06:00	1	Kein neuer Wert, daher letzter Wert 1
13:08:00	1	Kein neuer Wert, daher letzter Wert 1

Zeit	Gewichteter Mittelwert	Anmerkungen
13:10:00	1	Kein neuer Wert, daher letzter Wert 1
13:12:00	3	Neuer Wert 3

Versuch 1: Query mit GROUP BY und FILL - das wird wohl nix

```
/opt/loxberry/bin/plugins/stats4lox/s4linflux -database stats4lox -precision
rfc3339 -execute 'SELECT time_weighted_average("value") FROM "testdata" WHERE
time <=  1634476200000000000  GROUP BY time(2m) fill(previous)'
```

GROUP BY time(2m) gibt die Durchschnittswerte von 2 Minuten-Gruppen aus.

fill(previous) füllt nicht existierende Werte mit dem zuletzt verfügbaren Wert auf

Logisch müsste das unseren zeitlich gewichteten Mittelwert ergeben. Aber leider:

```
time                 mean
----                 ----
2021-10-17T13:00:00Z 7.5
2021-10-17T13:02:00Z 1
2021-10-17T13:04:00Z 1
2021-10-17T13:06:00Z 1
2021-10-17T13:08:00Z 1
2021-10-17T13:10:00Z 3
```

Das fill(previous) füllt leider nur die Zeiträume der GROUP-Bedingung auf, die überhaupt keinen Wert enthalten. Es füllt allerdings nicht die Zeiträume zwischen mehreren Datenpunkten auf. Daher berechnet das GROUP BY nur das arithmetische Mittel der eingetroffenen Werte, keinen zeitlich gewichteten Mittelwert.

Versuch 2: Subquery mit groupierten Werten

Wenn wir die obere Query modifizieren, dass nicht ein Durchschnittswert von 2 Minuten. sondern **1 Sekunde** gebildet wird, erhalten wir eine Datenreihe mit den Werten, die zu jeder Sekunde gültig waren:

```
/opt/loxberry/bin/plugins/stats4lox/s4linflux -database stats4lox -precision
rfc3339 -execute 'SELECT mean("value") FROM "testdata" WHERE time <=
 1634476200000000000  GROUP BY time(1s) fill(previous)'
name: testdata
time                 mean
----                 ----
2021-10-17T13:00:00Z 5
2021-10-17T13:00:01Z 5
2021-10-17T13:00:02Z 5
2021-10-17T13:00:03Z 5
2021-10-17T13:00:04Z 5
2021-10-17T13:00:05Z 10
2021-10-17T13:00:06Z 10
```

```
2021-10-17T13:00:07Z 10
2021-10-17T13:00:08Z 10
2021-10-17T13:00:09Z 10
2021-10-17T13:00:10Z 10
2021-10-17T13:00:11Z 10
2021-10-17T13:00:12Z 10

usw.
```

Wenn wir nun dieses Ergebnis nochmals gruppieren in 2-Minuten-Gruppen, dann sollte unser richtiger Mittelwert herauskommen:

```
/opt/loxberry/bin/plugins/stats4lox/s4linflux -database stats4lox -precision
rfc3339 -execute 'SELECT mean("mean") FROM (SELECT mean("value") FROM
"testdata" WHERE time <=  1634476200000000000  GROUP BY time(1s)
fill(previous)) WHERE time <= 1634476200000000000 GROUP BY time(2m)
fill(previous)'
```

(ich habe das Subquery blau markiert)

```
name: testdata
time                 mean
----                 ----
2021-10-17T13:00:00Z 9.791666666666666
2021-10-17T13:02:00Z 5.5
2021-10-17T13:04:00Z 1
2021-10-17T13:06:00Z 1
2021-10-17T13:08:00Z 1
2021-10-17T13:10:00Z 3
```

Damit haben wir für unser Beispiel tatsächlich einen zeitlich gewichteten Mittelwert berechnet!

Erkenntnisse und Nebenwirkungen

1. Es ist schade, dass Influx keinen zeitlich gewichteten Mittelwert als Standardfunktion mitliefert - schließlich ist es eine Datenbank speziell für Zeitserien, und es ist in vielen Situationen der Fall, dass Werte nicht in einem festen Intervall eintreffen.
2. Mit einer Subquery ist es trotzdem möglich, einen zeitlich gewichteten Mittelwert zu bilden, oder zumindest *annähernd* abzubilden (siehe 3.)
3. Die Genauigkeit dies auf diese Weise gebildeten Mittelswerts ist vom Intervall der Subquery abhängig - hier 1 Sekunde. Innerhalb dieser 1 Sekunde wird wieder der (falsche) arithmetische Mittelwert gebildet: Für 0,1 Sek. 1000W und 0,9 Sek. 10W ergibt für diese Sekunde das falsche Ergebnis 505W (statt 109W).
4. Die Datenmenge, die aufgrund dieser Abfrage durch Influx geschleift wird, ist **riesig** im Vergleich zu den tatsächlich vorhandenen Datenpunkten. Es ist möglich, dass man die Abfrage zeitlich limitieren muss (time), damit sie über größere Zeiträume hinweg überhaupt funktioniert.

Ich denke aber, für den "Hausgebrauch" sollten die Seiteneffekte verschmerzbar sein.

Ergebnisse in neues Influx Measurement schreiben

Es geht darum, dass die Ergebnisse einer Query selbst wieder in eine Zeitreihe in Influx geschrieben werden.

```
/opt/loxberry/bin/plugins/stats4lox/s4linflux -database stats4lox -precision
rfc3339 -execute 'SELECT mean("mean") INTO testdata_2min FROM (SELECT
mean("value") FROM "testdata" WHERE time <=  1634476200000000000  GROUP BY
time(1s) fill(previous)) WHERE time <= 1634476200000000000 GROUP BY time(2m)
fill(previous)'
```

Kontrolle:

```
/opt/loxberry/bin/plugins/stats4lox/s4linflux -database stats4lox -precision
rfc3339 -execute 'SELECT * from testdata_2min'
name: testdata_2min
time                 mean
----                 ----
2021-10-17T13:00:00Z 9.791666666666666
2021-10-17T13:02:00Z 5.5
2021-10-17T13:04:00Z 1
2021-10-17T13:06:00Z 1
2021-10-17T13:08:00Z 1
2021-10-17T13:10:00Z 3
```

Ergebnisse in ein CSV schreiben

```
/opt/loxberry/bin/plugins/stats4lox/s4linflux -database stats4lox -precision
rfc3339 -format csv -execute 'SELECT * from testdata_2min'
name,time,mean
testdata_2min,2021-10-17T13:00:00Z,9.791666666666666
testdata_2min,2021-10-17T13:02:00Z,5.5
testdata_2min,2021-10-17T13:04:00Z,1
testdata_2min,2021-10-17T13:06:00Z,1
testdata_2min,2021-10-17T13:08:00Z,1
testdata_2min,2021-10-17T13:10:00Z,3
```

oder direkt in eine Datei

```
/opt/loxberry/bin/plugins/stats4lox/s4linflux -database stats4lox -precision
rfc3339 -format csv -execute 'SELECT * from testdata_2min' > testdata.csv
```

Interessante Befehle und Querys

```
/opt/loxberry/bin/plugins/stats4lox/s4linflux
```

```
use stats4lox
```

Alle Measurements anzeigen

```
show measurements
```

Fields eines Measurements anzeigen

```
show field keys from "loxberry/mqttgateway"
```

Anzahl der Datensätze eines Measures

```
select count(*) from "UG Vorlauf";
```

Die Query dann durchaus ein paar Sekunden dauern.

Anzahl der Datensätze eines Fields

```
select count("pollms") from "loxberry/mqttgateway"
```

Erster und letzter Wert (Timestamps)

Das Ergebnis könnte jeweils mehrere Werte liefern

Erster Wert

```
SELECT * FROM "UG Vorlauf" GROUP BY * ORDER BY ASC LIMIT 1
```

Letzter Wert

```
SELECT * FROM "UG Vorlauf" GROUP BY * ORDER BY DESC LIMIT 1
```

From:
https://wiki.loxberry.de/ - **LoxBerry Wiki - BEYOND THE LIMITS**

Permanent link:
https://wiki.loxberry.de/plugins/statistics_4_loxone/s4l_cli_queries_gegen_influx_ausfuhren

Last update: **2022/09/16 16:33**