

# 木星时序数据库（JIC-TSDB） 用户手册

目录

- 1. 简介.....7
- 2. 数据市场空间与格局 .....7
- 3. 制造行业趋势分析.....8
- 4. 系统介绍 .....8
- 5. 使用说明 .....9
  - 5.1. 数据写入.....9
  - 5.2. 数据查询.....9
    - 5.2.1. SELECT.....9
    - 5.2.2. WHERE .....10
    - 5.2.3. GROUP BY .....14
    - 5.2.4. ORDER BY .....16
  - 5.3. 数据订阅.....18
    - 5.3.1. 配置项 .....18
    - 5.3.2. 创建订阅.....18
    - 5.3.3. 显示订阅.....19
    - 5.3.4. 删除订阅.....19

1. 简介

mxTsdb 是华龙讯达自主设计、研发的一款面向工业企业的云原生分布式时序数据库，提供单机和分布式版本，具备卓越的读写性能和高效的数据分析能力，支持主流开发语言和多形态部署（如云、Docker、物理机等），存储分析一体化，易扩展。致力于解决工业互联网场景下海量设备数采数据的高效存储和分析，以进一步降低企业运营和运维成本，提升产品质量和生产效率。本文档是针对 OPCUA 时序数据库的操作说明书。

2. 数据市场空间与格局

在数字经济时代，“数据”被誉为“新黄金”。2023 年 10 月 25 日国家数据局正式揭牌，国家数据局是国家发展和改革委员会管理的机构，具体表述为：

组建国家数据局。负责协调推进数据基础制度建设，统筹数据资源整合共享

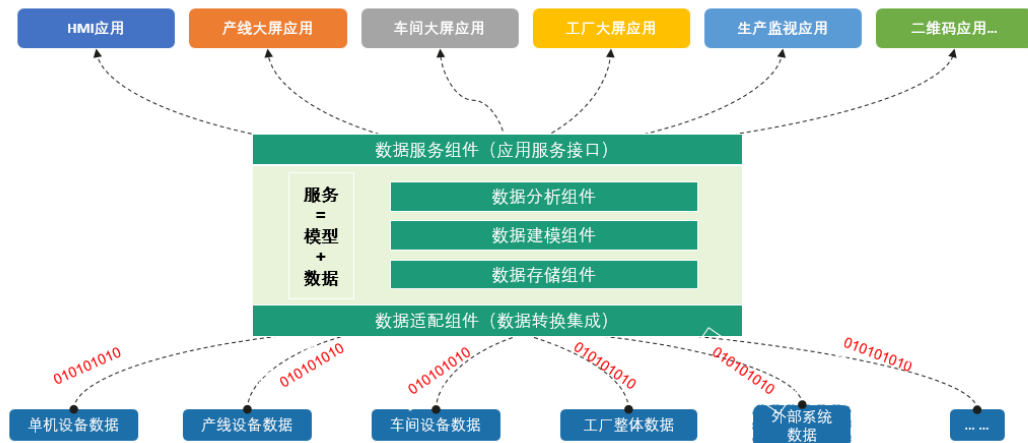
和开发利用，统筹推进数字中国、数字经济、数字社会规划和建设等，由国家发展和改革委员会管理。

近年来，我国数字经济蓬勃发展，成为我国经济增长的新动能。根据国家互联网信息办公室发布的《数字中国发展报告(2022年)》，2022年，我国数据产量达8.1ZB，数字经济规模达50.2万亿元，均居世界第二。

“未来几十年，我们将经历一场由新一代信息技术驱动的深刻的经济社会变革，数字化转型已成为当今世界经济社会发展不可逆转之势。”中国科学院院士梅宏表示，为促进数字经济新形态的有序形成和健康发展，必须夯实数据要素市场、数字治理体系与数据技术体系“三大基石”。

### 3. 制造行业趋势分析

云计算推动数据库向云原生快速演进，也为数据库技术与数字孪生技术的结合提供了强大算力。从云原生(Cloud-Native)到云孪生(Cloud-Twin)，作为数据处理的最核心系统——数据库将开启一个崭新时代。孪生数据库简单说就是对物理世界“镜像”的数字化描述，形象而言就是把物理世界搬进数据库中。



### 4. 系统介绍

JIC-TSDB 木星时序数据库是华龙讯达自主设计、研发的，面向工业企业的一种高性能、低成本、稳定可靠、云原生的时序数据存储及管理服务，具备卓越的读写性能、高效的数据分析能力及实时的数据处理能力，解决了工业企业在构建新型数字化的过程中设备采集点数量巨大、数据采集频率高造成的存储成本高、

写入和查询分析效率低的问题。



## 5. 使用说明

### 5.1. 数据写入

如何将采集到的数据添加到 mxTsdb，在这里将给出简单的示例。

使用 `ts-cli` 命令将数据写入 mxTsdb 数据库，启动命令行界面（CLI）写入相关的数据库，并将 `Insert` 放在 `line protocol` 前面：

```
INSERT weather,location=us-midwest temperature=82 1465839830100400200
```

也可以使用 CLI 从文件导入 Line protocol 数据。

### 5.2. 数据查询

#### 5.2.1. SELECT

`SELECT` 语句执行数据检索。默认情况下，请求的数据将返回给客户端，同时结合 [SELECT INTO](#) 可以被转发到不同的表。

5.2.1.1. 语法

SELECT COLUMN\_CLAUSES FROM\_CLAUSE

5.2.1.2. SELECT 子句

SELECT 子句支持下面的格式:

格式	含义
SELECT *	返回所有的 tag 和 field
SELECT "<field_key>"	返回指定的 field
SELECT "<field_key>","<field_key>"	返回多个指定的 field
SELECT "<field_key>","<tag_key>"	返回指定的 field 和 tag。如果指定了 tag, 则必须指定至少一个 field
SELECT "<field_key>>::field,"<tag_key>>::tag	返回特定 field 和 tag。::[field   tag] 语法指定标识符类型。使用此语法区分具有相同名称的 field 和 tag

5.2.2. WHERE

---

WHERE 子句根据 field、tag、timestamp 来过滤数据。

5.2.2.1. 语法

SELECT COLUMN\_CLAUSES FROM\_CLAUSE WHERE <CONDITION> [(AND|OR)  
<CONDITION> [...]]

TAGS

tag\_key <operator> ['tag\_value']

在 WHERE 子句中，请对 tag value 用单引号括起来。如果 tag value 没有使用引号或者使用了双引号，那么不会返回任何查询结果，在大多数情况下，也不会返回错误。

支持的 operator 有：

操作符	含义
=	等于
<>	不等于
!=	不等于

operator 还支持：正则表达式。

Fields

field\_key <operator> ['string' | boolean | float | integer]

WHERE 子句支持对 field value 进行比较，field value 可以是字符串、布尔值、浮点数或者整数。

在 WHERE 子句中，请对字符串类型的 field value 用单引号括起来。如果字符串类型的 field value 没有使用引号或者使用了双引号，那么不会返回任何查询结果，在大多数情况下，也不会返回错误。

支持的 operator 有：

操作符	含义
=	等于
<>	不等于
!=	不等于
>	大于
>=	大于等于
<	小于
<=	小于等于

operator 还支持：算术运算和正则表达式。

Timestamp

对于大多数 SELECT 语句，默认的时间范围是全部时间范围。对于带 GROUP BY time()子句的 SELECT 语句，默认的时间范围是从时间最小的数据的时间到 now()。

#### 5.2.2.2. 示例

- 查询 field value 满足一定条件的数据

```
> SELECT * FROM "h2o_feet" WHERE "water_level" > 8
```

name: h2o\_feet

```
+-----+-----+-----+-----+
| time                | level description      | location    | water_level |
+-----+-----+-----+-----+
| 1566000000000000000 | between 6 and 9 feet   | coyote_creek | 8.12        |
| 1566000360000000000 | between 6 and 9 feet   | coyote_creek | 8.005       |
| [...]              |                         |              |             |
| 1568679120000000000 | between 6 and 9 feet   | coyote_creek | 8.189       |
| 1568679480000000000 | between 6 and 9 feet   | coyote_creek | 8.084       |
+-----+-----+-----+-----+
4 columns, 1503 rows in set
```

该查询返回 h2o\_feet 中的数据，这些数据满足条件：field key water\_level 的值大于 8。

- 查询 field value 和 tag value 都满足一定条件的数据

```
> SELECT "water_level" FROM "h2o_feet" WHERE "location" <> 'santa_monica' AND (water_level < -0.57 OR water_level > 9.95)
```

name: h2o\_feet

```
+-----+-----+
| time                | water_level |
+-----+-----+
| 1566976680000000000 | 9.957       |
| 1566977040000000000 | 9.964       |
| 1566977400000000000 | 9.954       |
| 1567002240000000000 | -0.587      |
| 1567002600000000000 | -0.61       |
```



```
| 1567002960000000000 | -0.591 |
| 1567091880000000000 | -0.594 |
| 1567092240000000000 | -0.571 |
+-----+-----+
2 columns, 8 rows in set
```

- 查询 **timestamp** 满足一定条件的数据

```
> SELECT * FROM "h2o_feet" WHERE time > now() - 7d
Elapsed: 1.062851ms
```

没有满足过去 7 天内的数据。

### 5.2.3. GROUP BY

---

GROUP BY 子句按用户指定的 **tag** 或者时间区间对查询结果进行分组。

GROUP BY 子句按以下方式对查询结果进行分组：

- 一个或多个指定的 tags
- 指定的时间间隔

#### 注意

不能使用 GROUP BY 对 fields 进行分组

#### 5.2.3.1. 语法

##### GROUP BY TAGS

按标签分组

```
SELECT COLUMN_CLAUSES FROM_CLAUSE [WHERE_CLAUSE] GROUP BY [* |
<tag_key>[,<tag_key>]]
```

## GROUP BY TIME INTERVALS

GROUP BY time()按用户指定的时间间隔对查询结果进行分组,time 和 tag 可以一起分组。

```
SELECT COLUMN_CLAUSES FROM_CLAUSE [WHERE_CLAUSE] GROUP BY
time(<time_interval>),[tag_key] [fill(<fill_option>)]
```

time(time\_interval)

GROUP BY time()子句中的 time\_interval（时间间隔）是一个持续时间（duration），决定了 mxTsdB 按多大的时间间隔将查询结果进行分组。例如，当 time\_interval 为 5m 时，那么在 WHERE 子句中指定的时间范围内，将查询结果按 5 分钟进行分组。

fill()

fill(<fill\_option>)是可选的，它会改变不含数据的时间间隔的返回值。

### 5.2.3.2. 示例

- 按单个 tag 对查询结果进行分组

```
> SELECT MEAN("water_level") FROM "h2o_feet" GROUP BY "location"
```

name: h2o\_feet

tags: location=coyote\_creek

```
+-----+-----+
| time | mean          |
+-----+-----+
| 0    | 5.3591424203039155 |
+-----+-----+
```

2 columns, 1 rows in set

name: h2o\_feet

```
tags: location=santa_monica
+-----+-----+
| time | mean |
+-----+-----+
| 0 | 3.5307120942458803 |
+-----+-----+
2 columns, 1 rows in set
```

- 将查询结果按 12 分钟的时间间隔进行分组

```
> SELECT COUNT("water_level") FROM "h2o_feet" WHERE time >= '2023-08-18T00:00:00Z'
AND time <= '2023-08-18T00:30:00Z' GROUP BY time(12m)
name: h2o_feet
+-----+-----+
| time | count |
+-----+-----+
| 1566086400000000000 | 4 |
| 1566087120000000000 | 4 |
| 1566087840000000000 | 4 |
+-----+-----+
2 columns, 3 rows in set
```

#### 5.2.4. ORDER BY

---

mxTsdb 默认按递增的时间顺序返回结果。第一个返回的数据点，其时间戳是最早的，而最后一个返回的数据点，其时间戳是最新的。ORDER BY time DESC 将默认的时间顺序调转，使得 mxTsdb 首先返回有最新时间戳的数据点，也就是说，按递减的时间顺序返回结果。

ORDER BY 子句仅支持对 time 排序。

##### 5.2.4.1. 语法

```
SELECT COLUMN_CLAUSES FROM_CLAUSE [WHERE_CLAUSE] [GROUP_BY_CLAUSE]
ORDER BY time [ASC|DESC]
```

### 5.2.4.2. 示例

- 首先返回最新的点

```
> SELECT "water_level","location" FROM "h2o_feet" WHERE time >= '2023-08-18T00:00:00Z'
AND time <= '2023-08-18T00:30:00Z' ORDER BY time DESC
```

name: h2o\_feet

time	water_level	location
1566088200000000000	2.267	santa_monica
1566088200000000000	8.012	coyote_creek
1566087840000000000	2.264	santa_monica
1566087840000000000	8.13	coyote_creek
1566087480000000000	2.329	santa_monica
1566087480000000000	8.225	coyote_creek
1566087120000000000	2.343	santa_monica
1566087120000000000	8.32	coyote_creek
1566086760000000000	2.379	santa_monica
1566086760000000000	8.419	coyote_creek
1566086400000000000	2.352	santa_monica
1566086400000000000	8.504	coyote_creek

3 columns, 12 rows in set

- 首先返回最新的点并且包含 GROUP BY time()子句

```
> SELECT COUNT("water_level") FROM "h2o_feet" WHERE time >= '2023-08-18T00:00:00Z'
AND time <= '2023-08-18T00:30:00Z' GROUP BY time(12m) ORDER BY time DESC
```

name: h2o\_feet

time	count
1566087840000000000	4
1566087120000000000	4
1566086400000000000	4

2 columns, 3 rows in set

## 5.3. 数据订阅

数据订阅意味着向一个 mxTsdB 集群发送的写入请求可以被转发到其它订阅结点。接下来的内容将介绍如何使用订阅功能。

### 5.3.1. 配置项

---

配置项包括：

```
[subscriber]
enabled = false
http-timeout = "30s"
insecure-skip-verify = false
https-certificate = ""
write-buffer-size = 100
write-concurrency = 15
```

- enabled: 只有当 enabled 为 true 时，这个功能才会运行
- http-timeout: http 客户端转发请求的超时时间
- insecure-skip-verify: 是否与订阅结点建立不安全的 https 连接
- https-certificate: CA 证书存放路径，如果这一项是空字符串，将会使用系统默认的证书
- write-buffer-size: 写入通道中暂存的未来得及转发的写入请求数量
- write-concurrency: 消费同一个通道的协程数量。默认值是 cpu 数\*2

### 5.3.2. 创建订阅

---

可以在 ts-cli 中使用下列命令创建订阅：

-- Pattern:

```
CREATE SUBSCRIPTION "<subscription_name>" ON "<db_name>". "<retention_policy>"
DESTINATIONS <ALL|ANY> "<subscription_endpoint_host>"
-- 使用默认的 retention policy
CREATE SUBSCRIPTION "<subscription_name>" ON "<db_name>" DESTINATIONS
<ALL|ANY> "<subscription_endpoint_host>"

-- Examples:
-- Create a SUBSCRIPTION on database 'mydb' and retention policy 'autogen' that sends data to
'example.com:9090' via HTTP.
CREATE SUBSCRIPTION "sub0" ON "mydb"."autogen" DESTINATIONS ALL
'http://example.com:9090'

-- Create a SUBSCRIPTION on database 'mydb' and default retention policy that round-robins the
data to 'h1.example.com:9090' and 'h2.example.com:9090' via https.
CREATE SUBSCRIPTION "sub0" ON "mydb" DESTINATIONS ANY
'https://h1.example.com:9090', 'https://h2.example.com:9090'
```

如果订阅节点开启了鉴权，可以通过修改 url 来设置认证：

```
CREATE SUBSCRIPTION "sub0" ON "mydb"."autogen" DESTINATIONS ALL
'http://username:password@example.com:9090'
```

### 5.3.3. 显示订阅

---

显示订阅是非常简单的，只有一种 SQL 语句写法：

```
show subscriptions
```

### 5.3.4. 删除订阅

---

可以在 ts-cli 中使用如下命令删除订阅：

```
--Pattern:
--在指定的 db_name 和 retention_policy 上删除订阅
DROP SUBSCRIPTION "<subscription_name>" ON "<db_name>". "<retention_policy>"
--Example:
DROP SUBSCRIPTION "sub0" ON "db0"."autogen"

--Pattern:
--在 db_name 上找到指定的订阅，并删除
DROP SUBSCRIPTION "<subscription_name>" ON "<db_name>"
--Example:
```

**DROP SUBSCRIPTION** "sub0" **ON** "db0"

--Pattern:

--在指定的 db 上删除所有订阅

**DROP ALL SUBSCRIPTIONS** **ON** "<db\_name>"

--Example:

**DROP ALL SUBSCRIPTIONS** **ON** "db0"

--Pattern:

--删除所有订阅

**DROP ALL SUBSCRIPTIONS**