

IginX SQL 使用文档

数据相关

插入数据

SQL

```
1 INSERT INTO <prefixPath> ((TIMESTAMP|TIME) (, <suffixPath>)+) VALUES (timeValue (, constant)+);
```

使用插入数据语句可以向指定的一条或有公共前缀的多条时间序列中插入数据。

1. 完整路径 <fullPath> = <prefixPath>.<suffixPath>，完整路径对应的时间序列若没有创建则会自动创建
2. time/timestamp 为必填关键字，代表插入数据的时间戳
3. timeValue 为时间戳，包含以下类型
 - a. long
 - b. now() 函数
 - c. yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss
 - d. 基于上述三者的加减 duration 运算表达式，如 now() + 1ms、2021-09-01 12:12:12 - 1ns
 - e. duration支持 Y|M|D|H|M|S|MS|NS 等单位
4. CONSTANT 为具体的数据
 - a. 包括boolean、float、double、int、long、string、空值(NaN、NULL)
 - b. 默认情况下对于整数类型和浮点数类型按照 long 和 double 解析
 - c. 如果想插入 float 和 int 值，应在具体数据后加上 f、i 后缀，如 2.56f、123i

示例

我们向 <test.test_insert.status> 和 <test.test_insert.version> 两条路径分别插入多条数据

SQL

```
1 IginX> INSERT INTO test.test_insert (time, status, version) VALUES
  (1633421948, true, "v1");
2 IginX> INSERT INTO test.test_insert (time, status, version) VALUES
  (1633421949, false, "v2");
3 IginX> INSERT INTO test.test_insert (time, status, version) VALUES
  (1633421950, true, "v3"), (1633421951, false, "v4"), (1633421952, true, "v5");
```

查询我们刚刚插入的结果

SQL

```
1 IginX> SELECT * FROM test.test_insert;
```

删除数据

SQL

```
1 DELETE FROM path (, path)* (WHERE <orExpression>)?;
```

使用删除语句可以删除指定的时间序列中符合时间删除条件的数据。在删除数据时，用户可以选择需要删除的一个或多个时间序列、时间序列的前缀、时间序列带*路径对多个时间区间内的数据进行删除

1. 删除数据语句不能删除路径，只会删除对应路径的制定数据
2. 删除语句不支持精确时间点保留，如 delete from a.b.c where time != 2021-09-01 12:00:00;
3. 删除语句不支持值过滤条件删除，如 delete from a.b.c where a.b.c >= 100;

示例

我们可以用下面语句删除全部序列对应的数据

SQL

```
1 IginX> DELETE FROM *;
```

我们可以用下面语句删除 <test.test_delete.*> 序列在2021-09-01 12:00:00 到 2021-10-01 12:00:00 对应的数据

SQL

```
1 IginX> INSERT INTO test.test_delete (time, status, version) VALUES (2021-09-01
12:22:01, true, "v1"), (2021-09-01 12:36:03, false, "v2"), (2021-11-01 12:00:0
0, true, "v3");
2
3 IginX> DELETE FROM test.test_delete WHERE time >= 2021-09-01 12:00:00 AND time
<= 2021-10-01 12:00:00;
```

查询删除后的结果

SQL

```
1 IginX> SELECT * FROM test.test_delete;
```

查询数据

SQL

```
1  SELECT <expression> (, <expression>)* FROM prefixPath <whereClause>? <specialC
   lause>?
2
3  <expression>
4      : <functionName>(<suffixPath>)
5      | <suffixPath>
6
7  <whereClause>
8      : WHERE TIME IN <timeInterval> (AND <orExpression>)?
9      | WHERE <orExpression>;
10
11 <orExpression>: <andExpression> (OR <andExpression>)*;
12
13 <andExpression>: <predicate> (AND <predicate>)*;
14
15 <specialClause>
16     : <limitClause>
17     | <groupByLevelClause>
18     | <groupByClause> <limitClause>?
19     | <groupByTimeClause> <limitClause>?
20     ;
21
22 <limitClause>
23     : LIMIT INT COMMA INT
24     | LIMIT INT <offsetClause>?
25     | <offsetClause>? LIMIT INT
26     ;
27
28 <offsetClause>: OFFSET INT;
29
30 <groupByClause>: GROUP <timeInterval> BY DURATION COMMA LEVEL OPERATOR_EQ INT
   (COMMA INT)*;
31
32 <groupByTimeClause>: GROUP <timeInterval> BY DURATION;
33
34 <groupByLevelClause>: GROUP BY LEVEL OPERATOR_EQ INT (COMMA INT)*;
35
36 <timeInterval>
37     : (timeValue, timeValue)
38     | (timeValue, timeValue]
39     | [timeValue, timeValue)
40     | [timeValue, timeValue]
```

查询数据语句支持简单范围查询、值过滤查询、聚合查询、降采样查询

- 1. 聚合查询函数，目前支持 FIRST_VALUE、LAST_VALUE、MIN、MAX、AVG、COUNT、SUM 七种
- 2. 降采样查询函数，目前支持 LAST、FIRST_VALUE、LAST_VALUE、MIN、MAX、AVG、COUNT、SUM 八种
- 3. timeRange 为一个连续的时间区间，支持左开右闭、左闭右开、左右同开、左右同闭区间
- 4. 层次聚合只支持 sum、count 和 avg 这三个聚合函数

示例

先插入如下数据

Plain Text					
1	-----				
2	Time	test.test_select.boolean	test.test_select.string	test.test_select.double	test.test_select.long
3	0	true	fq4RUeRjS8	0.5	1
4	1	false	QKzYVQBquj	1.5	2
5	2	true	qdYVJZOYXI	2.5	3
6	3	false	CicZibVAAc	3.5	4
7	4	true	c8Dq337a7d	4.5	5
8	5	false	jjlohugmSi	5.5	6
9	6	true	inCgynQcwN	6.5	7
10				
11	97	false	L5E42AIu4j	97.5	98
12	98	true	gzQQh2oBeg	98.5	99
13	99	false	9CR2VrtRrp	99.5	100

范围查询

查询序列 <test.test_select.string> 和 <test.test_select.double> 在 0-100ms 内的值

SQL	
1	IginX> SELECT string, double FROM test.test_select WHERE time >= 0 AND time <= 100;

值过滤查询

查询序列 <test.test_select.string> 和 <test.test_select.long> 在 0-100ms 内，且 <test.test_select.long> 的值大于 20 的值

SQL

```
1 IginX> SELECT string, long FROM test.test_select WHERE time >= 0 AND time <= 100 AND long > 20;
```

聚合查询

查询序列 <test.test_select.long> 在 0-100ms 内的平均值

SQL

```
1 IginX> SELECT AVG(long) FROM test.test_select WHERE time >= 0 AND time <= 100;
```

降采样查询

查询序列 <test.test_select.double> 在 0-100ms 内，每 10ms 的最大值

SQL

```
1 IginX> SELECT MAX(double) FROM test.test_select GROUP [0, 100] BY 10ms;
```

层次聚合查询

对于原先的结果集按照 level 分组：

例如 a.a.a.a、a.b.a.b、a.a.b.a 和 a.b.b.b 这四个序列

- 如果按照 level = 0 分组，那么则会查出来 1 个序列：a.*.*
- 如果按照 level = 0,1 分组，那么则会查出来 2 个序列：a.a.* 和 a.b.*
- 如果按照 level = 2 分组，那么则会查出来 2 个序列：*.a.* 和 *.b.*
- 如果按照 level = 3 分组，那么则会查出来 2 个序列：***.a 和 ***.b

查询 <test.*.*> 匹配的路径下所有路径的点数

SQL

```
1 IginX> SELECT count(*) FROM test.test_select GROUP BY LEVEL = 0;
```

查询 <test.test_select.*> 匹配的路径下所有路径的点数

SQL

```
1 IginX> SELECT count(*) FROM test.test_select GROUP BY LEVEL = 0,1;
```

数量限制查询

查询序列 <test.test_select.string> 在的前 10 条记录

SQL

```
1 IginX> SELECT string FROM test.test_select LIMIT 10;
```

查询序列 <test.test_select.string> 从第 5 条开始的 10 条记录

SQL

```
1 IginX> SELECT string FROM test.test_select LIMIT 10 OFFSET 5;
```

查询序列

SQL

```
1 SHOW TIME SERIES;
```

查询序列语句可以查询存储的全部序列名和对应的类型

统计数据总量

SQL

```
1 COUNT POINTS;
```

统计数据总量语句用于统计 IginX 中数据总量

清除数据

SQL

```
1 CLEAR DATA;
```

清除数据语句用于删除 IginX 中全部数据和路径

示例

清除数据，并查询清除之后的数据

SQL

```
1 IginX> CLEAR DATA;  
2 IginX> SELECT * FROM *;
```

系统相关

增加存储引擎

SQL

```
1 ADD STORAGEENGINE (ip, port, engineType, extra)+;
```

1. 添加引擎之前先保证示例存在且正确运行
2. engineType 目前只支持 IOTDB 和 INFLUXDB
3. 为了方便拓展 extra 目前是 string 类型，具体为一个 string 类型的 map

示例

添加一个 IOTDB11 实例—127.0.0.1: 6667，用户名密码均为root，连接池数量为130

SQL

```
1 ADD STORAGEENGINE (127.0.0.1, 6667, "iotdb11", "username:root, password:root,  
sessionPoolSize:130");
```

查询副本数量

SQL

```
1 SHOW REPLICA NUMBER;
```

查询副本数量语句用于查询现在 IginX 存储的副本数量

查询集群信息

SQL

```
1 SHOW CLUSTER INFO;
```

查询集群信息语句用于查询 IginX 集群信息，包括 IginX 节点、存储引擎节点、元数据节点信息等

```
IginX> show cluster info
IginX infos:
+-----+
| ID |      IP | PORT |
+-----+
| 0 | 0.0.0.0 | 6888 |
+-----+
Storage engine infos:
+-----+
| ID |      IP | PORT |  TYPE |
+-----+
| 0 | 127.0.0.1 | 6667 | iotdb11 |
| 1 | 127.0.0.1 | 6668 | iotdb12 |
+-----+
Meta Storage infos:
+-----+
|      IP | PORT |  TYPE |
+-----+
| 127.0.0.1 | 2181 | zookeeper |
+-----+
IginX>
```

用户权限管理

SQL

```
1 CREATE USER <username> IDENTIFIED BY <password>;
2 GRANT <permissionSpec> TO USER <username>;
3 SET PASSWORD FOR <username> = PASSWORD(<password>);
4 DROP USER <username>
5 SHOW USER (<username>)*
6
7 <permissionSpec>: (<permission>,+
8
9 <permission>: READ | WRITE | ADMIN | CLUSTER
```

用户权限管理语句用于 IginX 新增用户，更新用户密码、权限，删除用户等

示例

添加一个无任何权限的用户 root1，密码为 root1

SQL

```
1 CREATE USER root1 IDENTIFIED BY root1;
```

授予用户 root1 以读写权限

SQL

```
1 GRANT WRITE, READ TO USER root1;
```

将用户 root1 的权限变为只读

SQL

```
1 GRANT READ TO USER root1;
```

将用户 root1 的密码改为 root2

SQL

```
1 SET PASSWORD FOR root1 = PASSWORD(root2);
```

删除用户root1

SQL

```
1 DROP USER root1
```

展示用户 root2, root3 信息

SQL

```
1 SHOW USER root2, root3;
```

展示所有用户信息

SQL

```
1 SHOW USER;
```

Transform

注册 python 脚本

SQL

```
1 REGISTER PYTHON TASK <className> IN <filePath>;
```

注意：注册语句会对注册脚本的合法性进行校验，包括 filePath 是否指向一个 python 脚本，声明的 class 是否定义在 python 脚本中等

合法的Python脚本示例？

示例

SQL

```
1 REGISTER PYTHON TASK "Sum" IN "data/py_sum.py";
```

删除已注册的 python 脚本

SQL

```
1 DROP PYTHON TASK <className>;
```

示例

SQL

```
1 DROP PYTHON TASK "Sum";
```

展示已注册的 python 脚本

SQL

```
1 SHOW REGISTER PYTHON TASK;
```

示例

SQL

```
1 SHOW REGISTER PYTHON TASK;
```

提交 transform 任务

SQL

```
1 COMMIT TRANSFORM JOB <filePath>
```

注意：filePath 必须指向一个 yaml 文件，且 yaml 中使用的脚本必须已被注册，具体的格式示例为如下

示例

transform/job.yaml

YAML

```
1 taskList:
2   - taskType: iginx
3     dataFlowType: stream
4     timeout: 10000000
5     sql: select value1, value2, value3, value4 from transform;
6   - taskType: python
7     dataFlowType: stream
8     timeout: 10000000
9     className: AddOneTransformer
10  - taskType: python
11    dataFlowType: batch
12    timeout: 10000000
13    className: SumTransformer
14  - taskType: python
15    dataFlowType: stream
16    timeout: 10000000
17    className: RowSumTransformer
18
19 #exportType: none
20 #exportType: iginx
21 exportType: file
22 #exportFile: /path/to/your/output/dir
23 exportFile: /Users/cauchy-ny/Downloads/export_file_sum_sql.txt
```

SQL

```
1 COMMIT TRANSFORM JOB "transform/job.yaml";
```

查询 transform 任务状态

SQL

```
1 SHOW TRANSFORM JOB STATUS <jobId>;
```

示例

SQL

```
1 SHOW TRANSFORM JOB STATUS 12323232;
```