ALIBABA CLOUD

阿里云

表格存储

命令行工具

文档版本：20230209

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.下载

Tablestore CLI提供简洁、方便的管理命令，支持Windows、Linux和Mac平台。

Tablestore CLI的运行平台以及对应的下载地址请参见下表。

平台	下载地址
Windows	Windows10
Linux	• Linux（AMD64） • Linux（ARM64）
macOS	macOS

2.启动并配置

启动命令行工具后，配置AccessKey或者表格存储（Tablestore）实例等接入信息。

前提条件

已获取AccessKey（包括AccessKey ID和AccessKey Secret）。具体操作，请参见获取AccessKey。

> 说明 如果需要使用RAM用户进行操作，请确保已创建RAM用户并为RAM用户授权。具体操作，请参见配置RAM用户权限。

操作步骤

1. 解压缩下载的工具包，进入命令行工具根目录后，根据所用平台选择相应方式启动命令行工具。
   - 对于Windows平台，双击ts.exe文件。
   - 对于Linux和Mac平台，执行 `./ts` 命令。

     > 说明 如果在Linux系统或者Mac系统下无可执行权限，请执行 `chmod 755 ts` 命令赋权后再启动命令行工具。

   启动界面如下所示。

```
# Welcome to use Command Line Tool for Aliyun Tablestore. Current Version is '2021-11-11'.
#   _______     _    _          _
#  |__   __|   | |  | |        | |
#     | | __ _ | |__| | ___  ___| |_  ___  _ __   ___
#     | |/ _' || '_ \| |/ _ \/ __| __|/ _ \| '__| / _ \
#     | | (_| || |_) | ||  __/\__ \ |_| (_) | |  |  __/
#     |_|\__,_||_.__/|_| \___||___/\__|\___/|_|   \___|
#
# Please visit our product website: https://www.aliyun.com/product/ots
# You can also join our DingTalk Chat Group (ID: 11789671 or 23307953) to discuss and ask Tablest
ore related questions.
#
tablestore>
```

2. 配置接入信息。

   使用已存在的实例配置接入信息，示例如下：

```
config --endpoint https://myinstance.cn-hangzhou.ots.aliyuncs.com --instance myinstance --id NTS
VLeBHzgX2iZfcaXXPJ**** --key 7NR2DiotscDbauohSq9kSHX8BDp99bjs7eNpCR7o****
```

   如果还未创建实例，只需配置AccessKey ID和AccessKey Secret，示例如下：

```
config --id NTSVLeBHzgX2iZfcaXXPJ**** --key 7NR2DiotscDbauohSq9kSHX8BDp99bjs7eNpCR7o****
```

   配置项说明请参见下表。

配置项	是否必填	示例值	说明
--endpoint	否	https://myinstance.cn-hangzhou.ots.aliyuncs.com	实例的服务地址。更多信息，请参见服务地址。如果使用已存在的实例时，才需要配置此项。
--instance	否	myinstance	实例名称。如果使用已存在的实例时，才需要配置此项。
--id	是	NTSVLeBHzgX2iZfcaXXPJ****	阿里云账号或者RAM用户的AccessKey ID和AccessKey Secret。
--key	是	7NR2DiotscDbauohSq9kSHX8BDp99bjs7eNpCR7o****	

3.实例操作

本文介绍如何使用Tablestore CLI以命令行的方式管理表格存储实例。

开通表格存储服务

如果已经开通表格存储服务，请跳过此操作。服务仅需开通一次，开通过程免费。

命令格式如下，用于开通表格存储服务。

```
enable_service
```

返回结果如下：

```
Your service is enabled.
```

创建实例

在指定地域下创建一个实例。

> 重要 实例名需要在地域内全局唯一。如果出现实例名冲突错误，请重新命名。

- 命令格式

  ```
  create_instance -d description -n instanceName -r regionId
  ```

  配置项说明请参见下表。

配置项	是否必填	示例值	描述
-n	是	myinstance	实例名称。更多信息，请参见实例。
-r	是	cn-hangzhou	地域ID。更多信息，请参见地域。
-d	否	"First instance created by CLI."	实例描述信息。

- 示例

  在华东1（杭州）地域下创建一个实例myinstance。

  ```
  create_instance -d "First instance created by CLI." -n myinstance -r cn-hangzhou
  ```

查看实例信息

查看实例的信息，例如实例名称、创建时间、所属账号ID等。

- 命令格式

  ```
  describe_instance -r regionId -n instanceName
  ```

- 示例

查询华东1（杭州）地域下myinstance实例的信息。

```
describe_instance -r cn-hangzhou -n myinstance
```

返回结果如下：

```
{
  "Status": 1,
  "WriteCapacity": 5000,
  "ReadCapacity": 5000,
  "ClusterType": "SSD",
  "Timestamp": "",
  "UserId": "643941",
  "InstanceName": "myinstance",
  "CreateTime": "2021-10-31 14:19:43",
  "Network": "NORMAL",
  "Description": "First instance created by CLI.",
  "Quota": {
    "EntityQuota": 64
  },
  "TagInfos": {
    "TagInfo": []
  }
}
```

获取实例列表

获取指定地域下所有实例的列表。

- 命令格式

```
list_instance -r regionId
```

- 示例

  获取华东1（杭州）地域下所有实例列表。

```
list_instance -r cn-hangzhou
```

  配置项说明请参见下表。

配置项	是否必填	示例值	描述
-r	是	cn-hangzhou	地域ID。

  返回结果如下：

> **说明** 如果当前地域下未创建实例，则返回结果为空。

```
[
  "myinstance"
]
```

配置实例

配置实例的服务地址，请根据所处于的网络类型来选择。

- 命令格式

```
config --endpoint endpoint --instance instanceName
```

配置项说明请参见下表。

配置项	是否必填	示例值	描述
--endpoint	是	http://myinstance.cn-hangzhou.ots.aliyuncs.com	实例的服务地址，支持公网和VPC两种，请根据实际需要选择。域名的规则如下： ○ 公网：http(s)://<instance_name>.<region_id>.ots.aliyuncs.com ○ VPC：http(s)://<instance_name>.<region_id>.vpc.tablestore.aliyuncs.com
--instance	是	myinstance	实例名称。

- 示例

配置华东1（杭州）地域下myinstance实例为公网访问域名。

```
config --endpoint http://myinstance.cn-hangzhou.ots.aliyuncs.com --instance myinstance
```

返回结果如下：

```
{
  "Endpoint": "http://myinstance.cn-hangzhou.ots.aliyuncs.com",
  "AccessKeyId": "NTSVLeBHzgX2iZfcaXXPJ****",
  "AccessKeySecret": "7NR2DiotscDbauohSq9kSHX8BDp99bjs7eNpCR7o****",
  "Instance": "myinstance"
}
```

返回结果说明请参见下表。

配置项	是否必填	示例值	说明
Endpoint	否	https://myinstance.cn-hangzhou.ots.aliyuncs.com	实例的服务地址。更多信息，请参见服务地址。
Instance	否	myinstance	实例名称。
AccessKeyId	是	NTSVLeBHzgX2iZfcaXXPJ****	阿里云账号或者RAM用户的AccessKey ID和AccessKey Secret。
AccessKeySecret	是	7NR2DiotscDbauohSq9kSHX8BDp99bjs7eNpCR7o****	

4.宽表模型

4.1. 数据表操作

创建数据表后，您可以使用表、查询表信息、列出表名称、更新表以及删除表。

> 说明 关于宽表模型的更多信息，请参见宽表模型。

创建表

创建一张数据表，同时指定数据表的主键、数据生命周期（TimeToLive）等。您也可以通过导入JSON格式的配置文件来创建数据表。

命令格式

```
create -t tableName --pk '[{"c":"<primaryKeyName>", "t":"<primaryKeyType>"}, {"c":"<primaryKeyName>", "t":"<primaryKeyType>", "opt":"<options>"}]'--ttl <timeToLive> --version <maxVersion>
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-m，--mode	否	widecolumn	创建的表类型。取值范围如下： • widecolumn（默认）：数据表。 • timeseries：时序表。
-t，--table	是	mytable	数据表名称。
-p，--pk	是	[{"c":"uid","t":"string"}, {"c":"pid","t":"integer"}]	数据表主键列，以JSON格式的数组表示。包含如下字段： • c（必选）：主键列名称。 • t（必选）：主键列类型，取值范围为string、integer、binary。 • opt（可选）：可选配置，取值范围为none和auto，默认值为none。当取值为auto时，表示该主键列为自增列。 关于主键自增列的更多信息，请参见主键列自增。 重要 创建数据表时，属性列不需要定义。表格存储每行的属性列都可以不同，属性列的列名在写入数据时指定。关于写入数据到数据表的具体操作，请参见数据操作。 单表最多可设置4个主键，第一个主键默认为分区键。主键的配置及顺序设置后不能修改。

配置项	是否必填	示例值	说明
--ttl	否	864000	数据表中数据的保存时间。当数据的保存时间超过设置的数据生命周期时，系统会自动清理超过数据生命周期的数据。单位为秒。 如果不设置此项，则默认值为-1，表示数据永不过期。 取值必须大于等于86400秒（一天）或者必须-1（数据永不过期）。
--version	否	1	数据表中的属性列能够保留数据的最大版本个数。当属性列数据的版本个数超过设置的最大版本数时，系统会自动删除较早版本的数据。 如果不设置此项，则默认值为1，表示只保留最新版本的数据。 取值必须为非0整数。
--read_cu	否	0	为数据表配置预留读吞吐量或预留写吞吐量。默认值均为0，即完全按量计费。单位为CU。 容量型实例中的数据表的预留读/写吞吐量只能设置为0，不允许预留。 • 当预留读吞吐量或预留写吞吐量大于0时，表格存储会根据配置为数据表预留相应资源，且数据表创建成功后，将会立即按照预留吞吐量开始计费，超出预留的部分进行按量计费。 • 当预留读吞吐量或预留写吞吐量设置为0时，表格存储不会为数据表预留相应资源。
--write_cu	否	0	
-i，--input	否	/tmp/create_table_meta.json	通过JSON格式的配置文件创建数据表。

您也可以通过配置文件创建表，命令格式如下：

- Windows平台

```
create -i D:\\localpath\\filename.json
```

- Linux和Mac平台

```
create -i /localpath/filename.json
```

配置文件的示例如下：

```
{
  "Name": "mytable",
  "Meta": {
    "Pk": [
      {
        "C": "uid",
        "T": "string",
        "Opt": "none"
      },
      {
        "C": "pid",
        "T": "integer",
        "Opt": "none"
      }
    ]
  },
  "Option": {
    "TTL": 864000,
    "Version": 3
  },
  "CU": {
    "Read": 0,
    "Write": 0
  }
}
```

示例

创建名称为mytable的数据表，该数据表有uid（string类型）和pid（integer类型）两个主键列，表中数据永不过期。

```
create -t mytable --pk '[{"c":"uid", "t":"string"}, {"c":"pid", "t":"integer"}]'
```

创建名称为mytable的数据表，该数据表有uid（string类型）和pid（integer类型）两个主键列并设置第二主键列pid（integer类型）为自增列，表中数据永不过期。

```
create -t mytable --pk '[{"c":"uid", "t":"string"}, {"c":"pid", "t":"integer", "opt":"auto"}]'
```

创建名称为mytable的数据表，该数据表有uid（string类型）和pid（integer类型）两个主键列，表中数据在864000秒（即10天）后过期且只保留最新版本。

```
create -t mytable --pk '[{"C":"uid","t":"string"}, {"c":"pid","t":"integer"}]' --ttl 864000 --version 1
```

使用表

选择需要进行操作的表，用于后续表操作或者数据操作。

命令格式

```
use --wc -t <tableName>
```

配置项说明请参见下表，

配置项	是否必填	示例值	说明
--wc	否	不涉及	表示操作的表为数据表或者索引表。
-t，--table	是	mytable	表名称。

示例

使用mytable数据表。

```
use -t mytable
```

列出表名称

列出实例下的所有表名称、所有数据表名称或者所有时序表名称。

- 列出与当前表类型相同的所有表名称

  ```
  list
  ```

- 列出所有表名称

  ```
  list -a
  ```

- 列出所有数据表名称

  ```
  list -w
  ```

- 列出所有时序表名称

  ```
  list -t
  ```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-a，--all	否	不涉及	列出所有表名称。
-d，--detail	否	不涉及	列出表的详细信息。
-w，--wc	否	不涉及	列出所有数据表名称。
-t，--ts	否	不涉及	列出时序表名称。

更新表

更新数据表的数据生命周期、最大版本数等信息。

命令格式

```
alter -t <tableName> --ttl <timeToLive> --version <maxVersion> --read_cu <readCU> --write_cu <write CU>
```

示例

修改mytable数据表的数据生命周期为86400秒（即1天），最大版本数为1，且预留读CU和预留写CU均为0。

```
alter -t mytable --ttl 86400 --version 1 --read_cu 0 --write_cu 0
```

查看表信息

查看表的信息。您也可以将表信息保存到本地JSON格式的文件中。

命令格式

```
desc -t <tableName> -o /localpath/filename.json
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-t，--table	否	mytable	数据表或者索引表名称。
-f，--print_format	否	json	表信息的输出格式。取值范围为json（默认）和table。
-o，--output	否	/tmp/describe_table_meta.json	输出表信息到本地JSON格式的文件中。

示例

查看当前表的信息。

```
desc
```

查询当前表的信息并将表信息保存到本地文件describe_table_meta.json中。

```
desc -o  /tmp/describe_table_meta.json
```

删除表

删除不需要的表。

命令格式

```
drop -t <tableName> -y
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
--t，--table	是	mytable	数据表名称。
-y，--yes	是	不涉及	显示确认信息。命令中必须带有此配置项。

示例

删除mytable表。

```
drop -t mytable -y
```

4.2. 数据操作

通过命令行工具您可以在数据表中插入新数据、更新一行数据、读取数据、删除一行数据、扫描数据以及导入导出数据。

插入新数据

在表中插入新数据。您也可以通过导入JSON格式的配置文件来插入新数据到表中。

- 命令格式

```
put --pk '[primaryKeyValue, primaryKeyValue]' --attr '[{"c":"attributeColumnName", "v":"attributeColumnValue"}, {"c":"attributeColumnName", "v":"attributeColumnValue", "ts":timestamp}]' --condition condition
```

配置项说明请参见下表。

参数	是否必填	示例值	说明
-p，--pk	是	["86", 6771]	数据表主键的值，以数组表示。 **重要** ◦ 设置的主键个数和类型必须和数据表的主键个数和类型一致。 ◦ 当主键为自增列时，只需将自增列的值设置为占位符null。

参数	是否必填	示例值	说明
--attr	是	[{"c":"name", "v":"redchen"}, {"c":"country", "v":"china", "t":"string", "ts":15327798534}]	数据表属性列，以JSON格式的数组表示。每个属性列包含如下字段： ○ c（必选）：属性列名称。 ○ v（必选）：属性列的值。 ○ t（可选）：属性列类型，取值范围为integer、string（UTF-8编码字符串）、binary、boolean、double五种。当属性列类型为binary时必须设置此字段不可省略。 ○ ts（可选）：时间戳即数据的版本号，可以由系统自动生成或者自定义，如果不设置此参数，则默认由系统自动生成。更多信息，请参见数据版本和生命周期。
--condition	否	ignore	使用条件更新，可以设置原行的存在性条件。取值范围如下： ○ ignore（默认）：表示无论此行是否存在均会插入新数据，如果之前行已存在，则写入数据时会覆盖原有数据。 ○ exist：表示只有此行存在时才会插入新数据，写入数据时会覆盖原有数据。 ○ not_exist：表示只有此行不存在时才会插入数据。 关于条件更新的更多信息，请参见条件更新。
-i, --input	否	/temp/inputdata.json	通过JSON格式的配置文件插入数据。

您也可以通过配置文件插入数据，命令格式如下：

- Windows平台

```
put -i D:\\localpath\\filename.json
```

- Linux和Mac平台

```
put -i /localpath/filename.json
```

配置文件的示例如下：

```
{
  "PK":{
    "Values":[
      "86",
      6771
    ]
  },
  "Attr":{
    "Values":[
      {
        "C":"age",
        "V":32,
        "TS":1626860801604,
        "IsInt":true
      }
    ]
  }
}
```

- 示例

  在数据表中插入一行数据。该行的第一主键列值为“86”，第二主键列值为6771，属性列有name（string类型）和country（string类型）两列。

  ```
  put --pk '["86", 6771]' --attr '[{"c":"name", "v":"redchen"}, {"c":"country", "v":"china"}]'
  ```

  在数据表中插入一行数据，该行的第一主键列值为“86”，第二主键列值为6771，属性列有name（string类型）和country（string类型）两列。无论此行是否存在均会插入新数据，如果之前行已存在，则写入数据时会覆盖原有数据。

  ```
  put --pk '["86", 6771]' --attr '[{"c":"name", "v":"redchen"}, {"c":"country", "v":"china"}]'  --condition ignore
  ```

  在数据表中插入一行数据，该行的第一主键列值为“86”，第二主键列值为6771，属性列有name（string类型）和country（string类型）两列，并且country列的时间戳为15327798534。

  ```
  put --pk '["86", 6771]' --attr '[{"c":"name", "v":"redchen"}, {"c":"country", "v":"china", "t":"string", "ts":15327798534}]'
  ```

  当数据表中第二主键列为自增列时，在数据表中插入一行数据。该行的第一主键列值为“86”，第二主键列值为null，属性列有name（string类型）和country（string类型）两列。

  ```
  put --pk '["86", null]' --attr '[{"c":"name", "v":"redchen"}, {"c":"country", "v":"china"}]'
  ```

读取数据

读取表中的数据。您也可以将读取的数据导出到本地JSON格式的文件中。

> **说明** 如果读取的数据行不存在，则返回结果为空。

- 命令格式

  ```
  get --pk '[primaryKeyValue,primaryKeyValue]'
  ```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-p，--pk	是	["86",6771]	数据表主键的值，以数组表示。 重要 设置的主键个数和类型必须和数据表的主键个数和类型一致。
--columns	否	name,uid	读取的列集合，列名可以是主键列或属性列。如果不设置返回的列名，则返回整行数据。
--max_version	否	1	最多读取的版本数。
--time_range_start	否	1626860469000	读取版本号范围内的数据。time_range_start和time_range_end分别表示起始时间戳和结束时间戳，范围为左闭右开区间。
--time_range_end	否	1626865270000	
--time_range_specific	否	1626862870000	读取特定版本号的数据。
-o, --output	否	/tmp/querydata.json	输出查询结果到本地JSON格式的文件中。

- 示例

  读取第一主键列值为“86”，第二主键列值为6771的行数据。

```
get --pk '["86",6771]'
```

更新一行数据

更新表中的数据。您也可以通过导入JSON格式的配置文件来更新表中数据。

- 命令格式

```
update --pk '[primaryKeyValue, primaryKeyValue]' --attr '[{"c":"attributeColumnName", "v":"attributeColumnValue"}, {"c":"attributeColumnName", "v":"attributeColumnValue", "ts":timestamp}]' --condition condition
```

配置项说明请参见下表。

参数	是否必填	示例值	说明

参数	是否必填	示例值	说明
-p，--pk	是	["86", 6771]	数据表主键的值，以数组表示。 重要 设置的主键个数和类型必须和数据表的主键个数和类型一致。
--attr	是	[{"c":"name", "v":"redchen"}, {"c":"country", "v":"china", "ts":15327798534}]	数据表属性列，以JSON格式的数组表示。每个属性列包含如下字段： ○ c（必选）：属性列名称。 ○ v（必选）：属性列的值。 ○ t（可选）：属性列类型，取值范围为是integer、string（UTF-8编码字符串）、binary、boolean、double五种。当属性列类型为binary时必须设置此字段不可省略。 ○ ts（可选）：时间戳即数据的版本号，可以由系统自动生成或者自定义，如果不设置此参数，则默认由系统自动生成。
--condition	否	ignore	使用条件更新，可以设置原行的存在性条件。取值范围如下： ○ ignore（默认）：表示无论此行是否存在均会插入新数据，如果之前行已存在，则写入数据时会覆盖原有数据。 ○ exist：表示只有此行存在时才会插入新数据，写入数据时会覆盖原有数据。 ○ not_exist：表示只有此行不存在时才会插入数据。 关于条件更新的更多信息，请参见条件更新。
-i, --input	否	/tmp/inputdata.json	通过JSON格式的配置文件更新数据。

您也可以通过配置文件更新数据，命令格式如下：

- Windows平台

```
update -i D:\\localpath\\filename.json
```

- Linux和Mac平台

```
update -i /localpath/filename.json
```

配置文件的示例如下：

```
{
  "PK":{
    "Values":[
      "86",
      6771
    ]
  },
  "Attr":{
    "Values":[
      {
        "C":"age",
        "V":32,
        "TS":1626860801604,
        "IsInt":true
      }
    ]
  }
}
```

- 示例

更新第一主键列为“86”，第二主键列为6771的行数据。无论此行是否存在均会插入新数据，如果之前行已存在，则写入数据时会覆盖原有数据。

```
update --pk '["86", 6771]' --attr '[{"c":"name", "v":"redchen"}, {"c":"country", "v":"china"}]'  --condition ignore
```

删除一行数据

根据主键删除一行数据。

- 命令格式

```
delete --pk '[primaryKeyValue,primaryKeyValue]'
```

- 示例

删除第一主键列值为“86”，第二主键列值为6771的行数据。

```
delete --pk '["86", 6771]'
```

扫描数据

扫描获取整表中所有数据或者获取最多指定个数的行数据。

- 命令格式

```
scan --limit limit
```

配置项说明请见下表。

配置项	是否必填	示例值	说明
--limit	否	10	本次扫描返回的最大行数，可不配置。如果不设置此项，则表示扫描整表中所有数据。

- 示例

  扫描获取数据表中的最多10行数据。

```
scan --limit 10
```

导出数据

导出表中数据到本地JSON文件中。

- 命令格式

```
scan -o /localpath/filename.json -c attributeColumnName,attributeColumnName,attributeColumnName
```

  配置项说明请参见下表。

配置项	是否必填	示例值	说明
-c, --columns	是	uid,name	导出的列集合，列名可以是主键列或属性列。如果不设置列名，则导出整行数据。
--max_version	否	1	最多导出的版本数。
--time_range_start	否	1626865596000	导出版本号范围内的数据。time_range_start和time_range_end分别表示起始时间戳和结束时间戳，范围为前闭后开区间。
--time_range_end	否	1626869196000	
--time_range_specific	否	1626867396000	导出特定版本号的数据。
--backward	否	不涉及	导出数据按照主键降序排列。
-o, --output	是	/tmp/mydata.json	输出查询结果到本地指定路径的JSON格式文件中。
-l, --limit	否	10	本次查询最多返回的行数。
-b, --begin	否	1000	导出范围起始点和结束点之间的数据。
-e, --end	否	2000	

- 示例

导出当前表中全部数据到本地文件mydata.json。

```
scan -o /tmp/mydata.json
```

导出当前表中uid和name列的数据到本地文件mydata.json。

```
scan -o /tmp/mydata.json -c uid,name
```

导入数据

导入本地JSON文件中的数据到表中。

- 命令格式

```
import -i /localpath/filename.json --ignore_version
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-a，--action	否	put	导入数据的模式。取值范围如下： ○ put（默认）：如果某行数据已存在，则先删除原行数据（原行的所有列以及所有版本的数据），再写入新行数据。 ○ update：如果某行数据已存在，可以增加和删除一行中的属性列，删除属性列指定版本的数据，或者更新已存在的属性列的值。如果某行数据不存在，则新增一行数据。
-i, --input	是	/tmp/inputdata.json	通过JSON格式的本地文件导入数据到当前表。
--ignore_version	否	不涉及	忽略时间戳检查，使用当前时间作为时间戳。

本地文件的配置示例如下：

```
{"PK":{"Values":["redchen",0]},"Attr":{"Values":[{"C":"country","V":"china0"},{"C":"name","V":"redchen0"}]}}
{"PK":{"Values":["redchen",1]},"Attr":{"Values":[{"C":"country","V":"china1"},{"C":"name","V":"redchen1"}]}}
```

- 示例

导入mydata.json文件的数据到当前表。

```
import -i /tmp/mydata.json
```

导入mydata.json文件的数据到当前表，且使用当前时间作为时间戳。

```
import -i /tmp/mydata.json --ignore_version
```

4.3. 二级索引

为数据表创建二级索引后，您可以使用索引表、查看索引表信息、使用索引表查询数据以及删除索引表。

前提条件

- 已创建数据表，且数据表的最大版本数（max Versions）必须为1。
- 已为数据表创建预定义列。

创建二级索引

> **说明** 二级索引包括全局二级索引和本地二级索引。更多信息，请参见二级索引简介。

- 命令格式

```
create_index -t <tableName> -n <indexName> -i <indexType> --pk <primaryKeyName,primaryKeyName> --attr <definedColumn,definedColumn>
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-t，--table	否	mytable	数据表名称。
-n，--name	是	index0	二级索引名称。
-i，--index_type	否	global	二级索引类型。取值范围如下： ○ global（默认）：全局二级索引。 使用全局索引时，表格存储以异步方式将数据表中被索引的列和主键列的数据自动同步到索引表中，正常情况下同步延迟达到毫秒级别。 ○ local：本地二级索引。 使用本地二级索引时，表格存储以同步方式将数据表中被索引的列和主键列的数据自动同步到索引表中，当数据写入数据表后，即可从索引表中查询到数据。
-k，--pk	是	uid,pid	索引表的索引列，索引列为数据表主键和预定义列的组合。 使用本地二级索引时，索引表的第一个主键列必须与数据表的第一个主键列相同。
-a，--attr	是	col0,col1	索引表的属性列，索引表属性列为数据表的预定义列的组合。

配置项	是否必填	示例值	说明
-b，--without_base_data	否	不涉及	构建二级索引时不包括数据表中的存量数据。

- 示例

  为数据表mytable创建全局二级索引index0，该二级索引包含存量数据。

  ```
  create_index -t mytable -n index0 -i global --pk pid,uid -a name,col0
  ```

  为数据表mytable创建全局二级索引index1，该二级索引不包含存量数据。

  ```
  create_index -t mytable -n index1 -i global --pk pid,uid -a name,col0 -b
  ```

  为数据表mytable创建本地二级索引index2，该二级索引包含存量数据。

  ```
  create_index -t mytable -n index2 -i local -k uid,name -a col0,col1
  ```

使用表

选择需要进行操作的表，用于后续表操作或者数据操作。

- 命令格式

  ```
  use --wc -t <tableName>
  ```

  配置项说明请参见下表，

配置项	是否必填	示例值	说明
--wc	否	不涉及	表示操作的表为数据表或者索引表。
-t，--table	是	index0	索引表名称。

- 示例

  使用索引表index0。

  ```
  use -t index0
  ```

查看二级索引信息

查看数据表的信息。您也可以将表信息保存到本地JSON格式的文件中。

- 命令格式

  ```
  desc -t <tableName> -o /localpath/filename.json
  ```

  配置项说明请参见下表。

配置项	是否必填	示例值	说明
-t，--table	否	index0	数据表或者索引表名称。
-f，--print_format	否	json	表信息的输出格式。取值范围包括json（默认）和table。
-o，--output	否	/tmp/describe_table_meta.json	输出表信息到本地JSON格式的文件中。

- 示例

  查看当前表的信息。

  ```
  desc
  ```

  查询当前表的信息并将表信息保存到本地文件describe_table_meta.json中。

  ```
  desc -o /tmp/describe_table_meta.json
  ```

使用二级索引查询数据

读取单行数据

读取表中的数据。您也可以将读取的数据导出到本地JSON格式的文件中。

> **说明** 如果读取的数据行不存在，则返回结果为空。

- 命令格式

  ```
  get --pk '[primaryKeyValue,primaryKeyValue]'
  ```

  配置项说明请参见下表。

配置项	是否必填	示例值	说明
-p，--pk	是	["86",6771]	数据表主键的值，以数组表示。 **重要** 设置的主键个数和类型必须和数据表的主键个数和类型一致。
--columns	否	name,uid	读取的列集合，列名可以是主键列或属性列。如果不设置返回的列名，则返回整行数据。
--max_version	否	1	最多读取的版本数。

--time_range_start	否	1626860469000	读取版本号范围内的数据。time_range_start和time_range_end分别表示起始时间戳和结束时间戳，范围为左闭右开区间。
--time_range_end	否	1626865270000	
--time_range_specific	否	1626862870000	读取特定版本号的数据。
-o, --output	否	/tmp/querydata.json	输出查询结果到本地JSON格式的文件中。

- 示例
  读取第一主键列值为“86”，第二主键列值为6771的行数据。

```
get --pk '["86",6771]'
```

扫描数据

扫描获取整表中所有数据或者获取最多指定个数的行数据。

- 命令格式

```
scan --limit limit
```

配置项说明请见下表。

配置项	是否必填	示例值	说明
--limit	否	10	本次扫描返回的最大行数，可不配置。如果不设置此项，则表示扫描整表中所有数据。

- 示例
  扫描获取数据表中的最多10行数据。

```
scan --limit 10
```

导出数据

导出表中数据到本地JSON文件中。

- 命令格式

```
scan -o /localpath/filename.json -c attributeColumnName,attributeColumnName,attributeColumnName
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明

-c, --columns	是	uid,name	导出的列集合，列名可以是主键列或属性列。如果不设置列名，则导出整行数据。
--max_version	否	1	最多导出的版本数。
--time_range_start	否	1626865596000	导出版本号范围内的数据。time_range_start和time_range_end分别表示起始时间戳和结束时间戳，范围为前闭后开区间。
--time_range_end	否	1626869196000	
--time_range_specific	否	1626867396000	导出特定版本号的数据。
--backward	否	不涉及	导出数据按照主键降序排列。
-o, --output	是	/tmp/mydata.json	输出查询结果到本地指定路径的JSON格式文件中。
-l, --limit	否	10	本次查询最多返回的行数。
-b, --begin	否	1000	导出范围起始点和结束点之间的数据。
-e, --end	否	2000	

- 示例
  导出当前表中全部数据到本地文件mydata.json。

```
scan -o /tmp/mydata.json
```

  导出当前表中uid和name列的数据到本地文件mydata.json。

```
scan -o /tmp/mydata.json -c uid,name
```

删除二级索引

删除不需要的索引表。

- 命令格式

```
drop_index -t <tableName> -i <indexName> -y
```

  配置项说明请参见下表，

配置项	是否必填	示例值	说明
-t, --table	否	mytable	数据表名称。
-i, --index	是	index0	二级索引名称。

配置项	是否必填	示例值	说明
-y，--yes	是	不涉及	显示确认信息。命令中必须带有此配置项。

- 示例

  删除当前数据表的索引表index0。

  ```
  drop_index -i index0 -y
  ```

  删除数据表mytable的索引表index0。

  ```
  drop_index -t mytable -i index0 -y
  ```

4.4. 多元索引

为数据表创建多元索引后，您可以查看多元索引列表、查看多元索引信息、使用多元索引查询数据以及删除多元索引。

创建多元索引

创建一个多元索引。

- 命令格式

  ```
  create_search_index -n search_index_name
  ```

  配置项说明请参见下表。

配置项	是否必填	示例值	说明
-n，--name	是	search_index	多元索引名称。
-t，--table	否	mytable	数据表名称。

- 示例

  创建search_index多元索引。

  ```
  create_search_index -n search_index
  ```

  根据系统提示输入索引Schema，示例如下：

  ```
  {
    "IndexSetting": {
      "RoutingFields": null
    },
    "FieldSchemas": [
      {
        "FieldName": "gid",
        "FieldType": "LONG",
        "Index": true,
        "EnableSortAndAgg": true,
        "Store": true,
        "IsArray": false,
  ```

```
      "IsVirtualField": false
    },
    {
      "FieldName": "uid",
      "FieldType": "LONG",
      "Index": true,
      "EnableSortAndAgg": true,
      "Store": true,
      "IsArray": false,
      "IsVirtualField": false
    },
    {
      "FieldName": "col2",
      "FieldType": "LONG",
      "Index": true,
      "EnableSortAndAgg": true,
      "Store": true,
      "IsArray": false,
      "IsVirtualField": false
    },
    {
      "FieldName": "col3",
      "FieldType": "TEXT",
      "Index": true,
      "Analyzer": "single_word",
      "AnalyzerParameter": {
        "CaseSensitive": true,
        "DelimitWord": null
      },
      "EnableSortAndAgg": false,
      "Store": true,
      "IsArray": false,
      "IsVirtualField": false
    },
    {
      "FieldName": "col1",
      "FieldType": "KEYWORD",
      "Index": true,
      "EnableSortAndAgg": true,
      "Store": true,
      "IsArray": false,
      "IsVirtualField": false
    },
    {
      "FieldName": "col3V",
      "FieldType": "LONG",
      "Index": true,
      "EnableSortAndAgg": true,
      "Store": true,
      "IsArray": false,
      "IsVirtualField": true,
      "SourceFieldNames": [
        "col3"
      ]
```

```
    }
  ]
}
```

查看多元索引列表

查看当前数据表下多元索引的列表。

- 命令格式

```
list_search_index
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-a，--all	否	不涉及	显示所有表的多元索引列表。
-d，--detail	否	不涉及	显示多元索引的详细信息。

- 示例

查看当前表下的多元索引列表。

```
list_search_index
```

返回结果如下：

```
+-----------+--------------+
| TableName | IndexName    |
+-----------+--------------+
| mytable   | search_index |
+-----------+--------------+
```

查看多元索引信息

查看多元索引的信息。

- 命令格式

```
describe_search_index -n search_index_name
```

- 示例

查看search_index多元索引的信息。

```
describe_search_index -n search_index
```

使用多元索引查询数据

使用多元索引查询满足指定条件的数据以及对数据进行统计聚合操作。

- 命令格式

```
search -n search_index_name --return_all_indexed
```

- 示例

使用search_index多元索引查询表中数据，并返回所有建立索引的列。

```
search -n search_index --return_all_indexed
```

根据系统提示输入查询条件，示例如下：

```
{
  "Offset": -1,
  "Limit": 10,
  "Collapse": null,
  "Sort": null,
  "GetTotalCount": true,
  "Token": null,
  "Query": {
    "Name": "TermQuery",
    "Query": {
      "FieldName": "uid",
      "Term": 10001
    }
  },
  "Aggregations": [{
    "Name": "avg",
    "Aggregation": {
      "AggName": "agg1",
      "Field": "pid"
    }
  }]
}
```

删除多元索引

删除不需要的多元索引。

- 命令格式

```
drop_search_index -n search_index_name
```

- 示例

  删除search_index多元索引。

```
drop_search_index -n search_index
```

4.5. 通道服务

通道服务（Tunnel Service）是基于表格存储数据接口上的全增量一体化服务。通道服务提供了增量、全量、增量加全量三种类型的分布式数据实时消费通道。通过为数据表建立数据通道，您可以简单地实现对表中历史存量和新增数据的消费处理。

创建通道

为数据表创建一个通道。

- 命令格式

```
create_tunnel -n name
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-t，--table	否	mytable	数据表名称。
-n，--name	是	t1	通道名称。
-m，--mode	否	stream_data_only	通道类型。取值范围如下： ○ base_data_only：全量类型。只能消费处理全量数据。 ○ stream_data_only（默认）：增量类型。只能消费处理增量数据。 ○ base_and_stream：全量加增量类型。全量数据消费处理完成后，再消费处理增量数据。

- 示例

  为数据表创建t1通道。

  ```
  create_tunnel -n t1
  ```

  返回结果如下：

  ```
  New tunnel created, its id is '9933470d-8a5e-4972-a5b0-b7ae6f836460'.
  ```

获取通道信息

获取通道的Tunnel信息以及Channel信息。

- 命令格式

  ```
  describe_tunnel -n name
  ```

  配置项说明请参见下表。

配置项	是否必填	示例值	说明
-t，--table	否	mytable	数据表名称。
-n，--name	是	t1	通道名称。
-o，--output	否	D:\\otstest\\mytunnel.txt	将返回结果保存到本地文件中。

- 示例

  获取t1通道的信息。

  ```
  describe_tunnel -n t1
  ```

返回结果如下：

```
Tunnel Info:
+--------------------------------------+-----------+-----------+--------------+---------+
| TunnelId                             | TunnelName | TunnelType | Stage       | Expired |
+--------------------------------------+-----------+-----------+--------------+---------+
| 9933470d-8a5e-4972-a5b0-b7ae6f836460 | t1        | Stream    | ProcessStream | false   |
+--------------------------------------+-----------+-----------+--------------+---------+
Channel Info:
+-----------------------------------------------------+-------------+--------------+----------+-----------------
-------------+
| ChannelId                                           | ChannelType | ChannelStatus | ClientId | ChannelRPO
|
+-----------------------------------------------------+-------------+--------------+----------+-----------------
-------------+
| cfd2c05b-54b6-48ec-aa6f-feb427f0ca57_1635771329155688 | Stream     | OPEN         |          | 1970-
01-01 08:00:00 +0800 CST |
+-----------------------------------------------------+-------------+--------------+----------+-----------------
-------------+
```

模拟通道消费

创建通道后，通过模拟数据消费可以预览通道中的数据格式。

- 命令格式

```
consume_tunnel -n name -m mock_consume
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-c，--channel	否	cfd2c05b-54b6-48ec-aa6f-feb427f0ca57_1635771329155688	Channel ID。如果不设置此项，则表示消费所有Channel。
-t，--table	否	mytable	数据表名称。
-n，--name	是	t1	通道名称。
-m，--mode	是	mock_consume	消费模式。取值范围如下： ○ shadow_copy（默认）：复制线上Tunnel消费流量。 ○ mock_consume：模拟消费数据，但不会更新断点记录信息。 ○ real_consume：真实消费数据，会同时更新断点记录信息。不推荐使用此模式。

- 示例

在t1通道上模拟数据消费。

```
consume_tunnel -n t1 -m mock_consume
```

执行命令后，当向表中写入数据时，屏幕中会显示数据消费记录，返回示例如下：

```
Starting consume tunnel 't1' of table 'mytable', it may take a few seconds to start, please wait...
{"Type":0,"Timestamp":1636360028961786,"SequenceInfo":{"Epoch":0,"Timestamp":1636360028961786,"RowIndex":1},"PrimaryKey":{"PrimaryKeys":[{"ColumnName":"uid","Value":"86"},{"ColumnName":"pid","Value":6771}]},"Columns":[{"Type":0,"Name":"name","Value":"redchen","Timestamp":1636360028962},{"Type":0,"Name":"country","Value":"china","Timestamp":1636360028962}]}
```

删除通道

删除不需要的通道。

- 命令格式

```
drop_tunnel -n name -y
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-t，--table	否	mytable	数据表名称。
-n，--name	是	t1	通道名称。
-y，--yes	是	不涉及	显示确认信息。命令中必须带有此配置项。

- 示例

删除t1通道。

```
drop_tunnel -n t1 -y
```

4.6. SQL查询

进入SQL模式后，使用SQL语句绑定表的映射关系、获取映射表列表、查看映射表信息以及查询表中数据。

> 说明 关于SQL查询的更多信息，请参见SQL概述。

进入SQL模式

执行 `sql` 命令，进入SQL模式。

绑定映射关系

对于数据表，您需要绑定数据表的映射关系后才能进行数据查询

执行以下命令，为mytable数据表绑定映射关系。

```
CREATE TABLE mytable(
   `uid` VARCHAR(1024),
   `pid` BIGINT(20),
   `b` DOUBLE,
   `c` BOOL,
   `d` MEDIUMTEXT,
   PRIMARY KEY(`uid`,`pid`)
);
```

获取映射表列表

获取实例下所有映射表列表。

执行 `SHOW TABLES;` 命令，获取映射表列表。

返回结果如下：

```
+----------------------+
| Tables_in_myinstance |
+----------------------+
| mytable              |
+----------------------+
| mytstable            |
+----------------------+
| mytstable::meta      |
+----------------------+
```

其中mytable为绑定的数据表，mytstable为时序数据表，mytstable::meta为时序元数据表。

查看映射表信息

查看映射表的描述信息。

- 命令格式

  ```
  DESCRIBE table_name;
  ```

- 示例

  查询mytable表的信息。

  ```
  DESCRIBE mytable;
  ```

  返回结果如下：

```
+-------+---------------+------+-----+-------+
| Field | Type          | Null | Key | Extra |
+-------+---------------+------+-----+-------+
| uid   | varchar(1024) | NO   | PRI |       |
+-------+---------------+------+-----+-------+
| pid   | bigint(20)    | NO   | PRI |       |
+-------+---------------+------+-----+-------+
| b     | double        | YES  |     |       |
+-------+---------------+------+-----+-------+
| c     | tinyint(1)    | YES  |     |       |
+-------+---------------+------+-----+-------+
| d     | mediumtext    | YES  |     |       |
+-------+---------------+------+-----+-------+
```

删除映射表

当表的属性列发生变化时，您可以删除表的映射关系后重新创建。

- 命令格式

```
DROP MAPPING TABLE table_name;
```

- 示例

  删除mytable映射表。

```
DROP MAPPING TABLE mytable;
```

查询表数据

使用SELECT语句查询表中数据。

执行以下命令，查询mytable表中的所有数据。

```
SELECT * FROM mytable;
```

退出SQL模式

执行 `exit;` 命令，退出SQL模式。

5.时序模型

5.1. 时序表操作

创建时序表后，您可以使用表、查询表信息、列出表名称、更新表以及删除表。

> 说明 关于时序模型的更多信息，请参见时序模型概述。

创建表

创建一张时序表，同时指定时序表的数据生命周期（TimeToLive）。

- 命令格式

```
create -m mode -t tableName --ttl timeToLive
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-m，--model	是	timeseries	创建的表类型。取值范围如下： ○ widecolumn（默认）：数据表。 ○ timeseries：时序表。
-t，--table	是	mytable	时序表名称。
--ttl	否	864000	时序表的数据存活时间。默认值为-1，表示数据永不过期。单位为秒。 当系统判断当前时间减去用户传入数据列的时间已经超过设置的数据生命周期时，系统会自动清理超过数据生命周期的数据。 重要 在时序表中，系统判断数据产生时间以用户传入的时间列为准，并非数据写入表中的时间。 取值：大于等于86400秒（一天）或-1（数据永不过期）。

- 示例

  创建名称为mytable的时序表，表中数据永不过期。

```
create -m timeseries -t mytable --ttl -1
```

使用表

选择需要进行操作的表，用于后续表操作或者数据操作。

- 命令格式

```
use --ts -t tableName
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
--ts	是	不涉及	表示操作的表为时序表。
-t，--table	是	mytable	时序表名称。

- 示例

  使用mytable时序表。

  ```
  use --ts -t mytable
  ```

列出表名称

列出实例下的所有表名称、所有数据表名称或者所有时序表名称。

- 列出与当前表类型相同的所有表名称

  ```
  list
  ```

- 列出所有表名称

  ```
  list -a
  ```

- 列出所有数据表名称

  ```
  list -w
  ```

- 列出所有时序表名称

  ```
  list -t
  ```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-a，--all	否	不涉及	列出所有表名称。
-d，--detail	否	不涉及	列出表的详细信息。
-w，--wc	否	不涉及	列出所有数据表名称。
-t，--ts	否	不涉及	列出时序表名称。

更新表

更新时序表的数据生命周期。

- 命令格式

```
alter --ttl timeToLive --ts
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
--ts	是	不涉及	表示操作的表为时序表。
--ttl	是	864000	时序表的数据存活时间。默认值为-1，表示数据永不过期。单位为秒。 当系统判断当前时间减去用户传入数据列的时间已经超过设置的数据生命周期时，系统会自动清理超过数据生命周期的数据。 重要 在时序表中，系统判断数据产生时间以用户传入的时间列为准，并非数据写入表中的时间。 取值：大于等于86400秒（一天）或-1（数据永不过期）。

- 示例

  修改当前表的数据生命周期为86400秒（即1天）。

```
alter --ttl 86400 --ts
```

查看表信息

查看时序表的信息。

- 命令格式

```
desc --ts -t tableName
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
--ts	否	不涉及	表示操作的表为时序表。
-t，--table	否	mytable	时序表名称，可不配置。
-f，--print_format	否	json	表信息的输出格式。取值范围为json（默认）和table。
-o，--output	否	/tmp/describe_table_meta.json	输出表信息到本地JSON格式的文件中。

- 示例

  查看当前表的信息。

```
desc
```

查看mytable时序表的信息。

```
desc --ts -t mytable
```

删除表

删除不需要的表。

- 命令格式

```
drop -t tableName --ts -y
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-t, --table	是	mytable	时序表名称。
-y, --yes	是	不涉及	显示确认信息。命令中必须带有此配置项。
--ts	是	不涉及	表示操作的表为时序表。

- 示例

  删除mytable表。

```
drop -t mytable --ts -y
```

5.2. 数据操作

通过命令行工具您可以在时序表中写入时序数据、导入时序数据、查询时序数据、检索时间线、扫描时间线以及更新时间线。

写入时序数据

写入时序数据到时序表中。

- 命令格式

```
putts --k '["measurement_name","data_source",["tagKey1=tagValue1","tagKey2=tagValue2"]]' --field '[{"c":"fieldname","v":"fieldvalue"},{"c":"bool_field","v":true},{"c":"double_field","v":1.1},{"c":"int_value","v":10,"isint":true}]' --time 1635162859000000
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明

配置项	是否必填	示例值	说明
--k	是	'["cpu","localhost",["region=hangzhou","os=ubuntu"]]'	时间线标识，以数组格式表示。包括以下内容： ○ 度量名称（measurement name）：表示度量值的类型，示例中为cpu。 ○ 数据源标识（data source）：表示产生数据的数据源，示例中为localhost。 ○ 标签（tags）：以数组表示，每个标签由tagKey和tagValue组成，表示为tagKey=tagValue的格式。
--field	是	'[{"c":"fieldname","v":"fieldvalue"},{"c":"bool_field","v":true},{"c":"double_field","v":1.1},{"c":"int_value","v":10,"isint":true}]'	时序数据的数据列，以JSON数据格式表示。每个数据列包含以下内容： ○ c（必选）：数据列名。 ○ v（必选）：数据列值，类型可选字符串、数值和Bool。 ○ isint（可选）：数值类型的数据列值是否按照整型处理，类型为Bool，取值范围为true和false。默认值为false，表示数值类型的数据列值按照浮点数处理。 当设置为true时，表示数据类型的数据列值按照整型处理。
--time	否	1635162859000000	该行时序数据对应的时间，格式为时间戳，单位为微秒（us）。

- 示例

  插入一行时序数据。

```
putts --k '["cpu","localhost",["region=hangzhou","os=ubuntu"]]' --field '[{"c":"fieldname","v":"fieldvalue"},{"c":"bool_field","v":true},{"c":"double_field","v":1.1},{"c":"int_value","v":10,"isint":true}]' --time 1635162859000000
```

导入时序数据

导入本地文件中的时序数据到时序表。

- 命令格式
  - Windows平台

```
import_timeseries --input D:\\localpath\\filename.txt
```

- Linux和Mac平台

```
import_timeseries --input /localpath/filename.txt
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
-i，--input	是	/temp/import_timeseries.txt	通过配置文件导入时序数据。

配置文件的示例如下：

```
cpu,hostname=host_0,region=cn-hangzhou usage_user=58i,usage_system=2i,usage_idle=24i 1609459200000000000
cpu,hostname=host_1,region=cn-hangzhou usage_user=58i,usage_system=2i,usage_idle=24i 1609459200000000000
```

示例文件中的每一行代表一行时序数据，每一行的数据格式为 `measurement_name,tags fields timestamp` ，各字段说明请参见下表。

配置项	是否必填	示例值	说明
measurement	是	cpu	度量值的类型。
tags	是	hostname=host_0,region=cn-hangzhou	每个标签由tagKey和tagValue组成，表示为tagKey=tagValue的格式。 其中第一个标签的tagValue会作为产生数据的数据源，例如示例值中的host_0为产生该行时序数据的数据源。
fields	是	usage_user=58i,usage_system=2i,usage_idle=24i	时序数据的数据列。每个数据列由columnKey和columnValue组成，表示为columnKey=columnValue的格式。 columnValue支持为整型或者浮点数。如果在数值结尾加i，则表示整型，否则表示浮点数。
timestamp	是	1609459200000000000	该行时序数据对应的时间，格式为时间戳，单位为纳秒（ns）。 重要 数据导入到时序表后，配置文件中数据的纳秒单位时间戳会自动转换为微秒单位时间戳。

- 示例

导入import_timeseries.txt文件中的时序数据到时序表中。

```
import_timeseries --input /temp/import_timeseries.txt
```

查询时序数据

查询指定范围内的时序数据。

- 命令格式

```
getts --k '["measurement_name","data_source",["tagKey1=tagValue1","tagKey2=tagValue2"]]' --time_start 0 --time_end 1635162900000000 --limit 100
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
--k	是	'["cpu","localhost",["region=hangzhou","os=ubuntu"]]'	时间线标识，以数组格式表示。包括以下内容： ○ 度量名称（measurement name）：表示度量值的类型，示例中为cpu。 ○ 数据源标识（data source）：表示产生数据的数据源，示例中为localhost。 ○ 标签（tags）：以数组表示，每个标签由tagKey和tagValue组成，表示为tagKey=tagValue的格式。
--time_start	是	0	要查询的时序数据的起始时间。
--time_end	是	1667638230000000	要查询的时序数据的结束时间。
--limit	否	100	最多返回条数，取值范围为1~5000。实际返回条数由服务端决定。

- 示例

查询度量名称为cpu，数据源标识为localhost，且标签为"region=hangzhou"和"os=ubuntu"的时间线中1667638230000000之前产生的所有时序数据。

```
getts --k '["cpu","localhost",["region=hangzhou","os=ubuntu"]]' --time_start 0 --time_end 1667638230000000 --limit 100
```

检索时间线

检索满足指定条件的时间线。

- 命令格式

> **说明** 您可以将命令中的query_ts_meta简写为qtm。

```
query_ts_meta --measurement measurement_name --datasource data_source --limit 10
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明
--measurement	否	cpu	度量名称。
--datasource	否	localhost	数据源标识。
--limit	否	100	最多返回条数。取回范围为1~1000。实际返回条数由服务端决定。
-e，--edit	否	{"measurement":"cpu","data_source":"localhost"}	通过输入JSON格式的内容配置查询条件。

您也可以通过执行 `qtm -e` 命令后，手动输入查询条件来实现复杂查询，查询条件示例如下：

```
{
  "Type":"COMPOSITE",
  "QueryCondition":{
    "Operator":"AND",
    "SubConditions":[
      {
        "Type":"MEASUREMENT",
        "QueryCondition":{
          "Operator":"EQUAL",
          "Value":"CPU"
        }
      },
      {
        "Type":"SOURCE",
        "QueryCondition":{
          "Operator":"EQUAL",
          "Value":"127.0.0.1"
        }
      },
      {
        "Type":"TAG",
        "QueryCondition":{
          "Operator":"GREATER_EQUAL",
          "TagName":"Region",
          "Value":"Jiangning"
        }
      }
    ]
  }
}
```

- 示例

  检索度量名称为cpu且数据源标识为localhost的时间线。

```
query_ts_meta --measurement cpu --datasource localhost --limit 10
```

扫描时间线

获取时序表中所有时间线或者指定个数的时间线。

- 命令格式

> 说明 您可以将命令中的query_ts_meta简写为qtm。

```
query_ts_meta --limit limit
```

配置项说明请见下表。

配置项	是否必填	示例值	说明
--limit	否	10	本次扫描返回的最大行数。如果不配置此项，则表示扫描整表中所有数据。

- 示例

```
query_ts_meta --limit 10
```

更新时间线

更新时间线的属性。

- 命令格式

> 说明 您可以将命令中的update_ts_meta简写为utm。

```
update_ts_meta --k '["measurement_name","data_source",["tag1=value1","tag2=value2"]]' --attrs '["key1=value1","key2=value2"]'
```

配置项说明请参见下表。

配置项	是否必填	示例值	说明

配置项	是否必填	示例值	说明
--k	是	'["cpu","localhost",["region=hangzhou","os=ubuntu"]]'	时间线标识，以数组格式表示。包括以下内容： ○ 度量名称（measurement name）：表示度量值的类型，示例中为cpu。 ○ 数据源标识（data source）：表示产生数据的数据源，示例中为localhost。 ○ 标签（tags）：以数组表示，每个标签由tagKey和tagValue组成，表示为tagKey=tagValue的格式。
--attrs	否	'["city=nanjing","region=jiangning"]'	时间线的属性，以数组表示。每个标签由key和value组成，表示为key=value格式。

- 示例

  更新指定时间线的属性为"city=nanjing"和"region=jiangning"。

```
update_ts_meta --k '["cpu","localhost",["city=hangzhou","region=xihu"]]' --attrs '["city=nanjing","region=jiangning"]'
```

5.3. SQL查询

进入SQL模式后，使用SQL语句获取映射表列表、查看映射表信息以及查询表中数据。

> 说明 关于SQL查询的更多信息，请参见SQL概述。

进入SQL模式

执行 `sql` 命令，进入SQL模式。

获取映射表列表

获取实例下所有映射表列表。

执行 `SHOW TABLES;` 命令，获取映射表列表。

返回结果如下：

```
+----------------------+
| Tables_in_myinstance |
+----------------------+
| mytable              |
+----------------------+
| mytstable            |
+----------------------+
| mytstable::meta      |
+----------------------+
```

其中mytable为绑定的数据表，mytstable为时序数据表，mytstable::meta为时序元数据表。

查看映射表信息

查看映射表的描述信息。

- 命令格式

  ```
  DESCRIBE table_name;
  ```

- 示例

  查询mytable表的信息。

  ```
  DESCRIBE mytable;
  ```

  返回结果如下：

  ```
  +-------+---------------+------+-----+-------+
  | Field | Type          | Null | Key | Extra |
  +-------+---------------+------+-----+-------+
  | uid   | varchar(1024) | NO   | PRI |       |
  +-------+---------------+------+-----+-------+
  | pid   | bigint(20)    | NO   | PRI |       |
  +-------+---------------+------+-----+-------+
  | b     | double        | YES  |     |       |
  +-------+---------------+------+-----+-------+
  | c     | tinyint(1)    | YES  |     |       |
  +-------+---------------+------+-----+-------+
  | d     | mediumtext    | YES  |     |       |
  +-------+---------------+------+-----+-------+
  ```

查询表数据

使用SELECT语句查询表中数据。

执行以下命令，查询mytstable时序数据表中的所有时序数据。

```
SELECT * FROM mytstable limit 10;
```

退出SQL模式

执行 `exit;` 命令，退出SQL模式。

6.查看选项

通过help选项来查看支持的所有选项信息。

- 命令格式

  ```
  help
  ```

  您也可以通过 `命令关键字 help` 的形式，查看单个命令的所有选项信息，例如执行 `alter help` 命令查看alter命令的所有选项信息。

- 返回结果

```
Commands:
  alter                Alter table 更新表信息
  clear                Clear the screen 清空屏幕
  config               Setup the configuration to access Tablestore 配置接入信息
  consume_tunnel       Consume tunnel and print stream data. 消费通道并打印stream数据
  create               Create a new table 创建表
  create_index         Create a new secondary index 创建二级索引
  create_instance      Create a new instance. 创建实例
  create_search_index  Create a new search index 创建多元索引
  create_tunnel        Create tunnel. 创建通道
  del                  Delete the specific row of table. 删除表中数据
  desc                 Describe table. 查看表信息
  describe_instance    Describe the instance, to get the detail info. 查看实例信息
  describe_search_index Describe the search index to get detail info. 查看多元索引信息
  describe_tunnel      Get the tunnel's detail info. 获取通道信息
  drop                 Drop the table 删除表
  drop_index           Drop the secondary index 删除二级索引
  drop_search_index    Drop search index. 删除多元索引
  drop_tunnel          Drop the tunnel. 删除通道
  enable_service       Enable ots service before you start to use Tablestore, it is totally free. 开通表格存储服务，开通过程免费
  exit                 Exit the program 退出程序
  export               Export the whole data of table to file. 导出表中数据到本地文件
  get                  Get row by primary key from table. 读取一行数据
  get_splits           Logically divide the data of the full table into several splits close to the specified size. 将全表数据在逻辑上划分成接近指定大小的若干分片
  getts                Get data from timeseries table 查询时序数据
  help                 Display help 查看帮助信息
  import               Load the data into table, only WideColumn table is supported. 导入数据到数据表中。只有数据表支持此操作
  import_timeseries    Load the data into time series table, only TimeSeries table is supported. 导入数据到时序表中。只有时序表支持此操作
  list                 List all tables 列出表名称
  list_instance        List all the instances. 列出实例列表
  list_search_index    List all the search indexes of the table. 列出多元索引列表
  list_tunnel          List all the tunnels of the table. 列出通道列表
  press_check          Check data for press 检测压测状态
  press_input          Input data for press 插入数据进行压测
  put                  Insert a row into table. 插入新数据到数据表
  putts                put data to a timeseries table 写入时序数据到时序表
  query_ts_meta        Query timeseries meta information. 检索时间线
  quit                 Quit the program 退出程序
  scan                 Scan table. 扫描表数据
  search               Execute search query on search index. 使用多元索引查询数据
  sql                  Run in SQL mode, then you can use SQL to select data. 进入SQL模式，您可以使用SQL语句查询数据
  update               Update the row in table, it will insert a new row if it is not exist. 更新一行数据
  update_ts_meta       Update attributes for timeseries 更新时间线的属性
  use                  Choose table to use. 选择进行操作的表
```