ALIBABA CLOUD

阿里云

时序时空数据库
时序数据库 InfluxDB® 版

文档版本：20210112

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.FAQ

本页面讨论了经常容易混淆的内容，以及跟其它数据库系统相比TSDB For InfluxDB®以意想不到的方式运行的地方。

管理（Administration）

- 如何识别TSDB For InfluxDB®的版本？
- shard group duration和保留策略之间的关系是什么？
- 当更改保留策略后，为什么数据没有丢失？
- 为什么TSDB For InfluxDB®无法解析微秒单位？

命令行界面（CLI）

- 如何使TSDB For InfluxDB®的CLI返回用户可读的时间戳？
- 非admin用户如何使用 `USE` 指定一个数据库？
- 如何使用TSDB For InfluxDB®的CLI将数据写入一个非默认的保留策略

数据类型

- 为什么不能查询布尔类型的field value？
- TSDB For InfluxDB®如何处理多个shard之间的field的类型差异？
- TSDB For InfluxDB®可以存储的最小和最大整数是多少？
- TSDB For InfluxDB®可以存储的最小和最大时间戳是多少？
- 如何知道存储在field中的数据类型？
- 是否可以改变field的数据类型？

InfluxQL函数

- 如何执行函数中的数学运算？
- 为什么查询将epoch 0作为时间戳返回？
- 哪些InfluxQL函数支持嵌套使用？

查询数据

- 什么决定了 `GROUP BY time()` 查询返回的时间间隔？
- 为什么查询没有返回任何数据或者只返回一部分数据？
- 为什么 `GROUP BY time()` 查询不返回发生在 `now()` 之后的时间戳？
- 是否可以对时间戳执行数学运算？
- 是否可以从返回的时间戳中识别写入精度？
- 当查询数据时，什么时候应该使用单引号，什么时候应该使用双引号？
- 为什么在创建一个新的默认（ `DEFAULT` ）保留策略后会丢失数据？
- 为什么带有 `WHERE OR` 时间子句的查询返回空结果？
- 为什么 `fill(previous)` 返回空结果？

- 为什么 `INTO` 查询会丢失数据?
- 如何查询tag key和field key名字相同的数据?
- 如何跨measurement查询数据?
- 时间戳的顺序是否重要?
- 如何 `SELECT` 有tag但没有tag value的数据?

序列和序列基数

- 为什么序列基数很重要?

写入数据

- 如何写入整型的field value?
- TSDB For InfluxDB®如何处理重复数据点?
- HTTP API需要怎样的换行符?
- 当将数据写入TSDB For InfluxDB®时，应该避免哪些文字和字符?
- 当写入数据时，什么时候应该使用单引号，什么时候应该使用双引号?
- 时间戳的精度是否重要?

如何识别TSDB For InfluxDB®的版本?

有多种方法可以识别您正在使用的TSDB For InfluxDB®版本:

curl路径/ping

```
$ curl -i 'https://<网络地址>:3242/ping?u=<账号名称>&p=<密码>'
HTTP/1.1204NoContent
Content-Type: application/json
X-Influxdb-Build: OSS
X-Influxdb-Version:1.7.x
```

启动TSDB For InfluxDB®的命令行界面:

```
$ influx -ssl -username <账号名称>-password <密码>-host <网络地址>-port 3242

Connected to https://<网络地址>:3242 version 1.7.x
```

shard group duration和保留策略之间的关系是什么?

TSDB For InfluxDB®将数据存储在shard group。一个shard group覆盖一个特定的时间间隔；TSDB For InfluxDB®通过查看相关保留策略的 `DURATION` 来确定时间间隔。下表列出了RP的 `DURATION` 和一个shard group的时间间隔之间的默认关系:

RP持续时间（duration）	Shard group时间间隔
< 2 days	1 hour
>= 2 days and <= 6 months	1 day
> 6 months	7 days

使用 `SHOW RETENTION POLICIES` 查看保留策略的shard group duration。

当更改保留策略后，为什么数据没有丢失？

有几个原因可以解释为什么保留策略改变后数据没有马上丢失。

第一个也是最有可能的原因是，默认情况下，TSDB For InfluxDB®每30分钟检查并强制执行一次RP。您可能需要等到下一次RP检查，TSDB For InfluxDB®才能删除在RP的新 `DURATION` 之外的数据。

第二个可能的原因是，更改RP的 `DURATION` 和 `SHARD DURATION` 会导致意外的数据保留。TSDB For InfluxDB®将数据存储在shard group，每个shard group覆盖一个特定的RP和时间间隔。当TSDB For InfluxDB®强制执行RP时，整个shard group的数据会被删除，而不是单个数据点。TSDB For InfluxDB®不能拆分shard group。

如果RP的新 `DURATION` 小于旧的 `SHARD DURATION` ，并且TSDB For InfluxDB®正在将数据写入一个旧的、 `DURATION` 较长的shard group，那么系统将强制把所有数据存储在该shard group中，即使该shard group中有些数据已经在新的 `DURATION` 之外。一旦shard group中所有数据都在新的 `DURATION` 之外，TSDB For InfluxDB®将会删除整个shard group，然后系统开始将数据写入具有新的、更短 `SHARD DURATION` 的shard group，避免进一步意想不到的数据保留。

为什么TSDB For InfluxDB®无法解析微秒单位？

在不同场景下：写入、查询以及在TSDB For InfluxDB®命令行界面（CLI）设置精度，用于指定微秒时间单位的语法也不同。下表显示了每个类别支持的语法：

	使用HTTP API写入数据	所有查询	在CLI设置精度
u	√	√	√
us	□	□	□
μ	□	√	□
μs	□	□	□

如何使TSDB For InfluxDB®的CLI返回用户可读的时间戳？

在您首次连接CLI时，请指定rfc3339精度：

```
$ influx -ssl -username <账号名称>-password <密码>-host <网络地址>-port 3242-precision rfc3339
```

或者，在连接CLI后指定精度：

```
$ influx -ssl -username <账号名称>-password <密码>-host <网络地址>-port 3242
Connected to https://<网络地址>:3242 version 1.7.x
> precision rfc3339
>
```

请查看文档命令行界面了解更多有用的CLI选项。

非admin用户如何使用 USE 指定一个数据库？

如果非admin用户拥有数据库的 READ 和/或 WRITE 权限，那么可以执行 USE <database_name> 语句。如果非admin用户尝试用 USE 指定一个他们没有 READ 和/或 WRITE 权限的数据库，那么系统会返回错误：

```
ERR:Database<database_name> doesn't exist. Run SHOW DATABASES for a list of existing databases.
```

> 说明
>
> 注释：SHOW DATABASES查询只返回那些非admin用户有READ和/或WRITE权限的数据库。

如何使用TSDB For InfluxDB®的CLI将数据写入一个非默认的保留策略

请使用语法 INSERT INTO [<database>.]<retention_policy> <line_protocol> 将数据写入一个非默认的保留策略。（只允许在CLI中使用这种方式指定数据库和保留策略。如果通过HTTP写入数据，必须使用参数 db 和 rp 分别指定数据库和保留策略，指定保留策略是可选的。）

示例：

```
> INSERT INTO one_day mortality bool=true
Using retention policy one_day
> SELECT * FROM "mydb"."one_day"."mortality"
name: mortality
---------------
time                bool
2016-09-13T22:29:43.229530864Z  true
```

请注意，您需要完全限定measurement来查询非默认保留策略中的数据。使用以下语法完全限定measurement：

```
"<database>"."<retention_policy>"."<measurement>"
```

为什么不能查询布尔类型的field value?

写入和查询布尔值的语法不一样。

布尔值语法	写入	查询
t，f	√	□
T，F	√	□
true，false	√	√
True，False	√	√
TRUE，FALSE	√	√

例如，`SELECT * FROM "hamlet" WHERE "bool"=True` 返回所有 `bool` 等于 `TRUE` 的数据点，但是，`SELECT * FROM "hamlet" WHERE "bool"=T` 不会返回任何结果。

TSDB For InfluxDB®如何处理多个shard之间的field的类型差异?

field value可以是浮点数、整数、字符串或者布尔值。在一个shard里面，field value的数据类型不能不一样，但是在不同的shard，field value的数据类型可以不同。

SELECT语句

`SELECT` 语句返回所有的field value**如果**这些值有相同的数据类型。如果在不同的shard里面field value的数据类型不一样，那么TSDB For InfluxDB®首先会执行类型转换（如果适用的话），然后返回所有值，并且按以下数据类型的顺序返回结果：浮点数，整数，字符串，布尔值。

如果在您的数据中，field value的类型不同，请使用语法 `<field_key>::<type>` 查询不同的数据类型。

示例

measurement `just_my_type` 有一个名为 `my_field` 的field，`my_field` 在四个不同的shard中有四个field value，并且每个field value的数据类型不一样（分别是浮点数、整数、字符串和布尔值）。

`SELECT *` 只返回浮点型和整型的field value。请注意，在返回结果中TSDB For InfluxDB®强制将整数转换成浮点数。

```
> SELECT * FROM just_my_type

name: just_my_type
------------------
time                my_field
2016-06-03T15:45:00Z9.87034
2016-06-03T16:45:00Z7
```

`SELECT <field_key>::<type> [...]` 返回所有数据类型。TSDB For InfluxDB®将每种类型的数据输出到单独的列中，并且使用递增的列名表示。在可能的情况下，TSDB For InfluxDB®将field value转换成另一种数据类型；它将整数 `7` 转换成第一列中的浮点数，将浮点数 `9.879034` 转换成第二列中的整数。TSDB For InfluxDB®不能将浮点数或整数转换成字符串或布尔值。

```
> SELECT "my_field"::float,"my_field"::integer,"my_field"::string,"my_field"::boolean FROM just_my_type

name: just_my_type
------------------
time                my_field  my_field_1  my_field_2  my_field_3
2016-06-03T15:45:00Z9.870349
2016-06-03T16:45:00Z77
2016-06-03T17:45:00Z                a string
2016-06-03T18:45:00Z                          true
```

SHOW FIELD KEYS查询

`SHOW FIELD KEYS` 返回field key对应的每个shard中的每种数据类型。

示例

measurement `just_my_type` 有一个名为 `my_field` 的field， `my_field` 在四个不同的shard中有四个field value，并且每个field value的数据类型不一样（分别是浮点数、整数、字符串和布尔值）。

`SHOW FIELD KEYS` 返回所有四种数据类型：

```
> SHOW FIELD KEYS

name: just_my_type
fieldKey  fieldType
-----------------
my_field  float
my_field  string
my_field  integer
my_field  boolean
```

TSDB For InfluxDB®可以存储的最小和最大整数是多少？

TSDB For InfluxDB®将所有整数存储为有符号的int64数据类型。int64有效的最小和最大值分别是 `-9223372036854775808` 和 `9223372036854775807` 。请查看Go builtins获得更多相关信息。

使用接近最小/最大整数但依旧在限制范围内的值可能会导致非预期的结果；有些函数和运算符会在计算过程中将数据类型int64转换成float64，这会引起溢出问题。

TSDB For InfluxDB®可以存储的最小和最大时间戳是多少？

最小的时间戳是 `-9223372036854775806` 或 `1677-09-21T00:12:43.145224194Z` ，最大的时间戳是 `9223372036854775806` 或 `2262-04-11T23:47:16.854775806Z` 。超出该范围的时间戳会返回解析错误。

如何知道存储在field中的数据类型？

`SHOW FIELD KEYS` 查询还返回field的数据类型。

示例

```
> SHOW FIELD KEYS FROM all_the_types
name: all_the_types
--------------------
fieldKey  fieldType
blue    string
green    boolean
orange    integer
yellow    float
```

是否可以改变field的数据类型？

目前，在改变field的数据类型上面，TSDB For InfluxDB®提供非常有限的支持。语法 `<field_key>::<type>` 支持将field value从整数转换为浮点数或者从浮点数转换为整数。请查看文档**数据探索**获得更多关于转换操作的信息。无法将浮点数或整数转换为字符串或布尔值（反之亦然）。

我们列出了可用于更改field数据类型的方法：

将数据写入一个不同的field

最简单的解决方法就是将具有新数据类型的数据写入到同一个序列中的不同field。

使用shard系统

在一个shard里面，field value的数据类型不能不一样，但是在不同的shard，field value的数据类型可以不同。

如果要更改field的数据类型，用户可以使用 `SHOW SHARDS` 查询来识别当前shard的 `end_time` 。如果数据点的时间戳发生在 `end_time` 之后，那么TSDB For InfluxDB®允许将不同数据类型的数据写入到一个现有的field中（例如，field原来接受整数，但是在 `end_time` 之后，该field可以接受浮点数）。

请注意，这不会改变原来shard里面field的数据类型。

如何执行函数中的数学运算？

目前，TSDB For InfluxDB®不支持函数内的数学运算。我们建议使用子查询作为解决方法。

示例

InfluxQL不支持以下语法：

```
SELECT MEAN("dogs"-"cats")from"pet_daycare"
```

相反，我们可以使用子查询获得相同的结果：

```
> SELECT MEAN("difference") FROM (SELECT "dogs"-"cat" AS "difference" FROM "pet_daycare")
```

请查看文档**数据探索**获得更多关于子查询的信息。

为什么查询将epoch 0作为时间戳返回？

在TSDB For InfluxDB®中，epoch 0（ `1970-01-01T00:00:00Z` ）通常用作空时间戳（null timestamp），如果您请求的查询中没有时间戳返回，例如，对于没有规定时间范围的聚合函数，TSDB For InfluxDB®返回epoch 0作为时间戳。

哪些InfluxQL函数支持嵌套使用？

以下InfluxQL函数支持嵌套使用：

- `COUNT()` 嵌套 `DISTINCT()`
- `CUMULATIVE_SUM()`
- `DERIVATIVE()`
- `DIFFERENCE()`
- `ELAPSED()`
- `MOVING_AVERAGE()`
- `NON_NEGATIVE_DERIVATIVE()`
- `HOLT_WINTERS()` 和 `HOLT_WINTERS_WITH_FIT()`

关于如何使用子查询代替嵌套函数，请查看文档**数据探索**。

什么决定了 `GROUP BY time()` 查询返回的时间间隔？

`GROUP BY time()` 查询返回的时间间隔符合TSDB For InfluxDB®的预设时间段或者符合用户指定的偏移间隔。

示例

预设时间段

以下查询计算 `sunflowers` 在6:15pm到7:45pm之间的平均值，并将这些平均值按一小时进行分组：

```
SELECT mean("sunflowers")
FROM "flower_orders"
WHERE time >='2016-08-29T18:15:00Z' AND time <='2016-08-29T19:45:00Z' GROUP BY time(1h)
```

下面的结果展示了TSDB For InfluxDB®如何维护它的预设时间段。

在这个示例中，6pm是一个预设的时间段，7pm也是一个预设的时间段。由于 `WHERE` 子句中指定了查询的时间范围，所以在计算6pm时间段对应的平均值时不包括在6:15pm之前的数据，但是用于计算6pm时间段平均值的数据必须发生在6pm这个小时里。对于7pm时间段也是一样；用于计算7pm时间段平均值的数据必须发生在7pm这个小时里。虚线部分展示了用于计算每个平均值的数据点。

请注意，虽然结果中第一个时间戳是 `2016-08-29T18:00:00Z` ，但是在该时间段的查询结果不包含发生在 `2016-08-29T18:15:00Z` （ `WHERE` 子句中指定的开始时间）之前的数据。

原始数据： 结果：

```
name: flower_orders                    name: flower_orders
————————————----------
time            sunflowers          time            mean
2016-08-29T18:00:00Z342016-08-29T18:00:00Z22.332
|--|2016-08-29T19:00:00Z62.75
2016-08-29T18:15:00Z|28|
2016-08-29T18:30:00Z|19|
2016-08-29T18:45:00Z|20|
|--|
|--|
2016-08-29T19:00:00Z|56|
2016-08-29T19:15:00Z|76|
2016-08-29T19:30:00Z|29|
2016-08-29T19:45:00Z|90|
|--|
2016-08-29T20:00:00Z70
```

偏移间隔

以下查询计算 `sunflowers` 在6:15pm到7:45pm之间的平均值，并将这些平均值按一小时进行分组，同时，该查询还将TSDB For InfluxDB®的预设时间段偏移 `15` 分钟：

```
SELECT mean("sunflowers")
FROM "flower_orders"
WHERE time >='2016-08-29T18:15:00Z' AND time <='2016-08-29T19:45:00Z' GROUP BY time(1h,15m)
---
|
                                        offset interval
```

在这个示例中，用户指定的偏移间隔将TSDB For InfluxDB®的预设时间段前移了 `15` 分钟。现在，6pm时间段的平均值包括在6:15pm和7:15pm之间的数据，7pm时间段的平均值包括在7:15pm和8:15pm之间的数据。虚线部分展示了用于计算每个平均值的数据点。

请注意，现在结果中第一个时间戳是 `2016-08-29T18:15:00Z` ，而不是 `2016-08-29T18:00:00Z` 。

原始数据： 结果：

```
name: flower_orders                    name: flower_orders
————————————----------
time            sunflowers          time            mean
2016-08-29T18:00:00Z342016-08-29T18:15:00Z30.75
|--|2016-08-29T19:15:00Z65
2016-08-29T18:15:00Z|28|
2016-08-29T18:30:00Z|19|
2016-08-29T18:45:00Z|20|
2016-08-29T19:00:00Z|56|
|--|
|--|
2016-08-29T19:15:00Z|76|
2016-08-29T19:30:00Z|29|
2016-08-29T19:45:00Z|90|
2016-08-29T20:00:00Z|70|
|--|
```

为什么查询没有返回任何数据或者只返回一部分数据？

对于为什么查询没有返回任何数据或者只返回一部分数据，有几种可能的解释。我们在下面列出了一些最常见的原因：

保留策略

第一个也是最常见的解释与保留策略（RP）有关。TSDB For InfluxDB®自动从数据库的默认（ `DEFAULT` ）RP中查询数据。如果您的数据不是存储在默认的RP，TSDB For InfluxDB®不会返回任何结果，除非您明确指定所使用的RP。

SELECT子句中的tag key

在 SELECT 子句中，至少需要包含一个field key，查询才会返回数据。如果 SELECT 子句只包含一个或多个tag key，查询会返回空的结果。请查看文档**数据探索**获得更多相关信息。

查询时间范围

另一个可能的解释与查询的时间范围有关。默认情况下，大多数 SELECT 查询涵盖在 1677-09-21 00:12:43.145224194 UTC和 2262-04-11T23:47:16.854775806Z UTC之间的时间范围。 SELECT 查询还包括 GROUP BY time() 子句，但是，它涵盖的时间范围在 1677-09-21 00:12:43.145224194 和 now() 之间。如果您的数据发生在 now() 之后，那么 GROUP BY time() 查询不会覆盖这些发生在 now() 之后的数据。如果查询语句包括 GROUP BY time() 子句，并且有数据发生在 now() 之后，您需要为时间范围提供一个上限。

标识符名字

最后一个常见的解释与schema有关（field和tag有相同的名字）。如果field key和tag key相同，那么在所有查询中优先考虑field。在查询中，你需要使用 ::tag 语法指定tag key。

为什么 GROUP BY time() 查询不返回发生在 now() 之后的时间戳？

大多数 SELECT 语句的默认时间范围在 1677-09-21 00:12:43.145224194 UTC 和 2262-04-11T23:47:16.854775806Z UTC 之间。对于带 GROUP BY time() 子句的 SELECT 语句，默认的时间范围在 1677-09-21 00:12:43.145224194 和 now() 之间。

如果要查询时间戳发生在 now() 之后的数据，带 GROUP BY time() 子句的 SELECT 语句必须在 WHERE 子句中提供一个时间上限。

在下面的示例中，第一个查询涵盖时间戳在 2015-09-18T21:30:00Z 和 now() 之间的数据，第二个查询涵盖时间戳在 2015-09-18T21:30:00Z 和 now() 之后180个星期之间的数据。

```
> SELECT MEAN("boards") FROM "hillvalley" WHERE time >='2015-09-18T21:30:00Z' GROUP BY time(12m) fill(
none)



> SELECT MEAN("boards") FROM "hillvalley" WHERE time >='2015-09-18T21:30:00Z' AND time <= now()+180w
GROUP BY time(12m) fill(none)
```

请注意， WHERE 子句必须提供一个时间上线来覆盖默认的 now() 上限。下面的查询只是将 now() 的下限重置，使得查询的时间范围在 now() 和 now() 之间：

```
> SELECT MEAN("boards") FROM "hillvalley" WHERE time >= now() GROUP BY time(12m) fill(none)
>
```

请查看文档**数据探索**获得更多关于时间语法的信息。

是否可以对时间戳执行数学运算？

目前，在TSDB For InfluxDB®中，不能对时间戳执行数学运算。更多关于时间的计算必须由接收查询结果的客户端执行。

对时间戳使用InfluxQL函数，TSDB For InfluxDB®仅提供有限的支持。ELAPSED()函数返回单个field中时间戳之间的差值。

是否可以从返回的时间戳中识别写入精度？

不管提供的写入精度是多少，TSDB For InfluxDB®将所有时间戳存储为纳秒。需要注意的一个重要事项是，当返回查询结果时，数据库会不动声色地删除时间戳后面的零，使原始的写入精度很难识别。

在下面的示例中，tag `precision_supplied` 和 `timestamp_supplied` 分别显示了用户在写入数据时提供的时间精度和时间戳。因为TSDB For InfluxDB®默默地将返回的时间戳后面的零删除了，所以从返回的时间戳中很难识别写入精度。

```
name: trails
------------
time                value  precision_supplied  timestamp_supplied
1970-01-01T01:00:00Z3   n           3600000000000
1970-01-01T01:00:00Z5   h           1
1970-01-01T02:00:00Z4   n           7200000000000
1970-01-01T02:00:00Z6   h           2
```

当查询数据时，什么时候应该使用单引号，什么时候应该使用双引号？

用单引号将字符串类型的值括起来（例如，tag value），但是不要用单引号将标识符（数据库名字、保留策略名字、用户名、measurement的名字、tag key和field key）括起来。

如果标识符以数字开头，或包含除 `[A-z,0-9,_]` 外的字符，或者标识符是InfluxQL关键字，那么需要使用双引号将标识符括起来。如果标识符不属于这些类别之一，可以不需要使用双引号将它们括起来，但是我们还是建议用双引号将它们括起来。

示例：

合法的查询：`SELECT bikes_available FROM bikes WHERE station_id='9'`

合法的查询：`SELECT "bikes_available" FROM "bikes" WHERE "station_id"='9'`

合法的查询：`SELECT MIN("avgrq-sz") AS "min_avgrq-sz" FROM telegraf`

合法的查询：`SELECT * from "cr@zy" where "p^e"='2'`

非法的查询：`SELECT 'bikes_available' FROM 'bikes' WHERE 'station_id'="9"`

非法的查询：`SELECT * from cr@zy where p^e='2'`

用单引号将日期时间字符串括起来。如果您使用双引号将日期时间字符串括起来，TSDB For InfluxDB®会返回错误（ `ERR: invalid operation: time and *influxql.VarRef are not compatible` ）。

示例：

合法的查询：`SELECT "water_level" FROM "h2o_feet" WHERE time > '2015-08-18T23:00:01.232000000Z' AND time < '2015-09-19'`

非法的查询：`SELECT "water_level" FROM "h2o_feet" WHERE time > "2015-08-18T23:00:01.232000000Z" AND time < "2015-09-19"`

请查看文档**数据探索**获得更多关于时间语法的信息。

为什么在创建一个新的默认（ DEFAULT ）保留策略后会丢失数据？

当您在数据库中创建一个新的默认保留策略（RP）后，在旧的默认RP中的数据依旧保存在旧的RP中。对于不指定RP的查询，将会自动查询新默认RP中的数据，所有旧数据可能会丢失。为了查询旧数据，必须完全限定查询中的数据。

示例：

在measurement fleeting 中的所有数据属于默认的RP，该RP的名字为 one_hour ：

```
> SELECT count(flounders) FROM fleeting
name: fleeting
--------------
time            count
1970-01-01T00:00:00Z8
```

现在我们创建一个新的默认RP（ two_hour ），并执行相同的查询：

```
> SELECT count(flounders) FROM fleeting
>
```

为了查询旧数据，我们必须通过完全限定 fleeting 来指定旧的默认RP：

```
> SELECT count(flounders) FROM fish.one_hour.fleeting
name: fleeting
--------------
time            count
1970-01-01T00:00:00Z8
```

为什么带有 WHERE OR 时间子句的查询返回空结果？

目前，TSDB For InfluxDB®不支持在 WHERE 子句中使用 OR 来指定多个时间范围。如果查询中的 WHERE 子句使用 OR 来指定多个时间范围，那么TSDB For InfluxDB®不会返回任何结果。

示例：

```
> SELECT * FROM "absolutismus" WHERE time ='2016-07-31T20:07:00Z' OR time ='2016-07-31T23:07:17Z'
>
```

为什么 fill(previous) 返回空结果？

如果前一个值在查询的时间范围之外，那么 fill(previous) 不会填充该时间段的值。

在下面的示例中，TSDB For InfluxDB®不会使用时间段 `2016-07-12T16:50:00Z - 2016-07-12T16:50:10Z` 的值填充时间段 `2016-07-12T16:50:20Z - 2016-07-12T16:50:30Z`，因为该查询的时间范围并不包含较早的时间段。

原始数据：

```
> SELECT * FROM "cupcakes"
name: cupcakes
---------------
time                chocolate
2016-07-12T16:50:00Z3
2016-07-12T16:50:10Z2
2016-07-12T16:50:40Z12
2016-07-12T16:50:50Z11
```

`GROUP BY time()` 查询：

```
> SELECT max("chocolate") FROM "cupcakes" WHERE time >='2016-07-12T16:50:20Z' AND time <='2016-07-12
T16:51:10Z' GROUP BY time(20s) fill(previous)
name: cupcakes
---------------
time                max
2016-07-12T16:50:20Z
2016-07-12T16:50:40Z12
2016-07-12T16:51:00Z12
```

为什么 `INTO` 查询会丢失数据？

默认情况下，`INTO` 查询将原始数据中的tag转换成新写入数据的field。这会导致TSDB For InfluxDB®覆盖之前由tag区分的数据点。在所有 `INTO` 查询中加上 `GROUP BY *`，可以将tag保留在新写入的数据中。

请注意，这种方式不适用于使用 `TOP()` 或 `BOTTOM()` 函数的查询。请查看文档InfluxQL函数获得更多关于 `TOP()` 和 `BOTTOM()` 的信息。

示例

原始数据

measurement `french_bulldogs` 包含一个tag `color` 和一个field `name`。

```
> SELECT * FROM "french_bulldogs"
name: french_bulldogs
---------------------
time                color  name
2016-05-25T00:05:00Z  peach  nugget
2016-05-25T00:05:00Z  grey   rumple
2016-05-25T00:10:00Z  black  prince
```

不使用GROUP BY *的INTO查询

不使用 `GROUP BY *` 子句的 `INTO` 查询将tag `color` 转换成新写入数据中的field。在原始数据中，数据点 `nugget` 和 `rumple` 仅由tag `color` 区分。一旦 `color` 变成field，TSDB For InfluxDB®会认为数据点 `nugget` 和 `rumple` 是重复的，它会用数据点 `rumple` 将数据点 `nugget` 覆盖。

```
> SELECT * INTO "all_dogs" FROM "french_bulldogs"
name: result
------------
time                written
1970-01-01T00:00:00Z3

> SELECT * FROM "all_dogs"
name: all_dogs
--------------
time                color  name
2016-05-25T00:05:00Z  grey   rumple          <---- no more nugget
2016-05-25T00:10:00Z  black  prince
```

使用GROUP BY *的INTO查询

使用 `GROUP BY *` 子句的 `INTO` 查询将tag `color` 保留在新写入的数据中。在这种情况下，数据点 `nugget` 和 `rumple` 依旧是不同的数据点，TSDB For InfluxDB®不会覆盖任何数据。

```
> SELECT "name" INTO "all_dogs" FROM "french_bulldogs" GROUP BY *
name: result
------------
time            written
1970-01-01T00:00:00Z3

> SELECT * FROM "all_dogs"
name: all_dogs
--------------
time            color name
2016-05-25T00:05:00Z peach nugget
2016-05-25T00:05:00Z grey  rumple
2016-05-25T00:10:00Z black prince
```

如何查询tag key和field key名字相同的数据？

使用语法 `::` 指定一个key是field key还是tag key。

示例

示例数据：

```
> INSERT candied,almonds=true almonds=50,half_almonds=511465317610000000000
> INSERT candied,almonds=true almonds=55,half_almonds=561465317620000000000

> SELECT * FROM "candied"
name: candied
-------------
time            almonds almonds_1 half_almonds
2016-06-07T16:40:10Z50    true    51
2016-06-07T16:40:20Z55    true    56
```

指定key是field：

```
> SELECT * FROM "candied" WHERE "almonds"::field >51
name: candied
-------------
time            almonds almonds_1 half_almonds
2016-06-07T16:40:20Z55    true    56
```

指定key是tag：

```
> SELECT * FROM "candied" WHERE "almonds"::tag='true'
name: candied
-------------
time           almonds almonds_1 half_almonds
2016-06-07T16:40:10Z50    true    51
2016-06-07T16:40:20Z55    true    56
```

如何跨measurement查询数据？

目前，无法跨measurement执行数学运算或分组。所有数据必须在同一个measurement下，才能一起查询这些数据。TSDB For InfluxDB®不是一个关系型数据库，跨measurement映射数据目前不是一个推荐的schema。

时间戳的顺序是否重要？

不重要。测试结果表明TSDB For InfluxDB®完成以下查询所需的时间差别非常小：

```
SELECT ... FROM ... WHERE time >'timestamp1' AND time <'timestamp2'
SELECT ... FROM ... WHERE time <'timestamp2' AND time >'timestamp1'
```

如何 SELECT 有tag但没有tag value的数据？

使用 '' 指定一个空的tag value。例如：

```
> SELECT * FROM "vases" WHERE priceless=''
name: vases
-----------
time           origin   priceless
2016-07-20T18:42:00Z8
```

为什么序列基数很重要？

TSDB For InfluxDB®维护系统中每个序列在内存中的索引。随着序列数量不断增加，RAM（内存）使用量也在不断增加。序列基数过大会导致操作系统终止TSDB For InfluxDB®进程，并抛出内存不足（OOM）异常。请查看文档InfluxQL参考了解关于序列基数的InfluxQL命令。

如何写入整型的field value？

当写入整数时，在field value末尾加上 i 。如果您不加上 i ，TSDB For InfluxDB®会把field value当作浮点数。

写入整数： value=100i 写入浮点数： value=100

TSDB For InfluxDB®如何处理重复数据点？

measurement的名字、tag set和时间戳唯一标识一个数据点。如果您提交的数据点跟已有的数据点相比，具有相同measurement、tag set和时间戳，但具有不同field set，那么该数据点的field set会变为旧field set和新field set的并集，如果有任何冲突以新field set为准。这是预期的结果。

例如：

旧数据点：`cpu_load,hostname=server02,az=us_west val_1=24.5,val_2=7 1234567890000000`

新数据点：`cpu_load,hostname=server02,az=us_west val_1=5.24 1234567890000000`

当您提交新数据点后，TSDB For InfluxDB®使用新的field value覆盖 `val_1` 的值， `val_2` 的值继续保留：

```
> SELECT * FROM "cpu_load" WHERE time =1234567890000000
name: cpu_load
--------------
time                az     hostname  val_1  val_2
1970-01-15T06:56:07.89Z  us_west  server02  5.247
```

为了存储这两个数据点，可以：

- 引入新的tag保证唯一性。

  旧数据点：`cpu_load,hostname=server02,az=us_west,uniq=1 val_1=24.5,val_2=7 1234567890000000`

  新数据点：`cpu_load,hostname=server02,az=us_west,uniq=2 val_1=5.24 1234567890000000`

  将新数据点写入TSDB For InfluxDB®后：

```
> SELECT * FROM "cpu_load" WHERE time =1234567890000000
name: cpu_load
--------------
time                az     hostname  uniq  val_1  val_2
1970-01-15T06:56:07.89Z  us_west  server02  124.57
1970-01-15T06:56:07.89Z  us_west  server02  25.24
```

- 时间戳增加一纳秒。

  旧数据点：`cpu_load,hostname=server02,az=us_west val_1=24.5,val_2=7 1234567890000000`

  新数据点：`cpu_load,hostname=server02,az=us_west val_1=5.24 1234567890000001`

  将新数据点写入TSDB For InfluxDB®后：

```
> SELECT * FROM "cpu_load" WHERE time >=1234567890000000 and time <=1234567890000001
name: cpu_load
--------------
time                az     hostname  val_1  val_2
1970-01-15T06:56:07.89Z      us_west  server02  24.57
1970-01-15T06:56:07.890000001Z  us_west  server02  5.24
```

HTTP API需要怎样的换行符？

TSDB For InfluxDB®的行协议依赖换行符（ \n ，这是ASCII 0x0A ）来表示一行的结束和新的一行的开始。文件或数据使用 \n 以外的换行符会导致以下错误： bad timestamp ， unable to parse 。

请注意，Windows使用回车键和换行符（ \r\n ）作为换行符。

当将数据写入TSDB For InfluxDB®时，应该避免哪些文字和字符？

InfluxQL关键字

如果您使用InfluxQL关键字作为标识符，您需要在每个查询中使用双引号将该标识符括起来。如果不使用双引号，会导致错误。标识符是连续查询名字、数据库名字、field key、measurement的名字、保留策略名字、tag key和用户名。

时间

关键字 time 是一个特例。 time 可以是一个连续查询名字、数据库名字、measurement的名字、保留策略名字和用户名。在这些情况下，不需要在查询中用双引号将 time 括起来。 time 不能是field key或tag key；TSDB For InfluxDB®拒绝写入将 time 作为field key或tag key的数据，对于这种数据写入，TSDB For InfluxDB®会返回错误。

示例

将time作为measurement，写入数据并查询它

```
> INSERT time value=1

> SELECT * FROM time

name: time
time                value
---------
2017-02-07T18:28:27.349785384Z1
```

在TSDB For InfluxDB®中， time 是一个有效的measurement名字。

将time作为field key，写入数据并尝试查询它

```
> INSERT mymeas time=1
ERR:{"error":"partial write: invalid field name: input field \"time\" on measurement \"mymeas\" is invalid dr
opped=1"}
```

在TSDB For InfluxDB®中， time 不是一个有效的field key。系统无法写入该数据点，并且返回 400 错误。

将time作为tag key，写入数据并尝试查询它

```
> INSERT mymeas,time=1 value=1
ERR:{"error":"partial write: invalid tag key: input tag \"time\" on measurement \"mymeas\" is invalid droppe
d=1"}
```

在TSDB For InfluxDB®中，`time`不是一个有效的tag key。系统无法写入该数据点，并且返回`400`错误。

字符

为了保持简单的正则表达式和引号，避免在标识符中使用以下字符：`\`反斜杠`^`尖号`$`货币符号`'`单引号`"`双引号`=`等号`,`逗号

当写入数据时，什么时候应该使用单引号，什么时候应该使用双引号？

- 通过行协议写入数据时，避免使用单引号和双引号将标识符括起来；请查看下面的示例，使用引号后的标识符会使查询变得复杂。标识符是连续查询名字、数据库名字、field key、measurement的名字、保留策略名字、subscription的名字、tag key和用户名。

  写入带双引号的measurement：`INSERT "bikes" bikes_available=3` 适用的查询：`SELECT * FROM "\"bikes\""`

  写入带单引号的measurement：`INSERT 'bikes' bikes_available=3` 适用的查询：`SELECT * FROM "\'bikes\'"`

  写入不带引号的measurement：`INSERT bikes bikes_available=3` 适用的查询：`SELECT * FROM "bikes"`

- 用双引号将字符串类型的field value括起来。

  写入：`INSERT bikes happiness="level 2"` 适用的查询：`SELECT * FROM "bikes" WHERE "happiness"='level 2'`

- 应该用反斜杠转义特殊字符，而不是用引号将其括起来。

  写入：`INSERT wacky va\"ue=4` 适用的查询：`SELECT "va\"ue" FROM "wacky"`

请查看文档行协议参考获得更多相关信息。

时间戳的精度是否重要？

重要。为了最大限度地提高性能，向TSDB For InfluxDB®写入数据时尽量使用最粗糙的时间精度。

在下面两个例子中，第一个请求使用默认精度(纳秒)，而第二个请求将精度设置为秒：

```
curl -i -XPOST "https://<网络地址>:3242/write?db=weather&u=<账号名称>&p=<密码>"--data-binary 'temperature,location=1 value=90 1472666050000000000'

curl -i -XPOST "https://<网络地址>:3242/write?db=weather&precision=s&u=<账号名称>&p=<密码>"--data-binary 'temperature,location=1 value=90 1472666050'
```

虽然性能会提高，但是代价是精度越粗糙，越有可能出现具有相同时间戳的重复数据点，可能会覆盖其它数据点。