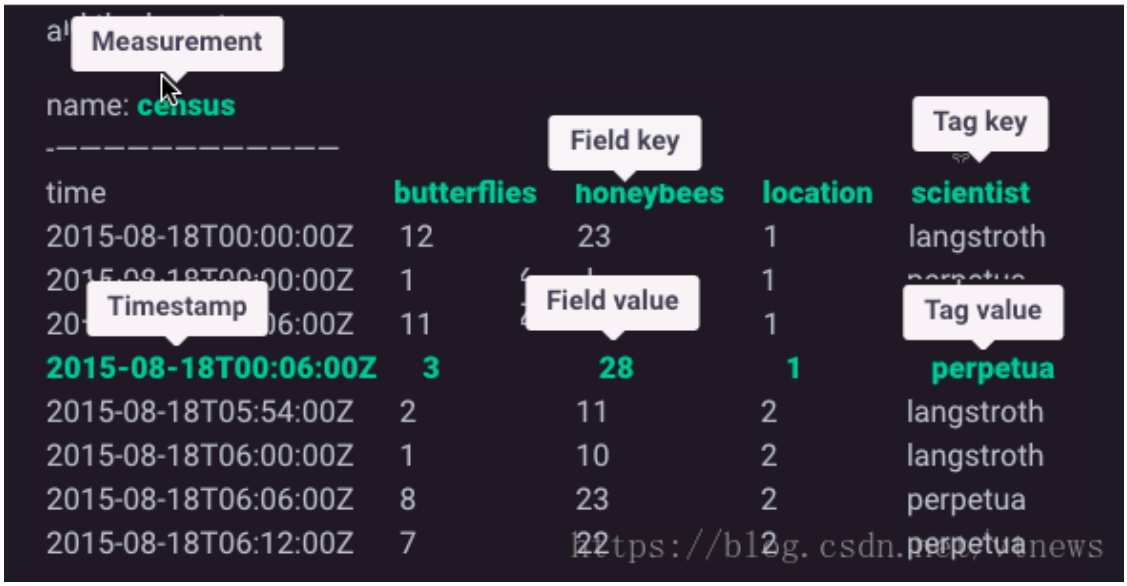
influxDB学习

1、基本概念

1.1、数据格式

- database: 数据库名，在 InfluxDB 中可以创建多个数据库，不同数据库中的数据文件是隔离存放的，存放在磁盘上的不同目录。
- retention policy: 存储策略，用于设置数据保留的时间，每个数据库刚开始会自动创建一个默认的存储策略 autogen，数据保留时间为永久，之后用户可以自己设置，例如保留最近2小时的数据。插入和查询数据时如果不指定存储策略，则使用默认存储策略，且默认存储策略可以修改。InfluxDB 会定期清除过期的数据。
- measurement: 测量指标名，例如 cpu_usage 表示 cpu 的使用率。
- tag sets: tags 在 InfluxDB 中会按照字典序排序，不管是 tagk 还是 tagv，只要不一致就分别属于两个 key，例如 host=server01,region=us-west 和 host=server02,region=us-west 就是两个不同的 tag set。
- tag--标签，在InfluxDB中，tag是一个非常重要的部分，表名+tag一起作为数据库的索引，是"key-value"的形式。
- field name: 例如上面数据中的 value 就是 fieldName，InfluxDB 中支持一条数据中插入多个 fieldName，这其实是一个语法上的优化，在实际的底层存储中，是当作多条数据来存储。
- timestamp: 每一条数据都需要指定一个时间戳，在 TSM 存储引擎中会特殊对待，以为了优化后续的查询操作。



1.2、与传统数据库对比

influxDB中的名词	传统数据库中的概念
database	数据库
measurement	数据库中的表
points	表里面的一行数据

1.3、points

Point由时间戳、数据、标签组成。

Point相当于传统数据库里的一行数据，如下表所示：

Point属性	传统数据库中的概念
time	每个数据记录时间，是数据库中的主索引(会自动生成)
fields	各种记录值（没有索引的属性）
tags	各种有索引的属性

1.4、Series

Series 相当于是 InfluxDB 中一些数据的集合，在同一个 database 中，retention policy、measurement、tag sets 完全相同的数据同属于一个 series，同一个 series 的数据在物理上会按照时间顺序排列存储在一起。

1.5、 Shard

Shard 在 InfluxDB 中是一个比较重要的概念，它和 retention policy 相关联。每一个存储策略下会存在许多 shard，每一个 shard 存储一个指定时间段内的数据，并且不重复，例如 7点-8点 的数据落入 shard0 中，8点-9点的数据则落入 shard1 中。

2、查询语句（数据探索）

2.1、Select

语法：

```
SELECT <field_key>[,<field_key>,<tag_key>] FROM <measurement_name>[,
<measurement_name>]
```

SLECT *：返回所有的field和tag。

SELECT "<field_key>"：返回特定的field。

SELECT "<field_key>","<field_key>"：返回多个field。

SELECT "<field_key>","<tag_key>"：返回特定的field和tag，SELECT在包括一个tag时，必须至少指定一个field。

SELECT "<field_key>"::field,"<tag_key>"::tag：返回特定的field和tag，::[field | tag]语法指定标识符的类型。

FROM子句：

FROM <measurement_name>：从单个measurement返回数据。

FROM <measurement_name>,<measurement_name>：从多个measurement中返回数据。

FROM <database_name>.<retention_policy_name>.<measurement_name>：从一个完全指定measurement中返回数据，这个完全指定是指指定了数据库和存储策略。

FROM <database_name>..<measurement_name>：从一个用户指定的数据库中返回存储策略为DEFAULT的数据。

example:

```
// 从measurement查询所有的field和tag
> SELECT * FROM "h2o_feet"
name: h2o_feet
--------------
time                   level description      location       water_level
2015-08-18T00:00:00Z   below 3 feet           santa_monica   2.064
2015-08-18T00:00:00Z   between 6 and 9 feet   coyote_creek   8.12
[...]
2015-09-18T21:36:00Z   between 3 and 6 feet   santa_monica   5.066
2015-09-18T21:42:00Z   between 3 and 6 feet   santa_monica   4.938

// measurement中查询特定tag和field
> SELECT "level description","location","water_level" FROM "h2o_feet"
// 从measurement中选择特定的tag和field，并提供其标识符类型
> SELECT "level description"::field,"location"::tag,"water_level"::field FROM
"h2o_feet"
// 从measurement中选择一个特定的field并执行基本计算
> SELECT ("water_level" * 2) + 4 from "h2o_feet"
// 从多个measurement中查询数据
> SELECT * FROM "h2o_feet","h2o_pH"
// 从完全限定的measurement中选择所有数据
> SELECT * FROM "NOAA_water_database"."autogen"."h2o_feet"
```

2.2、Where

语法：

```
SELECT_clause FROM_clause WHERE <conditional_expression> [(AND|OR)
<conditional_expression> [...]]
```

fields

`WHERE`子句支持field value是字符串，布尔型，浮点数和整数。
在`WHERE`子句中**单引号**来表示字符串字段值。无引号字符串字段值或双引号字符串字段值的查询不会返回任何数据，并且在大多数情况下也不会返回错误。

语法：

```
field_key <operator> ['string' | boolean | float | integer]
```

支持的操作符：`=` 等于 `<>` 不等于 `!=` 不等于 `>` 大于 `>=` 大于等于 `<` 小于 `<=` 小于等于

tags

`WHERE`子句中的用单引号来把tag value引起来

```
tag_key <operator> ['tag_value']
```

timestamps

对于大多数SELECT语句，默认时间范围为UTC的1677-09-21 00:12:43.145224194到2262-04-11T23:47:16.854775806Z。 对于只有GROUP BY time()子句的SELECT语句，默认时间范围在UTC的1677-09-21 00：12：43.145224194和now()之间。

```
// 查询有特定field的key value为字符串的数据
> SELECT * FROM "h2o_feet" WHERE "level description" = 'below 3 feet'
```

```
// 查询有特定field的key value并且带计算的数据
> SELECT * FROM "h2o_feet" WHERE "water_level" + 2 > 11.9
//查询tag_key不等于'santa_monica'并且water_level小于-0.59或大于9.95
> SELECT "water_level" FROM "h2o_feet" WHERE "location" <> 'santa_monica' AND
(water_level < -0.59 OR water_level > 9.95)

name: h2o_feet
--------------
time                   water_level
2015-08-29T07:18:00Z   9.957
2015-08-29T07:24:00Z   9.964
2015-08-29T07:30:00Z   9.954
2015-08-29T14:30:00Z   -0.61
2015-08-29T14:36:00Z   -0.591
2015-08-30T15:18:00Z   -0.594
// 根据时间戳来过滤数据
> SELECT * FROM "h2o_feet" WHERE time > now() - 7d
```

2.3、GROUP BY

语法

```
SELECT_clause FROM_clause [WHERE_clause] GROUP BY [* | <tag_key>[,<tag_key]]
```

`GROUP BY *`：对结果中的所有tag作group by。

`GROUP BY <tag_key>`：对结果按指定的tag作group by。

`GROUP BY <tag_key>,<tag_key>`：对结果数据按多个tag作group by，其中tag key的顺序不影响。

example:

```
// 对单个tag作group by
> SELECT MEAN("water_level") FROM "h2o_feet" GROUP BY "location"
name: h2o_feet
tags: location=coyote_creek
time                           mean
----                           ----
1970-01-01T00:00:00Z     5.359342451341401
name: h2o_feet
tags: location=santa_monica
time                           mean
----                           ----
1970-01-01T00:00:00Z     3.530863470081006

// 对多个tag作group by
> SELECT MEAN("index") FROM "h2o_quality" GROUP BY location,randtag

name: h2o_quality
tags: location=coyote_creek, randtag=1
time                   mean
----                   ----
1970-01-01T00:00:00Z  50.69033760186263

name: h2o_quality
tags: location=coyote_creek, randtag=2
time                    mean
```

```
----                    ----
1970-01-01T00:00:00Z    49.661867544220485

// 例三：对所有tag作group by
> SELECT MEAN("index") FROM "h2o_quality" GROUP BY *
```

GROUP BY time()

`GROUP BY time()` 返回结果按指定的时间间隔group by。

```
SELECT <function>(<field_key>) FROM_clause WHERE <time_range> GROUP BY
time(<time_interval>),[tag_key] [fill(<fill_option>)]
```

time(time_interval,offset_interval)： 决定InfluxDB按什么时间间隔group by

offset_interval是一个持续时间。它向前或向后移动InfluxDB的预设时间界限。offset_interval可以为正或负。

fill(<fill_option>)： 可选

example:

```
// 时间间隔为12分钟并且还对tag key作group by
> SELECT COUNT("water_level") FROM "h2o_feet" WHERE time >= '2015-08-
18T00:00:00Z' AND time <= '2015-08-18T00:30:00Z' GROUP BY time(12m),"location"

name: h2o_feet
tags: location=coyote_creek
time                    count
----                    -----
2015-08-18T00:00:00Z    2
2015-08-18T00:12:00Z    2
2015-08-18T00:24:00Z    2

name: h2o_feet
tags: location=santa_monica
time                    count
----                    -----
2015-08-18T00:00:00Z    2
2015-08-18T00:12:00Z    2
2015-08-18T00:24:00Z    2

// 查询结果间隔按18分钟group by，并将预设时间边界向前移动
> SELECT MEAN("water_level") FROM "h2o_feet" WHERE "location"='coyote_creek' AND
time >= '2015-08-18T00:06:00Z' AND time <= '2015-08-18T00:54:00Z' GROUP BY
time(18m,6m)

name: h2o_feet
time                    mean
----                    ----
2015-08-18T00:06:00Z    7.884666666666667
2015-08-18T00:24:00Z    7.502333333333333
2015-08-18T00:42:00Z    7.108666666666667

> SELECT MEAN("water_level") FROM "h2o_feet" WHERE "location"='coyote_creek' AND
time >= '2015-08-18T00:06:00Z' AND time <= '2015-08-18T00:54:00Z' GROUP BY
time(18m)
```

```
name: h2o_feet
time                   mean
----                   ----
2015-08-18T00:00:00Z   7.946
2015-08-18T00:18:00Z   7.6323333333333325
2015-08-18T00:36:00Z   7.238666666666667
2015-08-18T00:54:00Z   6.982
```

2.4、Into

语法：

```
SELECT_clause INTO <measurement_name> FROM_clause [WHERE_clause]
[GROUP_BY_clause]
```

INTO <measurement_name>：写入到特定measurement中
INTO <database_name>.<retention_policy_name>.<measurement_name>：写入到完整指定的measurement中。
INTO <database_name>..<measurement_name>：写入到指定数据库保留策略为DEFAULT。
INTO <database_name>.<retention_policy_name>.:MEASUREMENT FROM /<regular_expression>/：将数据写入与FROM子句中正则表达式匹配的用户指定数据库和保留策略的所有measurement。 :MEASUREMENT是对FROM子句中匹配的每个measurement的反向引用。（相当于从一个库的这些表，拷贝到新数据源的同名表下）
example:

```
> SELECT "water_level" INTO "h2o_feet_copy_1" FROM "h2o_feet" WHERE "location" =
'coyote_creek'
name: result
------------
time                   written
1970-01-01T00:00:00Z   7604
> SELECT * FROM "h2o_feet_copy_1"
name: h2o_feet_copy_1
----------------------
time                   water_level
2015-08-18T00:00:00Z   8.12
[...]
2015-09-18T16:48:00Z   4
```

2.5、LIMIT和SLIMIT子句

LIMIT , N指定每次measurement返回的points数。如果N大于measurement的点数，InfluxDB将从该测量中返回所有points。
SLIMIT , N指定从指定measurement返回的series数。如果N大于measurement中series数，InfluxDB将从该measurement中返回所有series。

语法：

```
SELECT_clause [INTO_clause] FROM_clause [WHERE_clause] [GROUP_BY_clause]
[ORDER_BY_clause] LIMIT <N>
```

```
SELECT_clause [INTO_clause] FROM_clause [WHERE_clause] GROUP BY *
[,time(<time_interval>)] [ORDER_BY_clause] SLIMIT <N>
```

```
SELECT_clause [INTO_clause] FROM_clause [WHERE_clause] GROUP BY *
[,time(<time_interval>)] [ORDER_BY_clause] LIMIT <N1> SLIMIT <N2>
```

example:

```
// 限制返回的点数
> SELECT "water_level","location" FROM "h2o_feet" LIMIT 3
// 例一：限制返回的series的数目
> SELECT "water_level" FROM "h2o_feet" GROUP BY * SLIMIT 1
// 例二：限制数据点数和series数并且包括一个GROUP BY time()子句
> SELECT MEAN("water_level") FROM "h2o_feet" WHERE time >= '2015-08-
18T00:00:00Z' AND time <= '2015-08-18T00:42:00Z' GROUP BY *,time(12m) LIMIT 2
SLIMIT 1

name: h2o_feet
tags: location=coyote_creek
time                   mean
----                   ----
2015-08-18T00:00:00Z   8.0625
2015-08-18T00:12:00Z   7.8245
```

2.6 OFFSET 和 SOFFSET

`OFFSET <N>` 从查询结果中返回分页的N个points
`SOFFSET <M>` 从查询结果中返回分页的M个series

```
SELECT_clause [INTO_clause] FROM_clause [WHERE_clause] [GROUP_BY_clause]
[ORDER_BY_clause] LIMIT_clause OFFSET <N> [SLIMIT_clause]
```

```
SELECT_clause [INTO_clause] FROM_clause [WHERE_clause] GROUP BY *
[,time(time_interval)] [ORDER_BY_clause] [LIMIT_clause] [OFFSET_clause]
SLIMIT_clause SOFFSET <N>
```

example:

```
> SELECT MEAN("water_level") FROM "h2o_feet" WHERE time >= '2015-08-
18T00:00:00Z' AND time <= '2015-08-18T00:42:00Z' GROUP BY *,time(12m) ORDER BY
time DESC LIMIT 2 OFFSET 2 SLIMIT 1 SOFFSET 1

name: h2o_feet
tags: location=santa_monica
time                   mean
----                   ----
2015-08-18T00:12:00Z   2.077
2015-08-18T00:00:00Z   2.09
```

2.7 TimeZone

`tz()` 子句返回指定时区的UTC偏移量

语法：

```
SELECT_clause [INTO_clause] FROM_clause [WHERE_clause] [GROUP_BY_clause]
[ORDER_BY_clause] [LIMIT_clause] [OFFSET_clause] [SLIMIT_clause]
[SOFFSET_clause] tz('<time_zone>')
```

example:

```
> SELECT "water_level" FROM "h2o_feet" WHERE "location" = 'santa_monica' AND
time >= '2015-08-18T00:00:00Z' AND time <= '2015-08-18T00:18:00Z'
tz('America/Chicago')

name: h2o_feet
time                        water_level
----                        -----------
2015-08-17T19:00:00-05:00   2.064
2015-08-17T19:06:00-05:00   2.116
2015-08-17T19:12:00-05:00   2.028
2015-08-17T19:18:00-05:00   2.126
```

2.9 Time syntax

语法：

```
SELECT_clause FROM_clause WHERE time <operator> ['<rfc3339_date_time_string>' |
'<rfc3339_like_date_time_string>' | <epoch_time>] [AND
['<rfc3339_date_time_string>' | '<rfc3339_like_date_time_string>' |
<epoch_time>] [...]]
```

InfluxDB不再支持在`WHERE`的绝对时间里面使用`OR`。

rfc3399时间字符串

```
'YYYY-MM-DDTHH:MM:SS.nnnnnnnnnZ'
```

`.nnnnnnnnn`是可选的，如果没有的话，默认是`.00000000`,rfc3399格式的时间字符串要用单引号引起来

epoch_time
Epoch时间是1970年1月1日星期四00:00:00（UTC）以来所经过的时间。默认情况下，InfluxDB假定所有epoch时间戳都是纳秒。也可以在epoch时间戳的末尾包括一个表示时间精度的字符，以表示除纳秒以外的精度

基本算术
所有时间戳格式都支持基本算术。用表示时间精度的字符添加或减去一个时间。

注意，InfluxQL中 +或- 之间用空格隔开。

example:

```
> SELECT "water_level" FROM "h2o_feet" WHERE "location" = 'santa_monica' AND
time >= '2015-08-18T00:00:00.000000000Z' AND time <= '2015-08-18T00:12:00Z'

name: h2o_feet
time                   water_level
----                   -----------
2015-08-18T00:00:00Z   2.064
2015-08-18T00:06:00Z   2.116
```

```
2015-08-18T00:12:00Z   2.028
//这个有点没理解
> SELECT "water_level" FROM "h2o_feet" WHERE "location" = 'santa_monica' AND
time >= 1439856000000000000 AND time <= 1439856720000000000

name: h2o_feet
time                   water_level
----                   -----------
2015-08-18T00:00:00Z   2.064
2015-08-18T00:06:00Z   2.116
2015-08-18T00:12:00Z   2.028

// 对RFC3339格式的时间戳的基本计算
> SELECT "water_level" FROM "h2o_feet" WHERE time > '2015-09-18T21:24:00Z' + 6m
// 对epoch时间戳的基本计算
> SELECT "water_level" FROM "h2o_feet" WHERE time > 24043524m - 6m
```

相对时间

使用`now()`查询的时间戳是相对于服务器当前时间戳的数据

语法：

```
SELECT_clause FROM_clause WHERE time <operator> now() [[ - | + ]
<duration_literal>] [(AND|OR) now() [...]]
```

`now()`是在该服务器上执行查询时服务器的Unix时间。`-`或`+`和时间字符串之间需要空格

example:

```
// 例一：用相对时间指定时间间隔
> SELECT "water_level" FROM "h2o_feet" WHERE time > now() - 1h

// 例二：用绝对和相对时间指定时间间隔
> SELECT "level description" FROM "h2o_feet" WHERE time > '2015-09-18T21:18:00Z'
AND time < now() + 1000d
```

2.10 正则表达式

InluxDB支持在以下场景使用正则表达式：

- 在`SELECT`中的field key和tag key；
- 在`FROM`中的measurement
- 在`WHERE`中的tag value和字符串类型的field value
- 在`GROUP BY`中的tag key

InfluxQL不支持在`WHERE`中使用正则表达式去匹配不是字符串的field value，以及数据库名和retention policy。

语法：

```
SELECT /<regular_expression_field_key>/ FROM /<regular_expression_measurement>/
WHERE [<tag_key> <operator> /<regular_expression_tag_value>/ | <field_key>
<operator> /<regular_expression_field_value>/] GROUP BY
/<regular_expression_tag_key>/
```

支持的操作符：

`=~` 匹配
`!~` 不匹配

example:

```
// 查询所有包含“l”的 field keys和 tag keys
> SELECT /l/ FROM "h2o_feet" LIMIT 1

// 查询包含level单词的无重复field key和tag key
> SELECT DISTINCT(/level/) FROM "h2o_feet" WHERE "location" = 'santa_monica' AND
time >= '2015-08-18T00:00:00.000000000Z' AND time <= '2015-08-18T00:12:00Z'

// 从包含temperature单词的 measurement中查询数据
> SELECT MEAN("degrees") FROM /temperature/
name: average_temperature
time            mean
----            ----
1970-01-01T00:00:00Z   79.98472932232272

name: h2o_temperature
time            mean
----            ----
1970-01-01T00:00:00Z   64.98872722506226
//查询location中包含m字符并且water_level大于3的water_level平均值
> SELECT MEAN(water_level) FROM "h2o_feet" WHERE "location" =~ /[m]/ AND
"water_level" > 3

// 不返回任何值
> SELECT * FROM "h2o_feet" WHERE "location" !~ /./
```

数据类型和cast(转换)

Field values可以是 floats, integers, strings, or booleans。 用`::`语法可以指定查询时field的类型

语法：

```
SELECT_clause <field_key>::<type> FROM_clause
```

example:

```
> SELECT "water_level"::float FROM "h2o_feet" LIMIT 4
name: h2o_feet
--------------
time                   water_level
2015-08-18T00:00:00Z   8.12
2015-08-18T00:00:00Z   2.064
2015-08-18T00:06:00Z   8.005
2015-08-18T00:06:00Z   2.116
```

Cast 操作

`::` 执行基本的cast操作,目前，InfluxDB支持将field值从 integers 转换 floats或者从 floats转换成 integers.

语法：

```
SELECT_clause <field_key>::<type> FROM_clause
```

type:float或integer

example:

```
// floats转换成 integer
> SELECT "water_level"::integer FROM "h2o_feet" LIMIT 4

name: h2o_feet
--------------
time                   water_level
2015-08-18T00:00:00Z   8
2015-08-18T00:00:00Z   2
2015-08-18T00:06:00Z   8
2015-08-18T00:06:00Z   2
//不支持的转换类型，无数据返回
> SELECT "water_level"::string FROM "h2o_feet" LIMIT 4
>
```

2.1.11 子查询

子查询是嵌套在另一个查询的`FROM`子句中的查询。使用子查询将查询作为条件应用于其他查询。子查询提供与嵌套函数和SQL`HAVING`子句类似的功能。

语法：

```
SELECT_clause FROM ( SELECT_clause FROM ( SELECT_statement ) [...] ) [...]
```

example:

```
> SELECT SUM("max") FROM (SELECT MAX("water_level") FROM "h2o_feet" GROUP BY
"location")

name: h2o_feet
time                   sum
----                   ---
1970-01-01T00:00:00Z   17.169
//内嵌的查询
> SELECT MAX("water_level") FROM "h2o_feet" GROUP BY "location"
name: h2o_feet
tags: location=coyote_creek
time                   max
----                   ---
2015-08-29T07:24:00Z   9.964

name: h2o_feet
tags: location=santa_monica
time                   max
----                   ---
2015-08-29T03:54:00Z   7.205
```