天翼云时序数据库 TSDB
用户指南

中国电信股份有限公司云计算分公司

目 录

1 产品概述 1

1.1 产品概述 1

1.2 产品功能 1

1.3 产品优势 3

1.4 名词解释 3

1.5 限制与约定 5

2 快速入门 6

2.1 概述 6

2.2 创建数据库 6

2.3 连接数据库 8

2.4 数据查询 9

3 操作指南 12

3.1 创建数据库 12

3.2 数据库列表 13

3.3 查询面板 16

3.4 数据洞察 19

3.5 数据时效 22

3.6 导入导出 23

3.7 数据清理 26

3.8 时间线清理 27

3.9 实例监控 28

3.10 安全设置 29
3.11 令牌管理 30
4 应用场景 32
4.1 物联网 IOT 设备监控及分析应用 32
4.2 互联网业务性能监控服务 32
4.3 工业及能源化工设备监测管理 33
5 JAVA SDK 文档 34
5.1 概述 34
5.2 安装 SDK 工具包 34
5.3 SDK 客户端配置 35
5.4 写入数据 37
5.5 查询 42
5.6 关闭客户端 47
5.7 调用其他接口 48
6 TSDB HTTP API 50
6.1 简介 50
6.2 URL 规范 50
6.3 API 接口调用 51
6.4 清理数据 61
6.5 聚合函数 63
7 常见问题 70
7.1 TSDB 有哪些使用约定？ 70
7.2 如何选择 TSDB 的实例规格？ 70
7.3 为什么我的示例数据导入提示成功，但是查询没有数据？ 70
7.4 如何写入数据到数据库中？ 70

7.5 如何进行数据查询？ 70
7.6 连接 TSDB 有什么注意事项？ 71
7.7 我的存储空间快使用满了，该怎么办？ 71
7.8 TSDB 如何进行数据可视化展示？ 71

1 产品概述

1.1 产品概述

天翼云时间序列数据库（Ctyun Time Series Database，简称 CT-TSDB），用于处理按照时间顺序变化的数据。提供对时序数据的高效存储、快速查询能力，以及丰富的数据管理能力。适合于物联网和互联网领域，实现对设备及业务服务的实时监控和告警。

1.2 产品功能

时间序列数据库提供数据接入、数据查询、数据时效管理、数据清理、时间线清理、导入导出，以及实例监控等功能，满足日常的数据库管理、监控和数据维护需求。

- 数据接入工具包：提供经济的应用开发套件（SDK）

SDK 包括创建和关闭 TSDB 客户端，完成初始化客户端或关闭客户端的操作，支持优雅关闭和强制关闭。可以通过写入操作，构建数据点，包括 Metric、Tag 和 Value，并将数据点写入 TSDB，支持同步阻塞的写数据和异步非阻塞写数据方式写入。通过查询操作，获取 Metric、Tag 列表，对当前数据进行过滤、分组和筛选，支持同步查询和异步查询。

- 数据查询：提供聚合查询和即席查询两种查询方式

可以对数据进行标签过滤、值过滤、分组、聚合等查询，并可以支持控制台可视化展示。支持按时间、按度量、按标签过滤，支持树形结构的数据洞察，实现灵活的数据查询。

- 数据管理：支持对数据、时间线的维护和管理

支持对时序数据的管理和维护，您可以对时序数据设置数据时效，过期数据将自动清理，也可手动清理时序数据，或按时间线清理。可导入示例数据用于功能体验，也可对时序数据进行导入导

出。

- 实例管理：支持数据库运行状态的实时监控

监控数据库实例的存储空间、时间序列使用情况，以及写入能力、读写请求数等性能指标。便于您准确的了解数据库运行状况，根据业务需要及时对数据库进行升级维护等管理。

1.2.1 数据读写

- 数据写入

可以通过**写入操作**，构建数据点，包括 Metric、Tag 和 Value，并将数据点写入 TSDB，支持同步阻塞的写数据和异步非阻塞写数据方式写入。

提供 API 方式高并发写入数据，可提供每秒千万级数据点写入能力，并可线性扩展。

- 数据读取

可以通过**读取操作**，获取 Metric、Tag 列表，对当前数据进行过滤、分组和筛选，支持同步查询和异步查询。

支持通过 Restful API 和控制台来查询数据，可以对数据进行标签过滤、值过滤、分组等查询。

1.2.2 数据管理

- 时间线管理

支持对时间线进行清理

- 数据管理

支持时间序列数据的写入、查询和删除

- 数据时效

可以开启数据时效，系统可以自动清除不在有效期内的数据，可设置最大数据有效期为 3 个月

- 数据导入导出

提供数据导入导出接口

1.2.3 数据计算

- 插值查询

提供插值查询能力，将未上传的数据补齐，默认支持线性插值算法。

- 聚合计算

提供 AVG、SUM、MAX 等 18 种聚合函数，可以将数据降精度聚合，并支持嵌套聚合。

- 按值过滤

提供按照标签过滤、按值过滤能力，实现快速数据过滤并返回查询结果。

1.2.4 压缩存储

采用优化后的高效压缩算法，可达 10-20 倍压缩率，大大降低存储成本，并提高数据写入能力。

1.2.5 数据库管理

支持对数据库实例进行订购、续订和退订，以及加载示例数据。

1.2.6 数据库监控

提供实时监控能力，提供对数据库的存储空间使用、时间序列使用、每秒写入能力、读取请求状态进行实时监控。

1.3 产品优势

- 强处理能力

支持数据插值以及多种聚合函数，提供按需写入能力，秒级查询

- 高可用

提供数据三副本，故障自动转移，系统平滑扩容，自动实现数据均衡

- 低成本存储

支持数据压缩，列式存储，高效的编码方式，数据定期合并，数据生命周期管理

- 简单易用

开箱即用，自带资源监控告警、数据管理等常用运维功能，提供 SDK 以及 Restful API

1.4 名词解释

TSDB：Time Series DataBase，时间序列数据库，用于保存时间序列（按时间顺序变化）的海量数据。

时间序列：基于稳定频率持续产生的一系列指标监测数据。例如：监测某城市的水位监测时，每半小时采集一个水位值而产生的一系列数据

度量（metric）：数据指标的类别，例如水位，风力和温度。

值（Value）：度量对应的值。例如 10 米（水位）、15 级（风力）和 20 ℃（温度）。

时间戳（timestamp）：数据产生的时间点。

标签（tag）：一个标签是一个 key-value 对，用于提供额外的信息，例如“城市（TagKey）= 广州（TagValue）”就是一个标签（Tag）。

- 标签键（TagKey，Tagk）：是指标项（Metric）监测指定的对象类型（会有对应的标签值来定位该对象类型下的具体对象）。 例如国家、省份、城市、机房、IP 等；
- 标签值（TagValue，Tagv）：是标签键（TagKey）对应的值。 例如，当标签键（TagKey）是“国家”时，可指定标签值（TagValue）为“中国”。

数据点（data point）："1 个 metric +1 个 timestamp+1 个 value + n 个 tag（n>=1）"唯一定义了一个数据点。当写入的 metric、timestamp、n 个 tag 都相同时，后写入的 value 会覆盖先写入的 value。

时间序列 ：也称作时间线。针对监测对象的某项指标（由度量和标签定义）按特定时间间隔（连续的时间戳）采集的一系列数据点就是一个时间序列。"1 个 metric + n 个 tag（n>=1）"定义了一个时间序列。

- tag 的 key 值和 value 值都相同才算同一个 tag，即 deviceid=1 和 deviceid=2 是两个标签。
- 注意不要将时间戳作为 tag，否则易导致时间序列超过限制。

分组（group）：可以按标签（tag）对数据点进行分组。

聚合函数（aggregator）：可以对一段时间的数据点做聚合，如每 10 分钟的和值、平均 值、最大值、最小值等。

数据库（database）：一个用户可以有多个数据库，一个数据库可以写入多个"度量"的"数据点"。

1.5 限制与约定

1.5.1 使用限制

- 用户默认扩容存储大小不超过 3T。
- 默认数据可设置有效期不超过 90 天。

1.5.2 命名限制

- 数据库实例名称格式要求：长度为 6-40 个字符。只能包含小写字母、数字和下划线，必须以字母开头。
- 数据库实例描述格式要求：最大长度为 80 个字符。允许为任意字符。
- 度量（metric）格式要求：长度为 1-40 个字符。只能包含大小写字母、数字、下划线、点号、横线（单字符），必须以字母开头。
- 标签键（tag-key）格式要求：长度为 1-100 个字符。只能包含大小写字母、数字、下 划线、点号、横线（单字符），必须以字母开头。
- 标签值（tag-value）格式要求：长度为 1-200 个字符。允许任何字符。

2 快速入门

2.1 概述

为了使新用户能够快速使用 TSDB，本章介绍快速入门的流程。

2.1.1 前提条件

您已经注册了天翼云账号并完成实名认证。否则，请先注册天翼云账号。

2.1.2 操作流程

1. 创建数据库：创建 TSDB 数据库。
2. 连接数据库：配置 TSDB 连接获取数据，目前仅支持通过 restful API 写入数据。
3. 数据查询及洞察：查看数据库信息，生成基于时间、度量分组或聚合后的图表，也可自定义查询维度对时序数据做读取和洞察。

2.2 创建数据库

TSDB 区别于传统的关系型数据库，主要用于存储和管理时间序列数据。针对时间序列数据的存储、查询和展现进行了专门的优化，从而获得极高的数据压缩能力、极优的查询性能，适用于物联网和互联网设计及主机监控告警等应用场景。

1. 登录天翼云官网。
2. 在顶部导航栏选择“**云计算>数据库>时序数据库 TSDB**”，进入“产品介绍”页面。
3. 点击**立即开通**，根据页面引导购买实例。
4. 在购买页面，填写配置信息，主要包括：

- 实例规格：每种规格有可支持的写入能力和时间线数量限制，可根据业务选择需购买。
- 数据库名称：设置数据库名称，由小写字母和数字组成，6~40 个字符。

- 写入额度：目前支持 5000，10000，30000 点/秒。
- 存储空间：每种规格有初始存储空间配置，可根据需要购买。
- 购买时长：TSDB 提供包月或包年购买方式，最少每次购买 1 个月。

5. 结果验证

登录 TSDB 控制台，单击左侧菜单栏的**数据库列表**查看已创建的实例。

2.2.1 加载示例数据

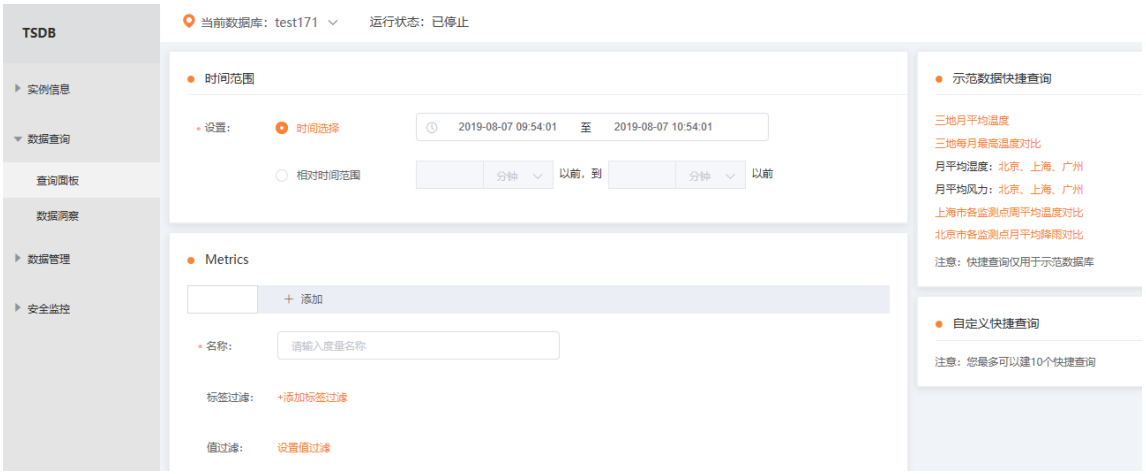
示例数据库用于帮助用户体验时序数据的各项功能。点击某个实例操作栏的“加载示例数据”，系统会自动将预置的示例数据导入该数据库实例。

2.2.2 生成示例数据库图表

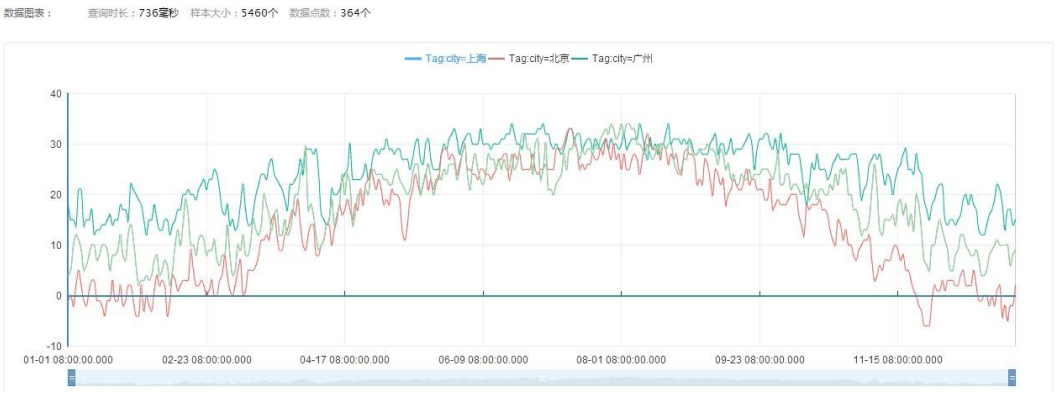
有关 TSDB 生成图表功能的具体介绍请参看生成图表，以下仅通过简单示例展示示例数据库 生成的图表。

注意： 生成的图表中横轴代表写入数据点的 timestamp 字段的值。

查询面板参数设置



生成图表



2.3 连接数据库

2.3.1 通过 API/SDK 写入数据

数据写入前提：

- 用户购买一台 ECS 主机，并可以访问 TSDB 数据库实例
- 用户在安全设置中已设置好白名单，并申请了安全令牌

TSDB 支持通过 restful API，Java SDK 写入数 据。

TSDB 提供的具体 API 接口如下：

- 写入数据点
- 获取度量列表
- 获取标签列表
- 查询数据点

2.3.2 通过控制台导入数据

详情参考：导入导出

2.4 数据查询

注意：

若单次查询时间超过 60s，系统将自动终止本次查询。如果出现查询超时，可以采取以下措施优化查询：

1. 减少单次请求中 query 个数
2. 缩短单个 query 中起止时间的间隔
3. 设置按值或按标签过滤，减少单个 query 涉及到的时间序列数目

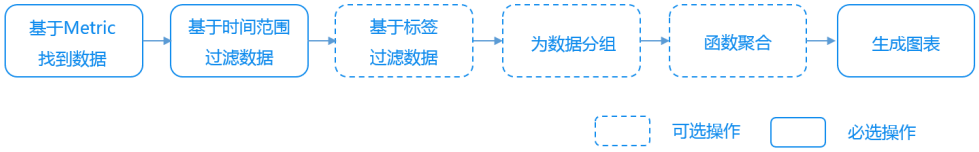
当 TSDB 中存储的数据量较大时，将数据按照用户指定的规则筛选出来需要较长的等待时间，有可能请求导致超时，造成查询失败。

通过查询面板，用户可以对当前数据进行筛选并生成图表，具体操作步骤如下：

1. 选择“**云计算>数据库>时序数据库 TSDB>产品介绍**”，点击**管理控制台**，进入“实例信息”->“基本信息”页面。

2. 在数据库列表中找到对应的数据库，点击“查询面板”，进入操作界面。

生成图表时的，系统对数据处理逻辑如下：

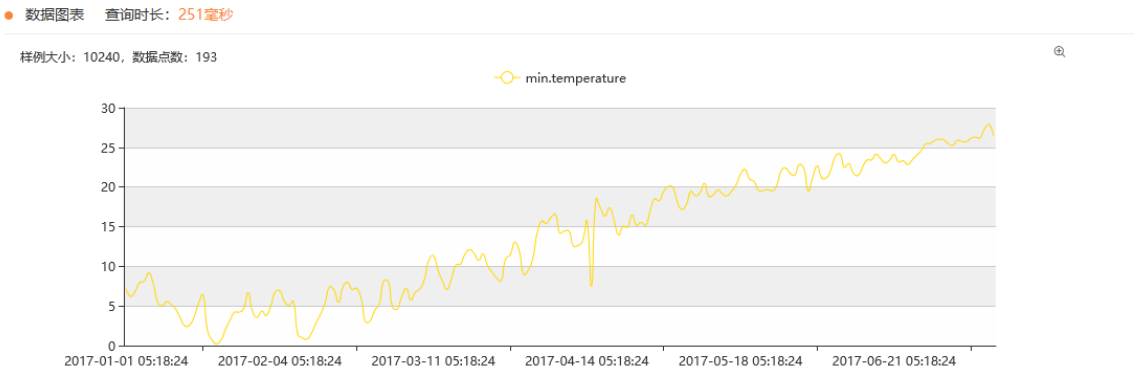


- 时间范围设置：支持“绝对时间”和“相对时间”两种设置方式。
- 名称：从下拉列表中选择当前数据库中已有的 Metrics 名称，用户可点击“添加”，新增多个 Metrics 进行数据对比。
- 标签过滤：指定标签对数据进行过滤。
- 值过滤：根据指定的数值范围对数据点进行过滤，仅显示出指定的数据点。
- 可基于以下二个维度进行数据分组：
  - ✓ 标签：将携带指定标签的数据分为一组，可输入多个标签。
  - ✓ 时间：以给定时间为步长，根据数据的时间戳（Timestamp），将数据分为指定数量的分组。例如：目标时长为 1 小时，分组个数为 12，表示将当前数据分成 12 组，每组数据的时间范围为 1 小时。
- 聚合函数：通过内置函数对数据进行处理，可添加多个函数，系统将按顺序对每个分组应用函数。当前支持的聚合函数包括：
  - ✓ Avg：以每个采样时间范围内的 value 的平均值作为结果
  - ✓ Dev：以每个采样时间范围内的 value 的标准差作为结果
  - ✓ Count：以每个采样时间范围内的点的数目作为结果

- ✓ First：以每个采样时间范围内的第一个 value 作为结果
- ✓ Last：以每个采样时间范围内的最后一个 value 作为结果
- ✓ LeastSquares：对每个采样时间范围内的 value 进行最小二乘法的拟合
- ✓ Max：以每个采样时间范围内的 value 的最大值作为结果
- ✓ Min：以每个采样时间范围内的 value 的最小值作为结果
- ✓ Percentile：每个采样时间范围内的 value 的第 p[百分位数]作为结果。
- ✓ Sum：以每个采样时间范围内的 value 的总和作为结果
- ✓ Diff：以每两个相邻的 value 的差值作为结果
- ✓ Div：以每个 value 除以一个除数作为结果
- ✓ Scale：以每个 value 乘以一个倍数作为结果

- 返回限制：系统将返回指定数量的数据点生成图表，最大值 10000。

3. 点击“生成图表”，系统将根据配置筛选数据并生成图表。



3 操作指南

3.1 创建数据库

时序数据库是用于存储和管理时间序列数据的专业化数据库，提供对时序数据的高效存储、快速查询以及丰富的数据管理等数据库服务。适用于物联网和互联网领域，实现对设备及业务服务的实时监控和告警。

创建数据库步骤如下：

1. 选择“产品服务>时序数据库 TSDB”，进入“实例信息”->。“实例列表”页面。

2. 点击“创建数据库”，进入创建数据库页面，填写配置信息，包括：

- 实例规格：每种规格有可支持的写入能力和时间线数量限制，可根据业务选择需购买。
- 数据库名称：设置数据库名称，长度为 6～40 个字符，只能包含小写字母、数字和下划线，必须以字母开头。
- 描述：设置数据库描述，最大长度为 80 个字符，允许为任意字符，数据库创建后可更改。
- 写入额度：目前支持 5000，10000，30000 数据点/秒。
- 存储空间：每种规格有初始存储空间配置，可根据需要购买。
- 最大时间序列数量：每种规格有规定的最大时间序列数量配置。
- 购买时长：TSDB 提供包月或包年购买方式，最少每次购买 1 个月。

3. 完成配置后点击“开通”，完成 TSDB 数据库的创建。

4. 返回“实例列表”页面，在数据库列表中查看已创建的实例。

3.2 数据库列表

用户在数据库列表可查看列表信息和实例基本信息，可创建新数据库，也可对已购买的实例进行续费、退订、升级等操作。

3.2.1 查看列表信息

选择“产品服务>时序数据库 TSDB”，进入“实例列表”页面。

可以查看数据库的列表信息，包括数据库名称、数据库 ID、运行状态、计费方式、到期时间、描述等信息。

3.2.2 查看基本信息

选择“产品服务>时序数据库 TSDB”，在“实例列表”页面，点击 “管理”进入。

可以查看数据库的基本信息和配置信息，基本信息包括数据库名称、数据库 ID、运行状态、描述、默认连接、VPC 标识；配置信息包括写入能力、最大时间序列、存储空间，可点击“返回列表”回到“实例列表”页面。

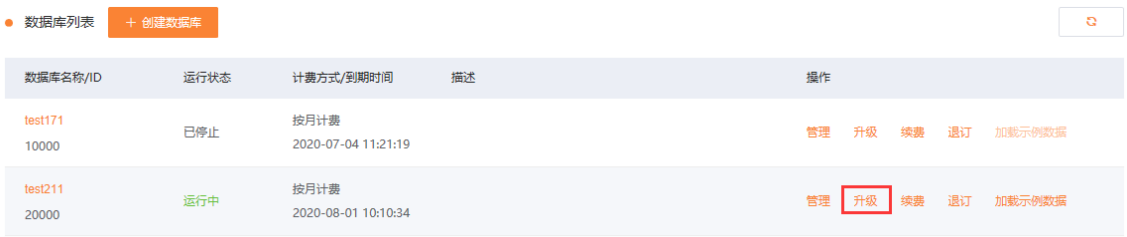
3.2.3 操作数据库

选择“产品服务>时序数据库 TSDB”，进入“实例列表”页面。操作数据库包括管理、续费、退订、加载示例数据。

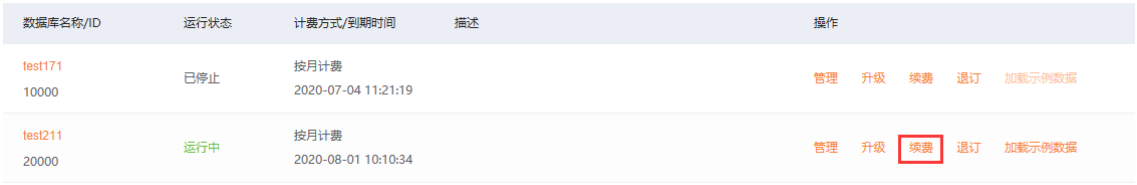
- 管理

点击“管理”，进入“基本信息”页面，为当前数据库编辑描述信息。

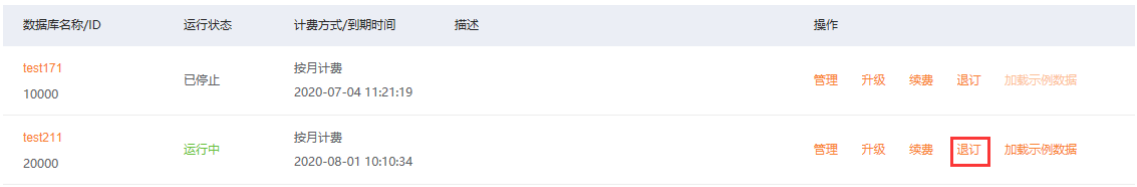
- 点击“升级”，进入升级操作页面，为当前的数据库进行升级，可按月度进行升级。



- 点击“续费”，进入续费操作页面，为当前的数据库进行续费，可按月度进行续费。



- 点击“退订”，进入退订操作页面，退订当前数据库。



- 点击“加载示例数据”并确认，将示例数据加载到当前数据库。

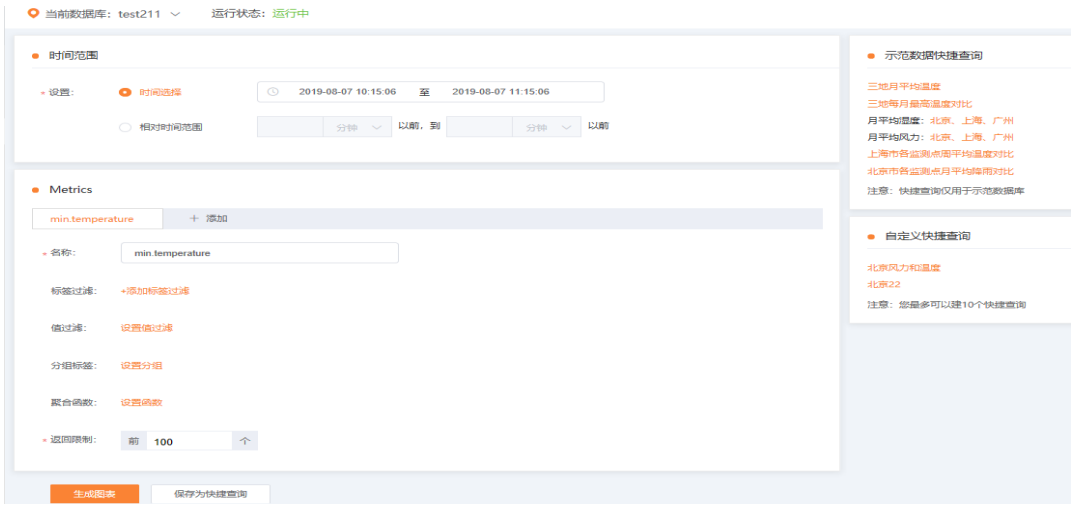
3.3 查询面板

当时序数据库存储的数据量较大时，将数据按照用户指定的规则筛选出来需要较长的时间，有可能请求超时，导致查询失败。

针对这种情况，用户可以对数据进行过滤、分组、聚合等，实现快速返回查询结果。

通过查询面板，用户可以对当前数据进行筛选并生成图表，可将查询条件保存为快捷查询，具体操作步骤如下：

1. 选择“产品服务>时序数据库 TSDB”，进入“查询面板”页面。



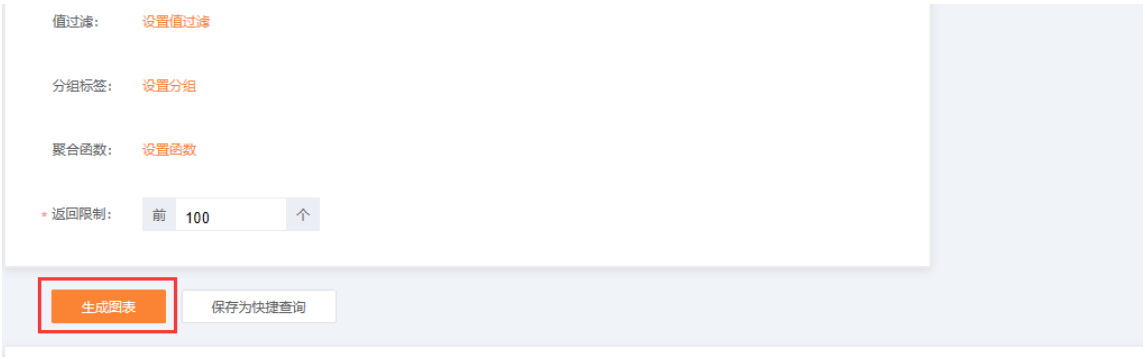
2. 添加时间范围和度量选项 Metrics：

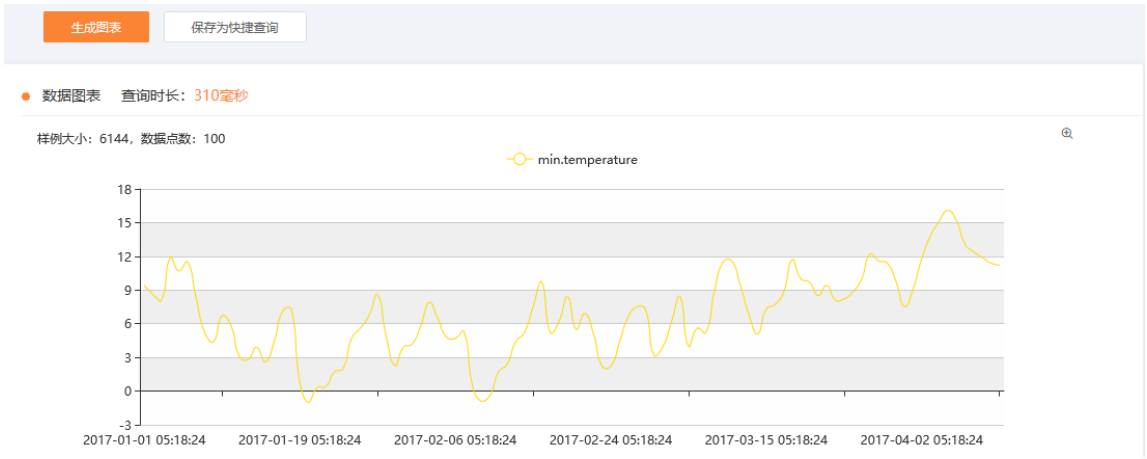
- 时间范围设置：支持“绝对时间”和“相对时间”两种设置方式；
- 名称：从下拉列表中选择当前数据库中已有的 Metrics 名称，用户可点击“添加”，新增多个 Metrics 进 行数据对比；
- 标签过滤：指定标签对数据进行过滤；
- 值过滤：根据指定的数值范围对数据点进行过滤，仅显示出指定的数据点；
- 分组：可基于标签进行数据分组：标签：将携带指定标签的数据分为一组，可输入多个标签，可将数据分成多组，系统将为每组数据生成图表。
- 聚合函数：通过内置函数对数据进行处理，可添加多个函数，系统将按顺序对每个分组应用函数。当前 支持的聚合函数包括：
  - Avg：以每个采样时间范围内的 value 的平均值作为结果
  - Dev：以每个采样时间范围内的 value 的标准差作为结果
  - Count：以每个采样时间范围内的点的数目作为结果
  - First：以每个采样时间范围内的第一个 value 作为结果

- Max：以每个采样时间范围内的 value 的最大值作为结果
- Median：以每个采样时间范围内 value 的中位数作为结果
- Min：以每个采样时间范围内的 value 的最小值作为结果
- Mimmax：以每个采样时间范围内，如果最小值丢失，则取最大数据点作为结果，不执行插值
- Mimmin：以每个采样时间范围内，如果最大值丢失，则取最小数据点作为结果，不执行插值
- Mult：以每个采样时间范围内的 value 值相乘，若 value 值不存在则线性插值
- Pfsum：以每个采样时间范围内的 value 值求和，若值不存在则向前插值
- Sum：以每个采样时间范围内的 value 的总和作为结果
- Zimsum：以每个采样时间范围内的指定时间戳计算所有数据点的总和作为结果，不执行插值

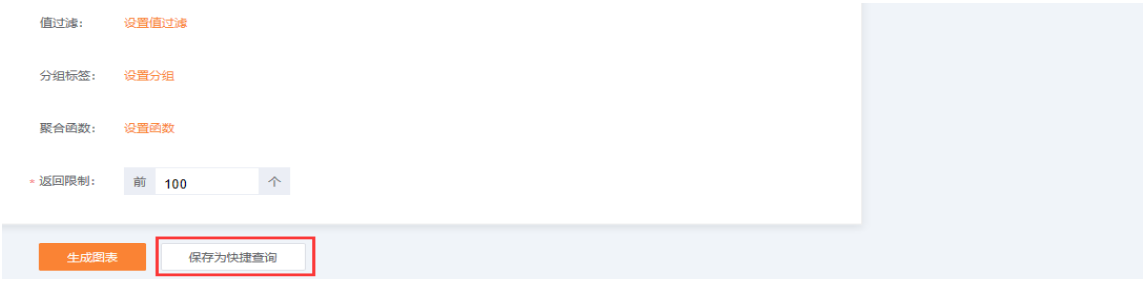
- 返回限制：系统将返回指定数量的数据点生成图表，最大值 10000。

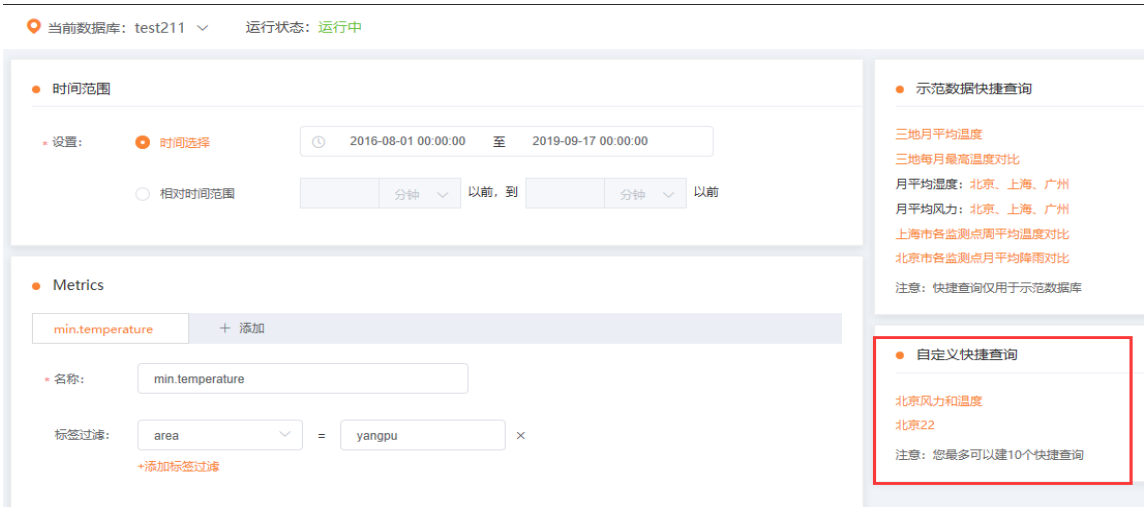
3. 点击“生成图表”按钮生成数据图表；





4. 或者点击“保存为快捷查询”按钮，方便下次查询。



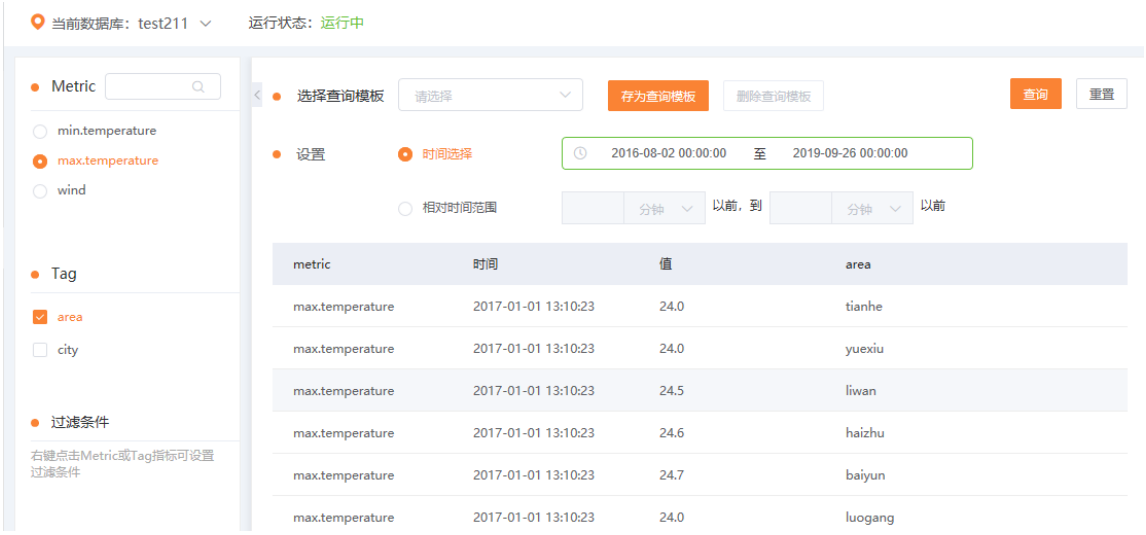


3.4 数据洞察

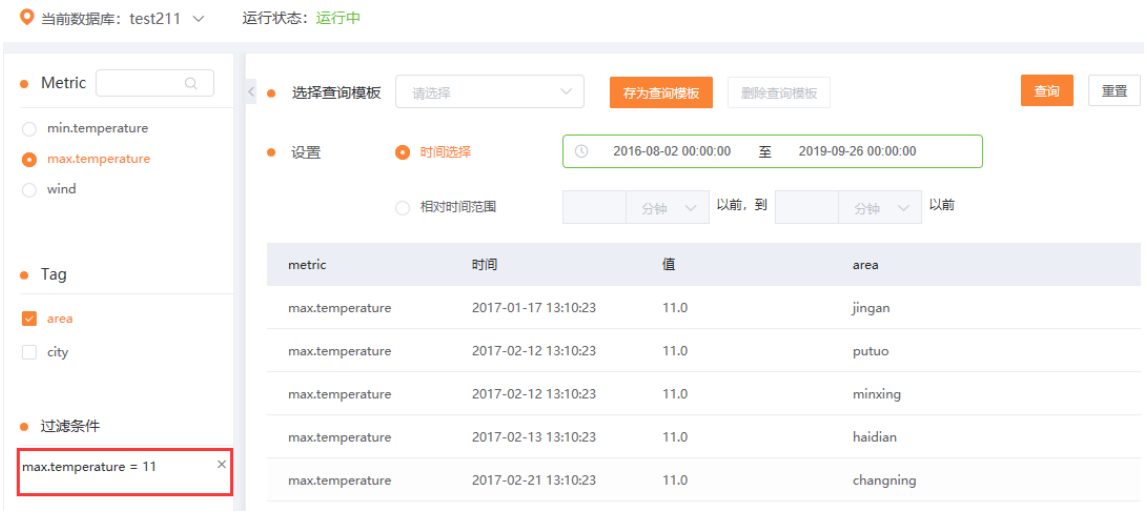
数据洞察为用户提供了灵活的查询方式，用户可根据自己的需求配置查询条件，系统自动查询出符合条件的数据，具体操作步骤如下：

1. 选择“产品服务>时序数据库 TSDB”，进入“数据洞察”页面。

2. 设置 Metric、Tag、时间选择以及相对时间范围。

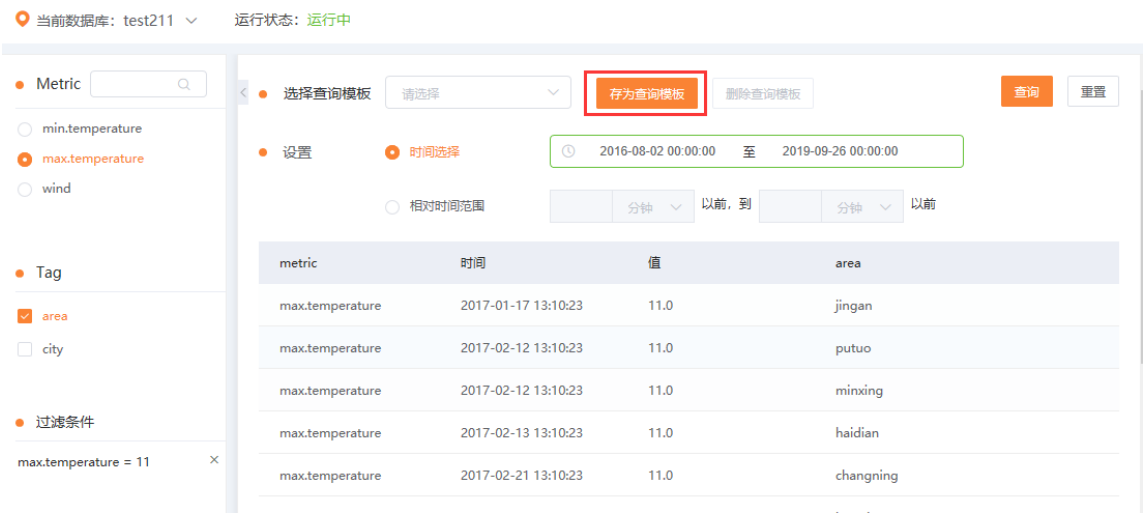


3. 可选中 Metric 或 Tag，右键进行设置过滤条件。



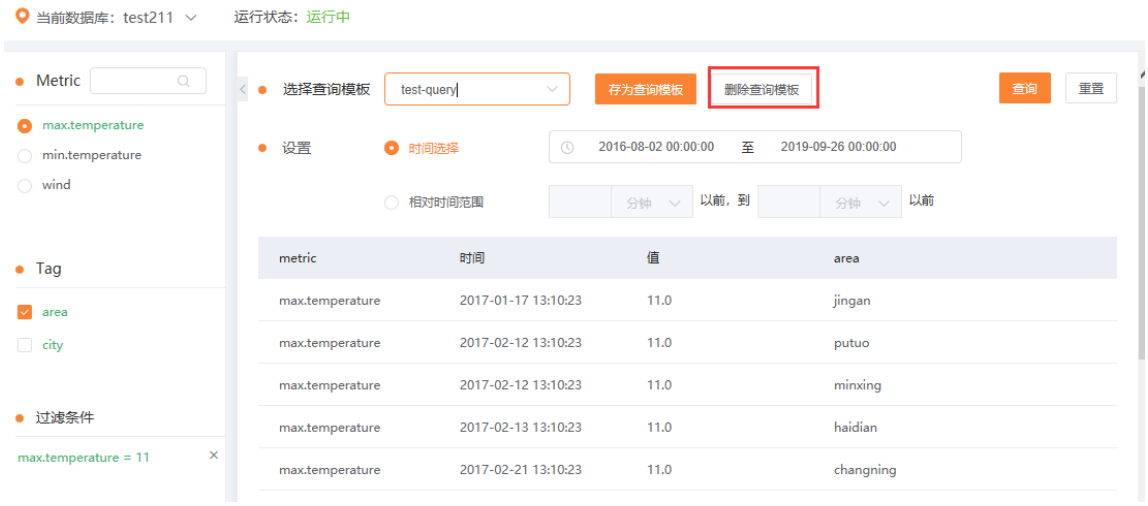
4. 点击“查询”按钮，查询当前数据库图表信息。

5. 可以选择“存为查询模板”，方便下次查询。



-
-

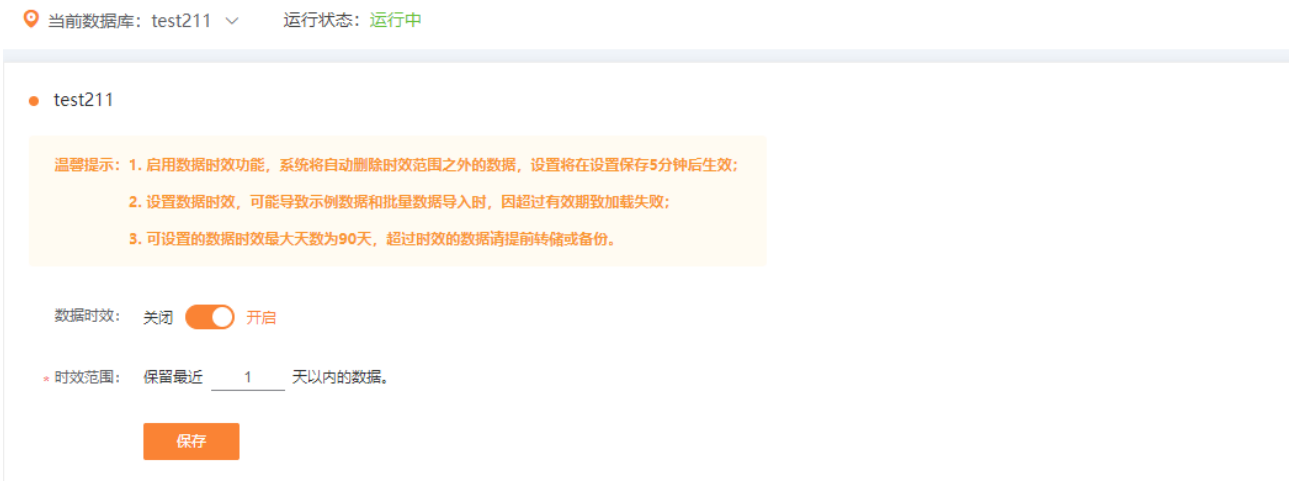
6. 点击“删除查询模板”，删掉当前选中的模板。



3.5 数据时效

随着业务规模增长和时间的不断累积，为了避免不重要的数据占用过多存储空间，我们提供自动清理和手动清理两种方式。自动清理就是设置数据的时效性，只保留一定时期内的数据，超时的数据系统将自动清理掉。

从左边菜单导航选择“数据时效”进入。 数据时效默认设置为“关闭”，需要定时清理数据时，将数据时效设置为“开启”，并设置时效范围，以天为单位，保存后，系统会保留时效内的数据。



注意：

1. 启用数据时效功能，系统将自动删除时效范围之外的数据，更改将在设置保存 5 分钟后生效；

2. 设置数据时效，可能导致示例数据和批量数据导入时，因超过有效期致加载失败；

3. 可设置的数据时效最大天数为 90 天，超过时效的数据请提前转储或备份。

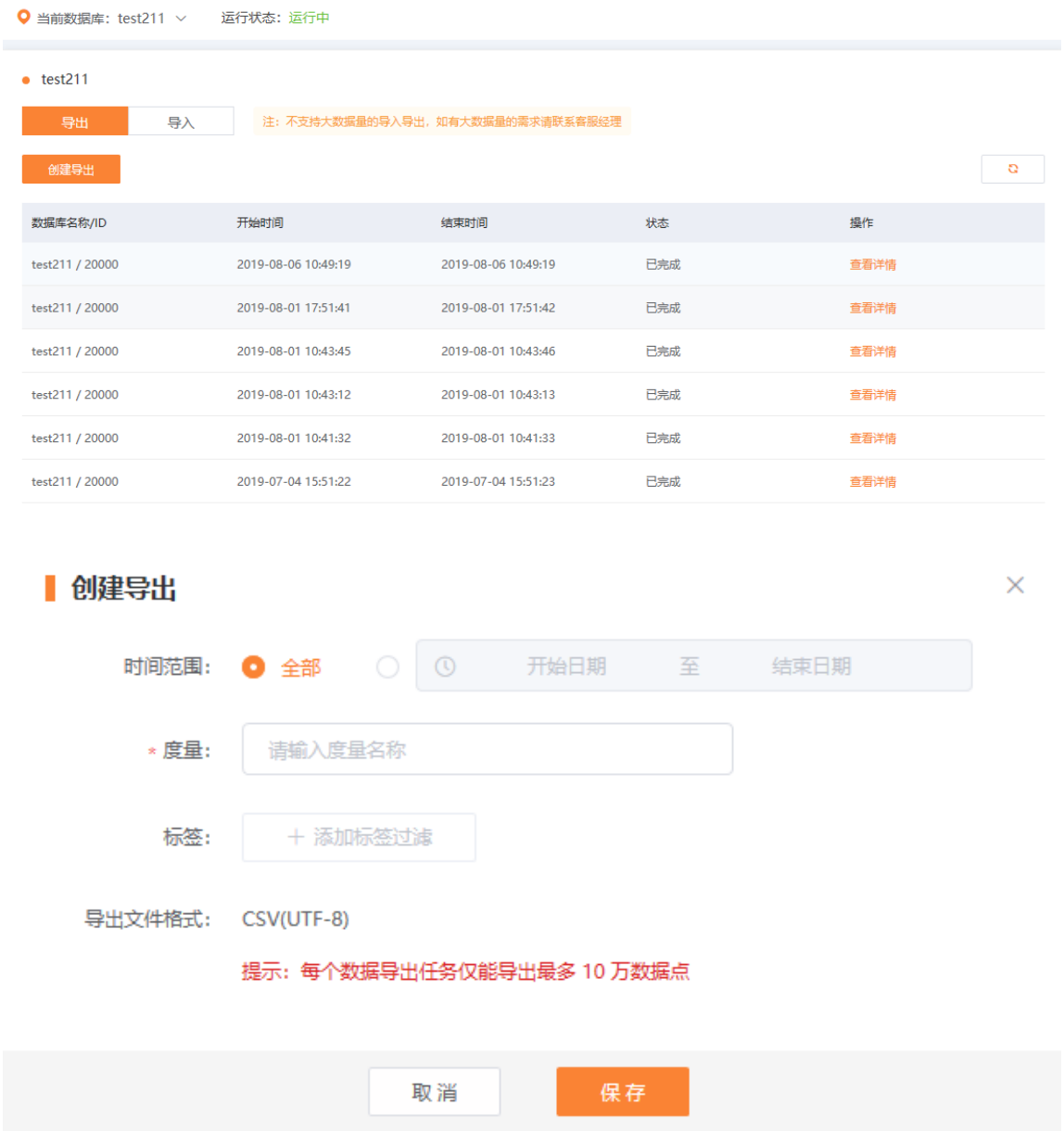
3.6 导入导出

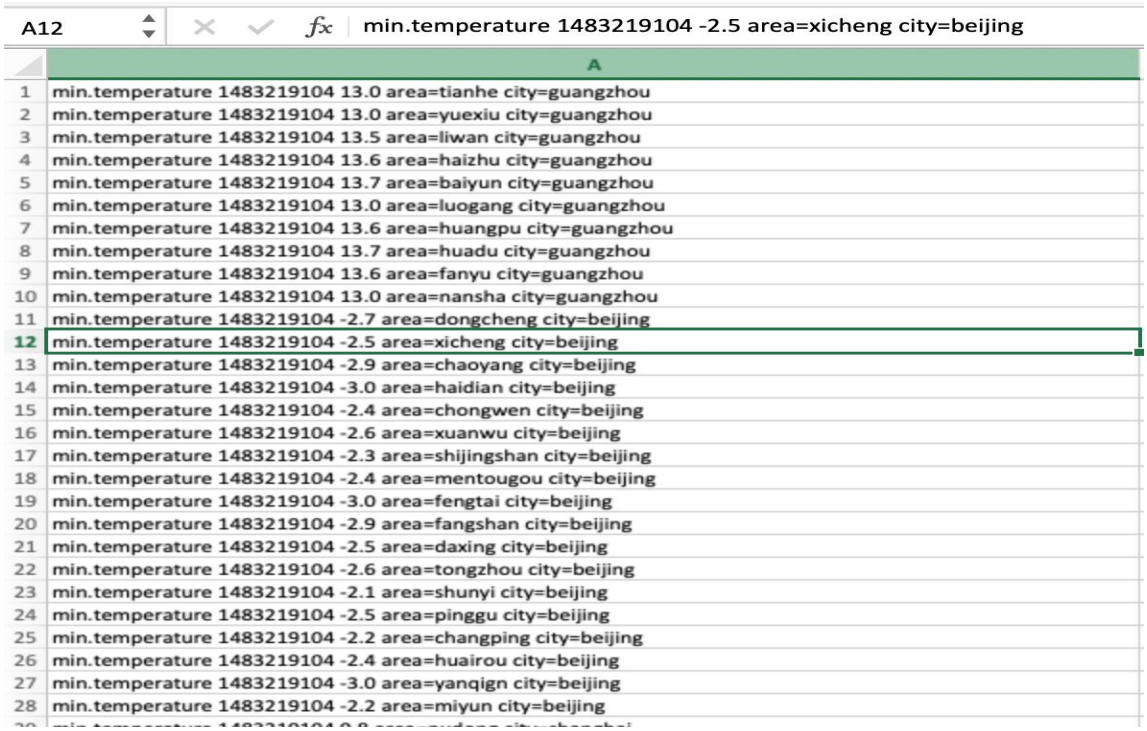
当用户需要导入比较小数据量（csv 文件小于 50M）时，可以从页面进行导入。从左边菜单导航选择“导入导出”进入后，切换到导入 Tab，点击“创建导入”按钮，将会出现导入的弹窗。可以在弹窗中设置导入的 CSV 文件编码，点击“选取文件”按钮将出现文件选择框，选择需要导入的 CSV 文件，最后点击“保存”，系统将会生成相应的任务去执行导入。（注意：导入 CSV 内容格式是以空格为分隔符的键值对，详见导出的样例）

在导入 Tab 页面下方是每次导入任务的详情，可以点击链接“查看详情”，在任务详情弹窗中可以查看当时导入任务的情况、以及导入成功的点数、若有错误信息也会一并展示。

当用户需要导出比较小数据量（10 万以下）时，可以从页面进行导出 。从左边菜单导航选择“导入导出”进入后，切换到导出 Tab，点击“创建导出”按钮，将会出现导出的弹窗。可以在弹窗中设置导出的时间范围、度量，最后点击“保存”，系统将同步生成 CSV 文件后下载到本地。（注意：导出 CSV 内容格式是以空格为分隔符的键值对 ）。

在导出 Tab 页面下方是每次导出任务的详情，可以点击链接“查看详情”，在任务详情弹窗中可以查看当时导出的条件、以及导出的点数等信息。


当前数据库：test211
运行状态：运行中
test211
导出
导入
注：不支持大数据量的导入导出，如有大数据量的需求请联系客服经理
创建导出
数据库名称/ID
开始时间
结束时间
状态
操作
test211 / 20000
2019-08-06 10:49:19
2019-08-06 10:49:19
已完成
查看详情
test211 / 20000
2019-08-01 17:51:41
2019-08-01 17:51:42
已完成
查看详情
test211 / 20000
2019-08-01 10:43:45
2019-08-01 10:43:46
已完成
查看详情
test211 / 20000
2019-08-01 10:43:12
2019-08-01 10:43:13
已完成
查看详情
test211 / 20000
2019-08-01 10:41:32
2019-08-01 10:41:33
已完成
查看详情
test211 / 20000
2019-07-04 15:51:22
2019-07-04 15:51:23
已完成
查看详情
创建导出
时间范围：
全部
开始日期
至
结束日期
* 度量：
请输入度量名称
标签：
+ 添加标签过滤
导出文件格式：
CSV(UTF-8)
提示：每个数据导出任务仅能导出最多 10 万数据点
取消
保存



A12 | min.temperature 1483219104 -2.5 area=xicheng city=beijing

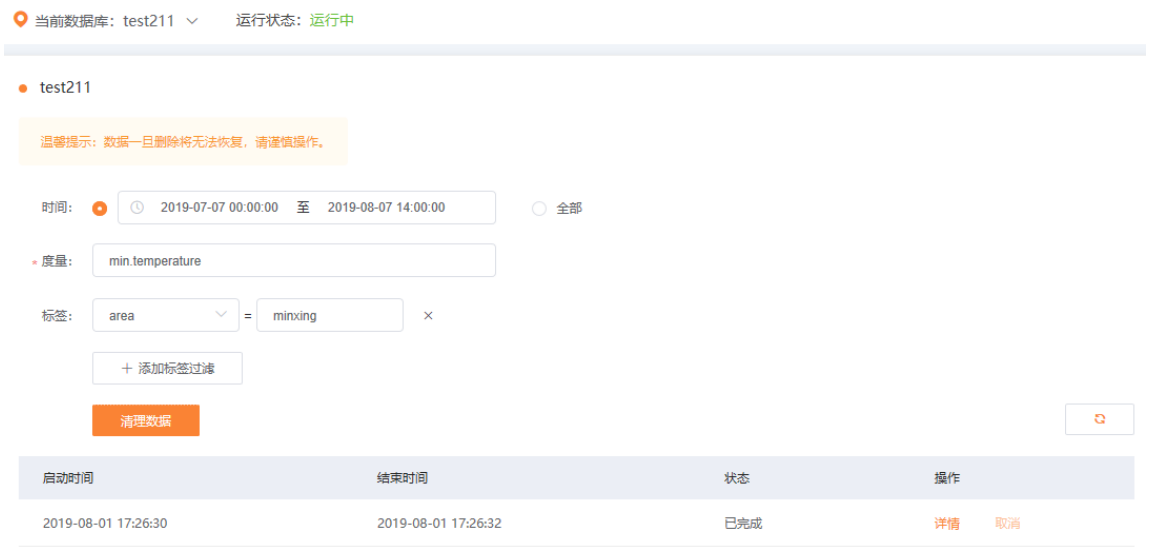
	A
1	min.temperature 1483219104 13.0 area=tianhe city=guangzhou
2	min.temperature 1483219104 13.0 area=yuexiu city=guangzhou
3	min.temperature 1483219104 13.5 area=liwan city=guangzhou
4	min.temperature 1483219104 13.6 area=haizhu city=guangzhou
5	min.temperature 1483219104 13.7 area=baiyun city=guangzhou
6	min.temperature 1483219104 13.0 area=luogang city=guangzhou
7	min.temperature 1483219104 13.6 area=huangpu city=guangzhou
8	min.temperature 1483219104 13.7 area=huadu city=guangzhou
9	min.temperature 1483219104 13.6 area=fanyu city=guangzhou
10	min.temperature 1483219104 13.0 area=nansha city=guangzhou
11	min.temperature 1483219104 -2.7 area=dongcheng city=beijing
12	min.temperature 1483219104 -2.5 area=xicheng city=beijing
13	min.temperature 1483219104 -2.9 area=chaoyang city=beijing
14	min.temperature 1483219104 -3.0 area=haidian city=beijing
15	min.temperature 1483219104 -2.4 area=chongwen city=beijing
16	min.temperature 1483219104 -2.6 area=xuanwu city=beijing
17	min.temperature 1483219104 -2.3 area=shijingshan city=beijing
18	min.temperature 1483219104 -2.4 area=mentougou city=beijing
19	min.temperature 1483219104 -3.0 area=fengtai city=beijing
20	min.temperature 1483219104 -2.9 area=fangshan city=beijing
21	min.temperature 1483219104 -2.5 area=daxing city=beijing
22	min.temperature 1483219104 -2.6 area=tongzhou city=beijing
23	min.temperature 1483219104 -2.1 area=shunyi city=beijing
24	min.temperature 1483219104 -2.5 area=pinggu city=beijing
25	min.temperature 1483219104 -2.2 area=changping city=beijing
26	min.temperature 1483219104 -2.4 area=huairou city=beijing
27	min.temperature 1483219104 -3.0 area=yanqign city=beijing
28	min.temperature 1483219104 -2.2 area=miyun city=beijing

注： 导出 CSV 文件样例

3.7 数据清理

除了设置数据时效的方式，用户还可以手动清除某些符合条件的数据。从左边菜单导航选择“数据清理”进入。 选择需要删除的时间段（或者全部），设置需要清理的度量、以及标签，点击“清理数据”按钮，系统将会生成相应的任务去执行清理。清理结果可以在下面任务列表进行查询。未

开始的任务可以进行取消，点击详情链接可以查看任务的详情以及删除的点数。



3.8 时间线清理

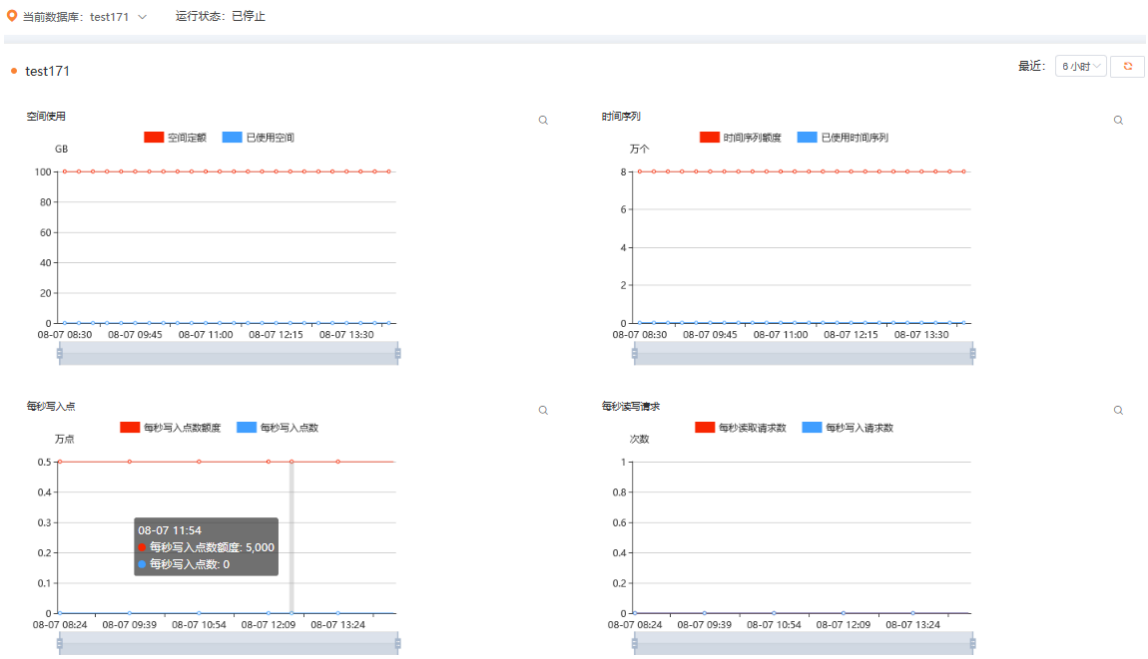
用户可以对不再使用的时间线进行清理，从左边菜单导航选择“时间线清理”进入。创建清理任务，在页面中先选择要清理的“度量”，然后选择该度量下要清理的标签，标签支持全选、搜索，标签块中最多选择 8 个，点击标签块旁边“选择”按钮，可在弹出框中选择符合条件的标签值，逐个

选择标签及标签值，完成清理条件选择，最后点击按钮“清理”，系统将会生成清理任务，可点击“重置”按钮重置时间线清理条件。

在页面的下面是清理时间线任务的历史列表，可以查看之前创建的所有清理时间线任务，点击链接“详情”将会出现任务详情的弹窗，可以查看清理时间线任务的条件、以及操作结果等信息。

3.9 实例监控

用户可以实时监控当前数据库的信息，了解数据库的使用情况。从左边菜单导航选择“实例监控”进入，可以看到实例的“空间使用”、“时间序列”、“每秒写入点”、“每秒读写请求”四个指标的曲线图。系统默认给出最近 6 小时内的监控信息，用户还可以点击右上角下拉菜单，切换时间范围 。

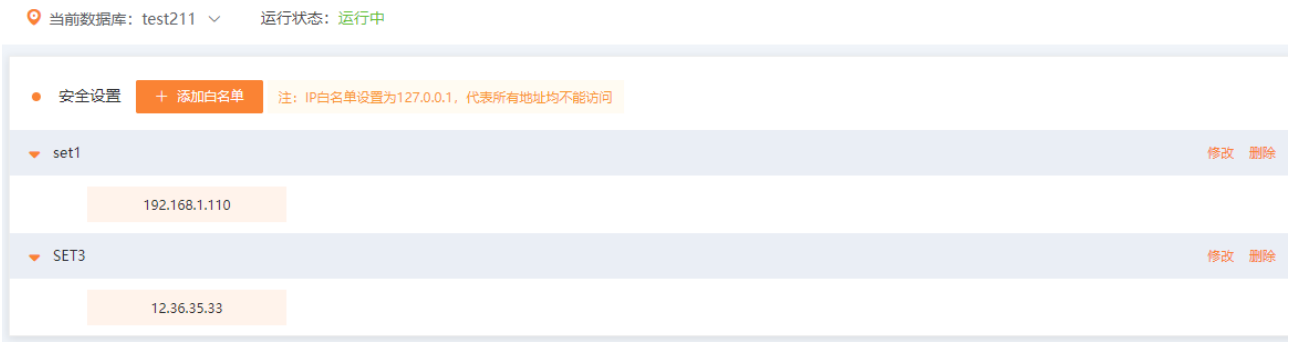


如果需要查看更详细的内容，可以点击每个图形右上角的放大按钮，放大当前曲线图，在大图中可以设置“统计项”、“采样周期”、“最近时间”等选项进行查看。

3.10 安全设置

用户可以设置访问当前数据库的白名单。从左边菜单导航选择“安全设置”进入，可以看到当前实例的已设置的白名单。可新增白名单分组及白名单，也可修改已设置的白名单。只有白名单分内的 IP 才能访问实例。

注：IP 白名单设置为 127.0.0.1，代表所有地址均不能访问



3.11 令牌管理

为加强数据访问安全管理，加强多业务场景下对数据的访问和使用管理，用户对实例的访问除了需要设置白名单外，还需要申请相应的访问令牌才能实现数据访问。每个实例可申请的有效令牌

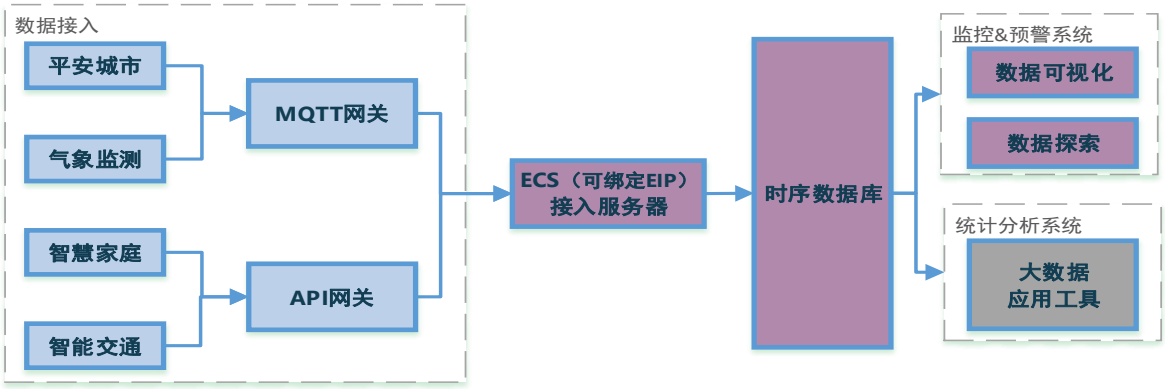
数不超过 5 个，可在令牌到期前进行修改续期，也可停用令牌。

4 应用场景

4.1 物联网 IOT 设备监控及分析应用

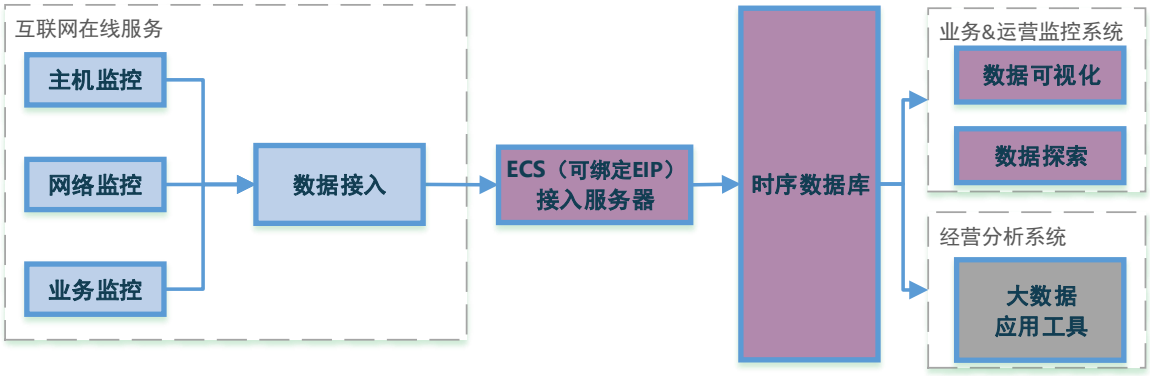
在物联网应用中，大量的物联网设备，要求满足海量数据高速写入、存储和低时延查询，TSDB 服务基于 HBase 的存储架构，轻松满足海量数据存储和查询场景。各种物联网设备状态数据实时高效写入到时序 数据库中。您可以：

- 通过时序数据库的数据 API 接口读取实时数据；
- 利用时序数据库的查询优势，对数据进行各种聚合运算得到数据统计报表；
- 通过时序数据库的控制台直观得到数据的变化趋势，帮助用户分析数据。



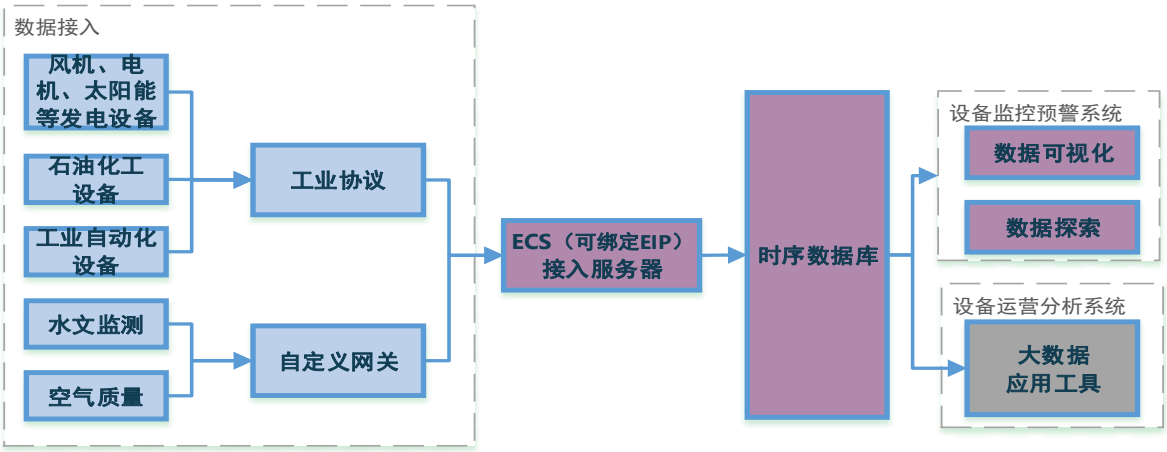
4.2 互联网业务性能监控服务

互联网企业需要对主机、网络及其他硬件设备的监控，对交易情况、处理性能、异常等进行实时监控告警，TSDB 可以提供高性价比的存储方案，实时的任意维度时序聚合计算，是运维监控数据存储的理想选择。



4.3 工业及能源化工设备监测管理

各类工业设备不断产生大量的时间序列数据，需要实时高效写入及聚合运算，从而及时准确的监测到异常状况， TSDB 正是为此而生 ，提供高效存取、高速访问、低成本存储、实时告警信息，实现设备实时监控分析系统。



5 Java SDK 文档

5.1 概述

本章主要介绍 TSDB 的安装和使用信息，帮助用户快速入门并且能够快速找到自己需要的功能方法。

5.2 安装 SDK 工具包

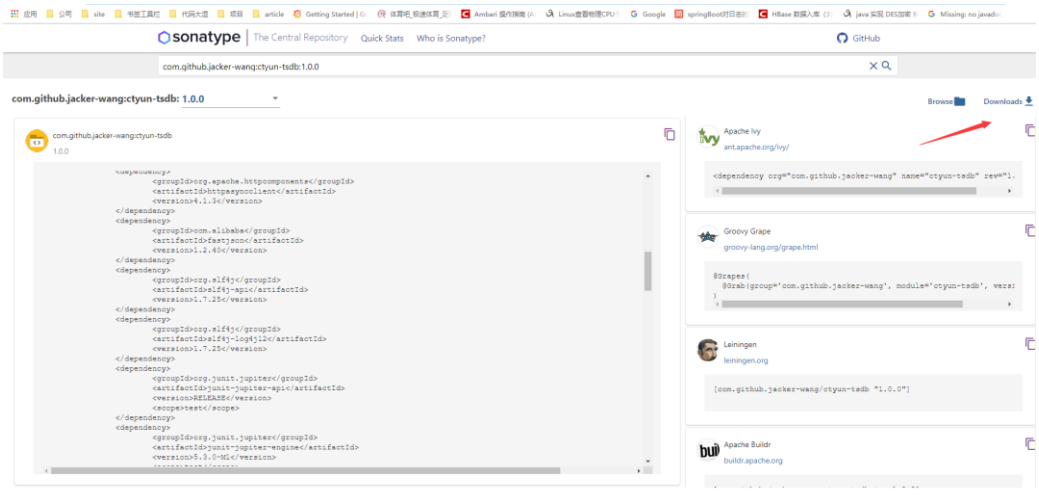
运行环境：

Java SDK 工具包可在 jdk1.7、jdk1.8 环境下运行。

使用方式：直接使用 JAR 包安装

步骤如下：

1 下载最新版 Java SDK 压缩工具包，在 sonatype 的仓库中可以下载，进入 jar 包所在网址 https://search.maven.org/artifact/com.github.jacker-wang/ctyun-tsdb/1.0.0/jar，具体操作见下图。



- 2 在 Eclipse 右键“工程 -> Properties -> Java Build Path -> Add JARs”。
- 3 添加第三方依赖工具包 ctyun-tsdb-{version}.jar 到工程中。

5.3 SDK 客户端配置

CtyunTsdbRequest 是操作 TSDB 的客户端操作类，所以使用 TSDB 的 SDK 前首先需要创建 CtyunTsdbRequest 的实例对象。而 CtyunTsdbRequest 的对象会用到 TSDB 的相关配置，所有配置相关都是在类 CtyunTsdbRequestConfig 中。

创建 CtyunTsdbRequest 类对象需要使用以下两个类：

CtyunTsdbRequestConfig 类：CtyunTsdbRequestConfig 客户端的基础配置类。

CtyunTsdbRequestFactory 类：CtyunTsdbRequest 类的工厂类。

通过 CtyunTsdbRequestConfig 类进行配置，然后放入 CtyunTsdbRequestFactory 类的 connect 方法中即可产生一个 CtyunTsdbRequest 对象。

注意：在创建 CtyunTsdbRequest 对象前，需要先创建 CtyunTsdbRequestConfig 对象。

示例代码

```
CtyunTsdbRequestConfig config;

CtyunTsdbRequest client;

// www.example.com 表示域名或地址。8585 表示 CtyunTsdbRequest 的网络端口。您实际的域名地址和网络端口可到控制台获取。

config = CtyunTsdbRequestConfig.builder("www.example.com", 8585).build();

 // 通过 CtyunTsdbRequestFactory 生成一个 CtyunTsdbRequest 对象。

client = CtyunTsdbRequestFactory.connect(config);
```

客户端的所有配置均由 CtyunTsdbRequestConfig 类进行配置。您可以通过 build() 方法构建 CtyunTsdbRequestConfig 对象。具体配置说明见下面的示例代码。

```
CtyunTsdbRequestConfig config = CtyunTsdbRequestConfig.

//配置地址，第一个参数可以是 TSDB 的域名或 IP。第二个参数表示 TSDB 端口。

 1. address("www.example.com ", 8585)
```

```
2.
3. // 网络连接池大小，默认为 64。
4. .httpConnectionPool(64)
5.
6. // HTTP 等待时间。
7. .httpConnectTimeout(9000)
8.
9. // IO 线程数，默认为 1。
10. .ioThreadCount(1)
11.
12. // 异步写开关。默认为 true。推荐异步写。
13. .asyncPut(true)
14.
15. // 异步写相关，客户端缓冲队列长度，默认为 10000。
16. .batchPutBufferSize(20000)
17.
18. // 异步写相关，缓冲队列消费线程数，默认为 5。
19. .batchPutConsumerThreadCount(2)
20.
21. // 异步写相关，每次批次提交给客户端点的个数，默认为 50。
22. .batchPutSize(50)
23.
24. // 异步写相关，每次等待最大时间限制，单位为 ms，默认为 300。
25. .batchPutTimeLimit(300)
26.
```

```
27. // 异步写相关，写请求队列数，默认等于连接池数。可根据读写次数的比例进行配置。
28. .putRequestLimit(100)
29.
30. // 异步写相关，不限制写请求队列数，若关闭可能导致 OOM，不建议关闭。
31. .closePutRequestLimit()
32.
33. // 异步写相关，异步批量 Put 回调接口。
34. .listenBatchPut(new BatchPutCallback() {
35. @Override
36.     public void response(String address, List<DataPoint> points, Result
    result) {
37.  }
38. })
39. // 流量限制，设置每秒最大提交 Point 的个数,默认为 1000。
40. .maxTPS(1000)
41. .build(); // 构造 CtyunTsdbRequestConfig 对象
```

5.4 写入数据

5.4.1 构造时间点

一个 DataPoint 表示一个时间线上某一个时刻的时间点插入数据之前构造数据点的实例如下：

实例一：

构建一个时间点。用单位为毫秒的时间戳表示时间，指定 Point 数据的 Metric 与多个 Tag。

```
DataPoint point1
=DataPoint.metric("Z").timestamp(System.currentTimeMillis()).value("123")

.tag("diqu", "beijing")
```

```
    .build();
```

示例二：

构建一个时间点。用单位为毫秒的时间戳表示时间，指定 Point 数据的 Metric，Tag 使用 Map 形式的键值对。

```
1. // 也可以'毫秒'为时间戳
2. long timestamp = System.currentTimeMillis();
3. // 使用 HashMap 表示 Tags
4. Map<String,String> tagsMap = new HashMap<String,String>();
5. tagsMap.put("tagk1", "tagv1");
6. tagsMap.put("tagk2", "tagv2");
7. // 构造 Point
8. DataPoint dataPoint = DataPoint.metric("test1").
9. tag(tagsMap)
10. .value(timestamp,123.456)
11. .build();
```

示例三

构建一个时间点，使用 `java.util.Date` 表示时间。

```
1. // 使用 java.util.Date 表示时间。
2. DataPoint point = DataPoint.metric("test1")
3. .tag("tagk1", "tagv1")
4. .value(new Date,123.456)
5. .build();
```

5.4.2 数据插入

TSDB 写数据有同步阻塞写数据和异步非阻塞写数据。

5.4.2.1 同步阻塞写数据

假设我们现在需要构建 500 个时间点提交给 TSDB。

示例代码

```
1.  List<DataPoint> dataPoints= new ArrayList< DataPoint >();
2.  构建 DataPoint
3.  for(int i = 0; i<500; i++) {
4.  long timestamp = System.currentTimeMillis();
5.  DataPoint dataPoint = DataPoint.metric("test1")
6.  .tag("tagk1", "tagv1")
7.  .value(timestamp, Math.random())
8.  .build();
9.  // 手动打包数据
10. dataPoints.add(dataPoint);
11. }
12.
13. // 手动打包后提交数据
14. tsdb.put(dataPoints)
```

- -

```
PutResult result = tsdb.put(ps);
```

返回提交概述。包含成功数和返回数。本质是调用 POST

/api/put?summary=true

```
PutSummaryResult summaryResult = tsdb.put(ps,
PutSummaryResult.class);
```

返回提交概述。包含成功数、返回数和失败原因。本质是调用 POST /api/put?details=true

```
PutDetailsResult detailsResult = tsdb.put(ps,
PutDetailsResult.class);
```

5.4.2.2 异步非阻塞写数据

异步写数据的方式比较简单，只要有 DataPoint 就可以提交数据。Client 会自动帮助您批量地提交数据，不需要您手动打包。如果没有特殊需求，推荐您使用异步非阻塞的写数据的方式。

示例代码

```
1. DataPoint dataPoint = DataPoint.metric("test1")
2. .tag("tagk1", "tagv1")
3. .value(timestamp, Math.random())
4. .build();
5. // 直接提交数据
6. tsdb. putAsynch (dataPoint);
```

5.4.2.3 异步非阻塞写数据回调设置

当您使用异步写数据的方式时，若需要获取 Put 接口的响应结果，则需要设置回调接口。

示例代码

```
1. final AtomicLong num = new AtomicLong();
2. // 回调对象
```

```
3. PostDataCallback  cb = new PostDataCallback () {
4. @Override
5. public void response(String address, List< DataPoint> input, Result output) {
6. long afterNum = num.addAndGet(input.size());
7. System.out.println("成功处理" + input.size() + ",已处理" + afterNum);
8. }
9. @Override
10. public void failed(String address, List< DataPoint> input, Exception ex) {
11.   ex.printStackTrace();
12.   long afterNum = num.addAndGet(input.size());
13.   System.out.println("失败处理" + input.size() + ",已处理" + afterNum);
14. }
15. CtyunTsdbRequestConfig config = CtyunTsdbRequestConfig
16. .address("www.example.com ", 8585)
17. .listenBatchPut(cb) // 设置回调接口
18. .config();
19. tsdb = CtyunTsdbRequestFactory.connect(config);
```

TSDB Client 提供了以下三种类型的 PutCallback 接口：

- PostDataCallback，即调用 `POST /api/put`
- PostDataSummaryCallback，即调用 `POST /api/put?summary=true`
- PostDataDetailsCallback，即调用 `POST /api/put?details=true`

您可以通过设置不同类型的 Callback 来根据需要选择返回的数据内容。不要在回调方法中做耗时操作。若有此需要，可以在把操作交给其他工作线程执

5.5 查询

tsdb 的查询有同步查询和异步返回结果查询两种方法。

5.5.1 同步查询

首先需要构造 Query 对象，一个 Query 对象即是一个查询条件：

示例代码

```
SubQuery subQuery =
SubQuery.metric("test.com").aggregator(AggregatorType.AVG).build();

Query query = Query.start(0).end(System.currentTimeMillis()).query(subQuery).
build();

client.query(query);
```

一个 Query 可以设置多个子查询 SubQuery(即多个查寻条件)。

5.5.2 异步返回结果查询

使用 query 方法同步的查询数据，并通过设置回调来设置异步查询后的行为

示例代码

```
QueryCallback cb = new QueryCallback() {

    @Override

    public void response(Query input, List<QueryResult> result) {

        System.out.println("查询参数：" + input);

        System.out.println("返回结果：" + result);

    }

};

tsdb.query(query, cb);
```

注意：TSDB 默认情况下，查询数据返回的时间戳是以秒为单位的，所以如果插点的时候时间戳为毫秒，那么查询这个 metric 返回的数据点是以秒为单位的（毫秒级的数据会丢失）

5.5.3 值过滤

值过滤是针对特定时间线过滤出指定范围的值。使用方法就是在指定的子查询里面指定一组特定的过滤器。

值过滤类型类型有图有以下几种：

1. eq 等于
2. ge 大于等于
3. gt 大于
4. le 小于等于
5. lt 小于
6. nq 不等于
7. bt between value=bt(10~100)

值过滤实例

```
List<Filter> lists = new ArrayList<Filter>();

Filter filter1 =
Filter.type(FilterType.ILITERAL_OR).tagk("city").filter("shanghai").build();

Filter filter2 = Filter.type(FilterType.BT).tagk("value").filter("1~2").build();

lists.add(filter1);

lists.add(filter2);

SubQuery subQuery =
SubQuery.metric("filter").filters(lists).aggregator(AggregatorType.NONE).build()
;

Query query =
Query.start(0).end(System.currentTimeMillis()).query(subQuery).build();
```

```
client.query(query, qcb);
```

5.5.4 过滤器

过滤器的类型有以下几种：

- iliteral_or 和 literal 类似，但不区分大小写
- wildcard 通配符
- not_literal_or 和 literal_or 相反
- not_iliteral_or 和 iliteral_or 相反
- iwildcard 同 wildcard 相同，但不区分大小写
- literal_or 采用单个字面值或| 管道分隔值列表并返回与区分大小写的结果匹配的时间序列
- regexp 正则表达式

过滤器需要在构造 SubQuery 的时候进行设置。

示例代码

```
SubQuery subQuery =
SubQuery.metric("test.com").filter(Filter.type(FilterType.LITERAL_OR).t
agk("diquA").filter("beijing").build()).aggregator(AggregatorType.NONE)
.build();

Query query = Query.start(0).end(System.currentTimeMillis() /
1000).query(subQuery).build();
```

5.5.5 查询最新时间点的数据

查询最新的时间点有 2 个接口，一个是利用 tsdb 原生的 api/query/last 接口进行查询最新时间点的数据。第二个是为了查找最新的多个数据点而扩展的功能接口。

原生接口示例代码

```
    SubLast subLast = SubLast.metric("test.com").build();

    Last last = Last.query(subLast).build();

System.out.println("result>>>"+ client.last(last));
```

扩展功能接口示例代码

```
    SubQuery subQuery
    =SubQuery.metric("test.com").aggregator(AggregatorType.NONE).build();

    Query query
    =Query.start(0).end(System.currentTimeMillis()).query(subQuery).build();

    System.out.println("\nLast>>" + client.last(query, 2).toString()+"\n");
```

扩展功能的实例代码是查找最新的 2 个时间点

5.5.6 根据 metric 查询时间线

方法 queryTsmetaByMetric 是针对 meta 表中的时间线的信息查询。和其它方法一样，在 TSDB 客户端中直接调用即可。

方法说明如下

```
    /**
         * @param metric
         * @return JSONArray
         * @description 根据 /api/uid/tsmeta?m=metric 查找其元数据集合
         */
```

5.5.7 根据 metric 查询标签

根据 metric 查询标签的接口是 Lookup，下面是查询 metric 为 example 的所有标签对。

示例代码

```
Lookup lk = Lookup.metric("example").limit(1).build();

String result = client.lookup(lk);
```

5.5.8 降采样查询

5.5.8.1 降采样说明

当查询的时间范围比较长，需返回一定精度的数据时使用。查询格式如下：

```
<interval><units>-<aggregator>[-fill policy]
```

其中：

- interval：指数值，如 5、60 等，特殊的“0all”表示时间维度聚合为一个点。
- units：s 代表秒，m 代表分，h 代表小时，d 代表天，dc 代表当前时区的天。
- aggregator：降采样使用的算子及其说明如下表所示。

算子	描述	插值方法
avg	平均值	线性插值（斜率拟合）
count	数据点数	插 0
first	取第一个值	-
last	取最后一个值	-
mimmin	最小值	插最大值
mimmax	最大值	插最小值
min	最小值	线性插值
max	最大值	线性插值
sum	求和	线性插值
zimsum	求和	插 0

5.5.8.2 Fill policy

Fill policy 即填值。降采样先把所有时间线按照指定精度切分，并把每个降采样区间内的数据做一次运算，降采样后如果某个精度区间没有值，fill policy 可以指定在这个时间点填充具体的值。比如某条时间线降采样后的时间戳为：t+0，t+20，t+30，此时如果不指定 fill policy，只有 3 个值，如果指定了 fill policy 为 null，此时间线会有 4 个值，其中 t+10 时刻的值为 null。

Fill policy 与具体填充值的对应如下表所示。

Fill Policy	填充值
none	默认行为，不填值
nan	NaN
Null	与 NaN 的行为相同，只是在序列化过程中它发出的是 null，而不是 NaN。
Zero	0

5.6 关闭客户端

5.6.1 优雅关闭

TSDB-Client 提供了优雅关闭功能。客户端在关闭过程中会被阻塞住，直到服务器处理完所有请求。建议您使用优雅关闭功能，以防数据丢失。

示例代码

```
1. tsdb.close(); // 等价于 tsdb.close(false);
```

5.6.2 强制关闭

若不需要使用优雅关闭功能，可以直接关闭客户端。

示例代码

```
1. tsdb.close(true);
```

5.7 调用其他接口

5.7.1 清理数据

删除一个 metric 在一段时间内的数据也是使用 Query 接口类进行删除操作，只需要将 delete 设置为 true 即可。

删除 metric 为 test 在一段时间段内的数据示例代码

```
SubQuery subQuery =
SubQuery.metric("test").aggregator(AggregatorType.NONE).build();

Query query =
Query.start(0).end(System.currentTimeMillis()).delete(true).query(subQuery)
.build();

// String result = client.query(query);

// 异步查询

client.query(query, qcb);
```

注意：方法并没有将 metric 全部删除。

5.7.2 删除时间线

删除时间线包括实际数据的删除和元数据的删除。有 2 个接口可供调用，一个接受 TimeLine 参数，一个接受 List<String tsuid>参数。

删除一个时间线的数据示例代码（组件 timeline 删除）

```
Timeline timeline = Timeline.metric("test").tag("diqu",
"guangzhou").build();

client.deleteTimeLine(timeline);
```

删除多个时间线的数据示例代码（根据 tsuid 删除）

```
List<String> tsuids=new ArrayList<String>();
tsuids.add("000062000001000028 ");
tsuids.add("000062000001000029");
client.deleteTimeLine(tsuids);
```

6 TSDB HTTP API

6.1 简介

标准的 Restful 风格是 TSDB API 采用的设计风格，一个唯一的 URL 确定一个 API 功能。对资源的访问是通过标准的 HTTP 请求进行的，包括请求的 GET、PUT、POST 等方法，其中包含的请求参数需要遵守约定的方式。

6.2 URL 规范

6.2.1 通用

- 所有编码都采用 UTF-8
- 日期格式采用 yyyy-MM-dd 方式，如 2015-08-10
- Content-type 为 application/json; charset=UTF-8
- object 类型的 key 必须使用双引号（“）括起来

6.2.2 请求消息体格式

- 请求消息体为 JSON 格式。下面是一个标准的写入一个 Point 的消息体：

```
{
    "metric": "m1",
    "value": 2,
    "timestamp":621977199,
    "tags": {
       "city": "guangzhou"
```

```
        }
    }
```

6.3 API 接口调用

6.3.1 写入数据

路径和方法：

请求路径	请求方法	功能
/api/put	POST	POST

请求参数：

参数	是否必须	描述	默认值
summary	否	是否返回摘要信息	fasle
details	否	是否返回详细信息	false

请求内容:

名称	类型	是否必需	是否有使用限制	描述
metric	String	是	英文，数字	指标名称
timestamp	Long	是	无	见下面的时间戳说明
tags	Map	是	英文字母、数字	数据点标签
value	Long, Double, String, Boolean	是	无	数据点值

时间戳注意事项：

本说明适用于读写数据（`/api/put`）和查询数据（`api/query`）两个接口。

时间戳的单位可以是秒或者毫秒。TSDB 会通过数值大小来判断时间戳的单位，规则如下：

1. 时间戳区间为 [4284768,9999999999]：判断为秒，表示的时间区间为:[1970-02-20 00:59:28, 2286-11-21 01:46:39
2. 时间戳区间为 [10000000000,9999999999999]：判断为毫秒，表示的时间区间为：[1970-04-27 01:46:40.000, 2286-11-21 01:46:39.999]
3. 时间戳区间为 (-∞,4284768)和(9999999999999,+∞)：判断为非法时间戳区间

数据值注意事项：

1. 同一时间线只允许写入同一种数据类型。
2. String 数值类型的数据值可以为任意字符，支持 JSON 字符串存储，最大为 20KB。

写入数据实例：

请求：/POST/api/put

请求体：

```
    [
      {
        "metric": "sys.cpu.core",
        "timestamp": 1226846400,
        1."value": 18,
        "tags": {
            "host": "1",
            "core": 1
        }
      },
      {
        "metric": "sys.cpu. core",
        "timestamp": 1226846400,
```

```
        "value": 9,
        "tags": {
            "host":  “2”,
            "core": 2
        }
      }
    ]
```

响应：

如果所有数据点写入成功，返回码为 204，如果有部分写入失败，返回码 400，响应内容为具体错误信息。

如果请求参数包含 summary 或者 details，返回信息包括如下属性：

参数	值类型	描述
success	Integer	写入成功的数据点数
failed	Integer	写入失败的数据点数
errors	Array	描述哪些数据点未写入以及其原因的数组，仅在指定 details 有效

请求参数为 summary 的响应实例：

请求：/POST/api/put?summary

请求体：

```
[
    {
        "metric": "sys.cpu.nice",
        "timestamp": 1226846400,
        "value": 9,
        "tags": {
           "host": "web00",
           "core": 1
        }
  }
]
```

响应：

```
{
  "failed": 1,
  "success": 0
}
```

请求参数为 details 的响应实例：

请求：/POST/api/put?details

请求体：

```
[
    {
        "metric": "sys.cpu.nice",
        "timestamp": 1226846400,
        "value": 9,
        "tags": {
           "host": "web00",
           "core": 1
        }
  }
]
```

响应：

```
{
        "errors": [
            {
                "datapoint": {
                    "metric": "sys.cpu.nice",
                    "timestamp": 1365465600,
                    "value": "NaN",
                    "tags": {
```

```
                        "host": "web01"
                    }
                },
                "error": "Unable to parse value to a number"
            }
        ],
        "failed": 1,
        "success": 0
    }
```

表示 0 个点写入成功，1 个点写入失败。失败的点由 datapoint 指定。该点写入失败的原因由 error 字段指定。

6.3.2 查询数据

6.3.2.1 查询请求方法

路径	方法	描述
/api/query	POST	查询数据

6.3.2.2 请求体

请求体为 JSON 格式，具体如下：

参数	类型	必须	描述
start	Long	是	开始时间
end	Long	否	结束时间
queries	Array	是	子查询条件

6.3.2.3 子查询

子查询格式为 JSON 格式，具体如下：

参数	类型	是否必须	描述
metric	String	是	度量指标
tags	Map	是	标签
aggregator	String	是	聚合函数
filters	List	否	过滤器
downsample	String	否	时间维度采样
limit	Integer	否	数据点限制

6.3.3 值过滤

值过滤是针对特定时间线过滤出指定范围的值。使用方法就是在指定的子查询里面指定一组特定标签。

值过滤类型类型有图有以下几种：

eq 等于
ge 大于等于
gt 大于
le 小于等于
lt 小于
nq 不等于
bt between value=bt(-10~100)

值过滤请求体：

```
{
    "start":0,
    "queries":[
        {
            "aggregator":"none",
            "metric":"metric_test",
            "tags":{
                "value":"lt(4)"
            }
```

```
            }
        ]
```

6.3.4 标签过滤器

有以下两种方法可以指定 filter：

在指定 tagk 时指定 filter：

1. tagk = *：对 tagk 下面的 tagv 做 groupBy，相同的 tagv 做聚合。
2. tagk = tagv1|tagv2： 分别对 tagk 下面的 tagv1 和 tagv2 数据做聚合。

使用 JSON 格式指定 filter：

参数	类型	是否必须	描述
type	String	是	过滤器类型
tagk	String	是	标签 key
filter	String	是	过滤表达式
groupBy	Boolean	是	是否对 tagv 做 groupBy

过滤器实例:

```
{
    "start": 1356998400,
    "end": 1356998460,
    "queries": [
        {
            "aggregator": "sum",
            "metric": "sys.cpu.0",
            "rate": "true",
            "filters": [
                {
                    "type":"wildcard",
                    "tagk":"host",
```

```
                    "filter":"*",
                    "groupBy":true
                },
                {
                    "type":"literal_or",
                    "tagk":"dc",
                    "filter":"lga|lga1|lga2",
                    "groupBy":false
                }
            ]
        },
        {
            "aggregator": "sum",
            "tsuids": [
                "000001000002000042",
                "000001000002000043"
            ]
        }
    ]
}
```

6.3.5 查询结果

查询成功的 HTTP 响应码为 200，响应内容为 JSON 格式数据。说明如下：

名称	说明
metric	指标名
tags	tagv 未做聚合的 tag
aggregateTags	tagv 做了聚合的 tag
dps	数据点对

查询结果实例：

```
[
    {
        "metric": "tsd.hbase.puts",
```

```
            "tags": {},
            "aggregatedTags": [
                "host"
            ],
            "dps": [
                [
                    1365966001,
                    25595461080
                ],

    ...
                [
                    1365974221,
                    25722266376
                ]
            ]
        }
    ]
```


6.3.6 查询 Metric、Tagk、Tagv

路径	方法	描述
/api/suggest	POST	查询 Metric、Tagk、Tagv

请求内容：

名称	类型	是否必须	描述
type	String	是	需要查询的类型，metrics，tagk，tagv
q	String	否	前缀过滤
max	Integer	否	最大返回个数

请求实例：

任务:查询 4 条以 “my” 为前缀的指标名称。

请求：POST/api/suggest

请求体：

```
{
"type": "metrics",
"q": "my_",
"max": 4
}
```

响应说明：

响应为查询到的字符串数组：

```
[ "my_metric1", "my_metric2", "my_metric3", "my_metric4" ]
```

6.3.7 查询时间线最新时间点

请求路径和方法：

路径	请求方法	描述
/api/query/last	POST	获得时间线最新写入的数据点

请求内容：

名称	类型	是否必需	描述
metric	String	是	最新时间点的 metric
tags	String	否	最新时间点标签

根据 tsuids 查询：

名称	类型	是否必需	描述
tsuid	String	是	最新时间线的

			tsuid

注意：时间线的 TSUID 可以通过 /api/search/lookup 接口查询。

请求 JSON 实例：

查询包含 metric 名字为“temperature” 和 tag 名字为”name=name1”的时间线的最新写入数据点。同时，查询 TSUID 为 00005B00005C00002E00005D0000EE00005E0000EF 的最新写入数据点。

请求：

POST /api/query/last

请求体：

```
{
    "queries": [{
        "metric": " temperature ",
        "tags": {
            " name=name1"
        }
    },
    {
        "tsuids": [ “00005B00005C00002E00005D0000EE00005E0000EF” ]
    }]
}
```

响应说明：

返回符合条件的时间线最新数据点。如果没有时间线满足条件，则返还空集。

6.4 清理数据

6.4.1 清理数据点

请求路径和方法：

路径	请求方法	描述
/api/query	POST	删除数据点

请求内容：

名称	类型	是否必需	描述
metric	String	是	时间点的 metric
tags	String	否	时间点标签
delete	Boolean	无	删除

删除时间点实例：

删除 metric 名字为“temperature” 和 tag 名字为”name=name1”的时间线的数据点。

请求：

POST /api/query

请求体：

```
{
    "queries": [{
        "metric": " temperature ",
        "tags": {
            " name=name1"
        }
                                    },
]
"delete":true
}
```

6.4.2 清理时间线

请求路径和方法：

路径	请求方法	描述
/api/uid/tsmeta	GET	删除时间线

请求内容：

名称	类型	是否必需	描述
tsuid	String	是	时间点的 metric
method_override	String	否	删除标记

删除时间线实例：

删除 tsuid 为"00002A000001000001"的时间线。

请求：

```
GET  /api/uid/tsmeta?tsuid=00002A000001000001&method_override=delete
```

以上操作的实质是删除 meta 表中的时间线信息和时间线计数器。

6.5 聚合函数

6.5.1 Avg

名称	说明	支持类型
Avg	平均值，以每个采样时间范围内的 value 的平均值作为	Number

请求参数

参数名称	参数类型	是否必须	说明
sampling	String	可选	采样的时间长度，如“10 min-utes”，若不填写则

6.5.2 Dev

名称	说明	支持类型
Dev	标准差，以每个采样时间范围内的value的标准差作为	Number

请求参数

参数名称	参数类型	是否必须	说明
sampling	String	可选	采样的时间长度，如“10 min- utes”，若不填写则 sampling 为整个查询时间范围

6.5.3 Count

名称	说明	支持类型
Count	数目，以每个采样时间范围内的点的数目作为	Number/String

请求参数

参数名称	参数类	是否必	说明
sampling	String	可选	采样的时间长度，如“10 min- utes”，若不填写则 sampling 为整个查询时间范围

6.5.4 First

名称	说明	支持类型
First	第一个，以每个采样时间范围内的第一个value作为结	Number/String

请求参数

参数名称	参数类型	是否必须	说明
sampling	String	可选	采样的时间长度，如“10 min- utes”，若不填写则 sampling 为整个查询时间范围

6.5.5 Last

名称	说明	支持类型
Last	最后一个，以每个采样时间范围内的最后一个 value 作为结	Number/String

请求参数

参数名称	参数类型	是否必	说明
sampling	String	可选	采样的时间长度，如“10 min- utes”，若不填写则 sampling 为整个查询时间范围

6.5.6 LeastSquares

名称	说明	支持类型
LeastSquares	最小二乘法，对每个采样时间范围内的 value 进行最小二乘法的拟合	Number

请求参数

参数名称	参数类型	是否必须	说明
sampling	String	可选	采样的时间长度，如“10 min- utes”，若不填写则 sampling 为整个查询时间范围

6.5.7 Max

名称	说明	支持类型
Max	最大值，以每个采样时间范围内的value的最大值作为	Number

请求参数

参数名称	参数类型	是否必须	说明
sampling	String	可选	采样的时间长度，如“10 minutes”，若不填写则sampling为整个查询时间范围

6.5.8 Min

名称	说明	支持类型
Min	最小值，以每个采样时间范围内的value的最小值作为结果	Number

请求参数

参数名称	参数类型	是否必须	说明
sampling	String	可选	采样的时间长度，如“10 minutes”，若不填写则sampling为整个查询时间范围

6.5.9 Percentile

名称	说明	支持类型
Percentile	百分位数，以每个采样时间范围内的value的p（见请求参数percentile）百分位数作为结果	Number

请求参数

参数名称	参数类型	是否必	说明
sampling	String	可选	采样的时间长度， 如“10 minutes”，若不填写则 sampling 为整个查询时间范围
percentile	Double	必须	百分数，取值范围为 (0,1]，如 0.1 表示 10%

6.5.10 Sum

名称	说明	支持类型
Sum	和值，以每个采样时间范围 内的 value 的总和	Number

请求参数

参数名称	参数类型	是否必须	说明
sampling	String	可选	采样的时间长度， 如“10 minutes”， 若不填写则 sampling 为整个查询时间范围

6.5.11 Diff

名称	说明	支持类型
Diff	差值，以每两个相邻的 value 的差值作为结果	Number

6.5.12 Div

名称	说明	支持类型
Div	以每个 value 除以一个除数（见请求参数 divisor）作为结果	Number

请求参数

参数名称	参数类型	是否必须	说明
divisor	Double	必须	除数，不能为 0

6.5.13 Scale

名称	说明	支持类型
Scale	以每个 value 乘以一个倍数（见请求参数 factor）作为结果	Number

请求参数

参数名称	参数类型	是否必须	说明
factor	Double	必须	倍数

6.5.14 Rate

名称	说明	支持类型
Rate	变化率，以每两个相邻的 value 在单位时间（见参数 timeUnit)的变化率作为结果	Number

请求参数

参数名称	参数类型	是否必须	说明
timeUnit	String	必须	时间单位

6.5.15 AdjacentUnique

名称	说明	支持类型

AdjacentUnique	相邻值去重，相同的值只保 留第一个，非相邻的值不去 重。譬如“1，1，2，2，2，1，1，3”，相邻值去重后的结果为“1，2，1，3”	Number/String/Bytes

7 常见问题

7.1 TSDB 有哪些使用约定？

- 一个请求最多写入 50 个数据点（data point）。
- 一个数据点（data point）最多 8 个标签（tag）。
- 一个查询最多返回 10 万个数据点（data point）。

7.2 如何选择 TSDB 的实例规格？

您可根据以下两个指标选择 TSDB 的实例规格：

- 时间线数：对应需要采集数据的设备的总的采集点数。一个采集点可以是某个设备上需要上传数据的一个传感器。计算公式：总设备数 * 单个设备的采集点数（或传感器数）。
- 写入效率：全局每秒需要写入时序数据库的平均数据点数（数据记录数）。计算公式：时间线数 * 单条时间线每秒需要写入的平均数据点数。

7.3 为什么我的示例数据导入提示成功，但是查询没有数据？

- 可能是您开启了“数据时效功能”。在开启数据时效功能后，过了数据时效的数据，是无法写入系统的，因此是查看不到数据的；
- 可能是查询的时间范围设置错误，示例数据的是 2017 年 1 月 1 日至 2017 年 5 月 31 日的数据。

7.4 如何写入数据到数据库中？

目前有两种方式：

- 一种是通过数据导入，对导入的数据量是有限制的，适合数据量不大的场景。
- 一种是通过 SDK 写入。实时数据可以通过 SDK 进行写入，此时用户需要申请 ECS 主机用于数据接入，部署自己的应用，适合于线上场景。

7.5 如何进行数据查询？

天翼云 TSDB 提供三种查询方式：

- 使用数据探索，查看数据；
- 使用查询面板，进行复杂查询；
- 使用 SDK 进行查询。

7.6 连接 TSDB 有什么注意事项？

- 必须将用于连接的 ECS 主机的 IP 添加到白名单：

  详见安全设置章节

- 申请访问令牌，访问请求中添加令牌：

  详见令牌管理章节

7.7 我的存储空间快使用满了，该怎么办？

当您在管理控制台监控到空间使用快满时，可以选择如下任一种方式：

- 在天翼云订购页面对存储进行扩容（默认可扩容的最大存储为 3T，如果需要扩容超过 3T 则需联系客服经理处理）；
- 可设置数据有效期，超过有效期的数据将自动清理（系统可设置数据有效期 90 天）；
- 将部分不再使用的数据备份到其他存储，并清理掉。

7.8 TSDB 如何进行数据可视化展示？

TSDB 目前支持以下两种方式进行数据可视化展示：

- 通过 TSDB 控制台对数据进行可视化展示。详情请见查询面板。
- 通过接入 Grafana 对数据进行可视化展示。将 TSDB 接入 Grafana 后，您可以利用 Grafana 的丰富易用的可视化工具更好地监控和分析来自 TSDB 的数据。