阿里云
表格存储

计算与分析

文档版本：20190919

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
	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。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能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，恢复业务所需时间约10分钟。
	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。	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。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	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。	`swich` *`{stand | slave}`*

目录

1 函数触发器

1.1 使用函数计算

函数计算（Function Compute）是一个事件驱动的服务，通过函数计算，用户无需管理服务器等运行情况，只需编写代码并上传。

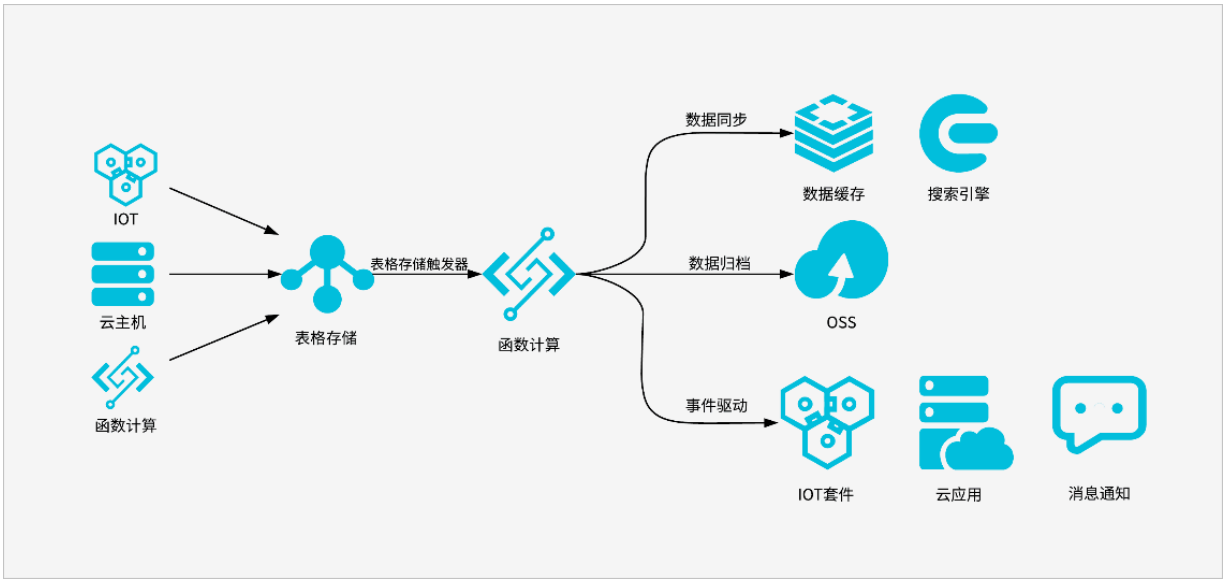
函数计算准备计算资源，并以弹性伸缩的方式运行用户代码，而用户只需根据实际代码运行所消耗的资源进行付费。详细信息请参见函数计算帮助文档。

函数计算示例请参见表格存储函数计算示例。

Tablestore Stream是用于获取Tablestore表中增量数据的一个数据通道，通过创建Tablestore触发器，能够实现Tablestore Stream和函数计算的自动对接，让计算函数中自定义的程序逻辑自动处理Tablestore表中发生的数据修改。详细信息请参见Tablestore Stream。

本节介绍如何使用函数计算对表格存储增量数据进行实时计算。

使用场景：



- 数据同步：同步表格存储中的实时数据到数据缓存、搜索引擎或者其他数据库实例中。
- 数据归档：增量归档表格存储中的数据到 OSS 等做冷备份。
- 事件驱动：利用触发器触发函数调用 IOT 套件、云应用 API 或者做消息通知等。

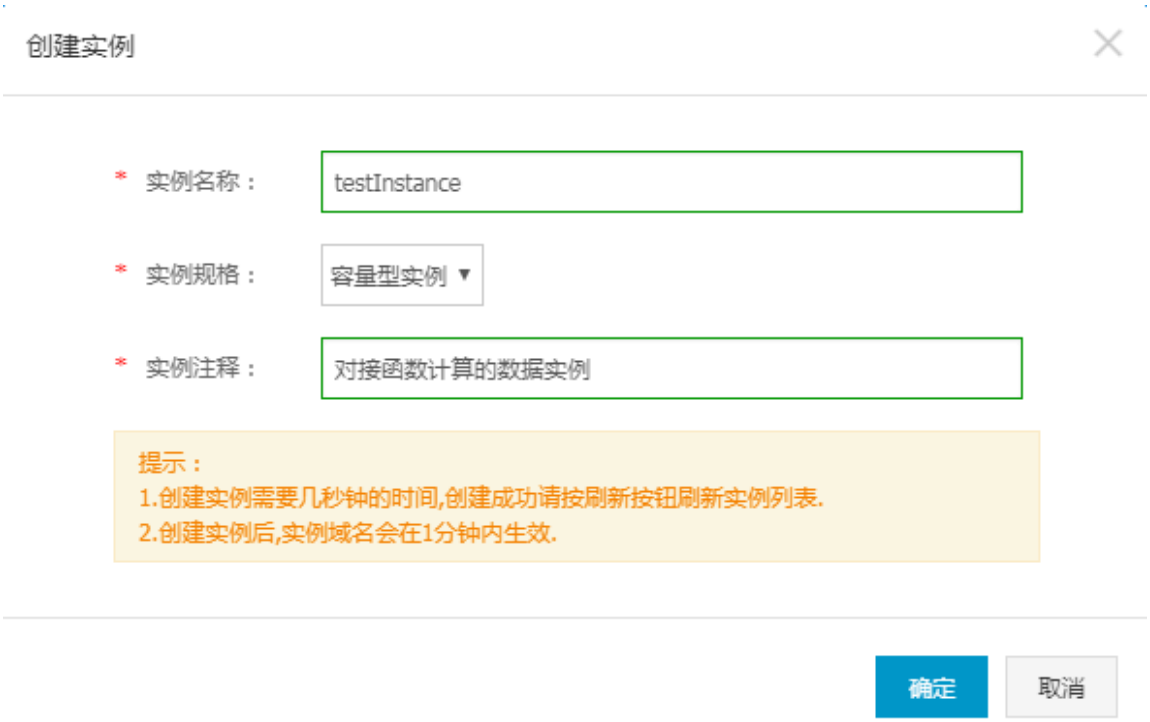
触发器详细信息请参见Tablestore 触发器。

配置Tablestore触发器

您可以通过控制台创建Tablestore触发器来处理Tablestore数据表的实时数据流。

1. 创建开通了Stream流的数据表。

   a) 在表格存储控制台创建实例。

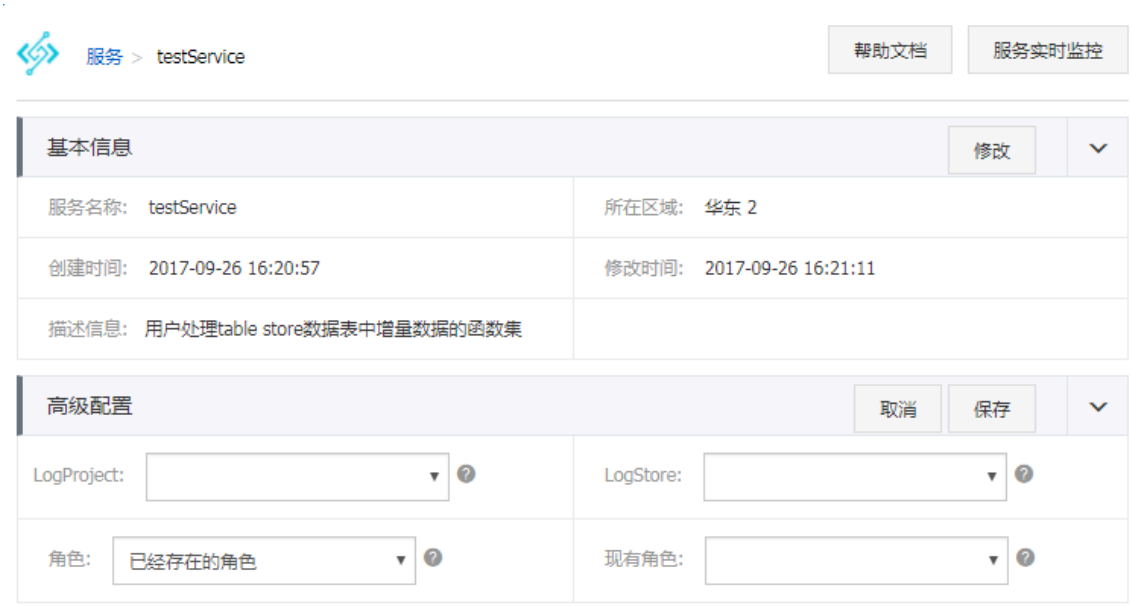


   b) 在该实例下创建数据表，并且开启数据表的Stream功能。

2. 创建函数计算的函数。

a) 在函数计算控制台创建服务（Service）。

b) 在Service的高级配置中，您可以配置服务的角色，用于授权函数进行日志收集，以及在计算函数中继续访问用户的其他资源。详细信息请参见函数计算权限模型。



c) 在新创建的Service下单击新建函数。

d) 在函数模板页面选择 空白函数。

e) 在触发器配置页面选择不创建任何触发器，然后单击下一步。

f) 配置函数信息。

Tablestore触发器会使用CBOR格式将增量数据编码为函数计算的Event，并调用用户函数。下图示例函数就是将编码后的Event重新解码和打印到日志中心，您可以在解码数据后进行任意数据处理。


服务 > testService > 函数
帮助文档
服务实时监控
函数模版
触发器配置
基础管理配置
信息核对
基础信息
* 函数名称：
streamProcessFunction
命名规范：
» 1. 只能包含字母，数字、下划线和中划线
» 2. 不能以数字、中划线开头
» 3. 长度限制在1-128之间
函数描述：
用于处理tablestore数据表增量数据的函数
* 运行环境：
python2.7
代码配置
* 代码上传方式：
在线编辑
OSS上传
本地上传
#!/usr/bin/env python
# coding=utf-8
import cbor
import logging
def handler(event, context):
logger = logging.getLogger()
records = cbor.loads(event)
logger.info('Tablestore stream records sample %s', records)
return 'ok'
环境配置
* 函数入口：
index.handler
格式为"[文件名].[函数名]"。例如hello_world.handler指定了函数的调用入口为hello_world.js文件中的handler函数。如果是代码直接上传，代码将保存为对应的文件名hello_world.js，只需关注函数名填写。
* 函数内存：
128
MB
* 超时时间：
3
秒
取消
上一步
下一步

3. 创建和测试Tablestore触发器。

   a) 在表格存储管理控制台新创建的数据表下，选择使用已有函数计算来创建触发器。

   b) 创建过程中需要授权Tablestore发送事件通知所需的权限。

      勾选完毕后，实际可以在RAM控制台查看到自动创建的授权角色AliyunTableStoreStreamNotificationRole。

数据处理

- 数据格式

  Tablestore触发器使用CBOR格式对增量数据进行编码构成函数计算的Event。增量数据的具体数据格式如下：

```
{
    "Version": "string",
    "Records": [
        {
            "Type": "string",
            "Info": {
                "Timestamp": int64
            },
            "PrimaryKey": [
                {
                    "ColumnName": "string",
                    "Value": formated_value
                }
            ],
            "Columns": [
                {
                    "Type": "string",
                    "ColumnName": "string",
                    "Value": formated_value,
                    "Timestamp": int64
                }
            ]
        }
    ]
```

```
}
```

- 成员定义
  - Version
    - 含义: payload版本号，目前为Sync-v1
    - 类型: string
  - Records
    - 含义: 数据表中的增量数据行数组
    - 内部成员:
      - Type
        - 含义: 数据行类型，包含PutRow、UpdateRow和DeleteRow
        - 类型: string
      - Info
        - 含义: 数据行基本信息
        - 内部成员:
          - Timestamp
          - 含义: 该行的最后修改UTC时间
          - 类型: int64
      - PrimaryKey
        - 含义: 主键列数组
        - 内部成员
          - ColumnName
            - 含义: 主键列名称
            - 类型: string
          - Value
            - 含义: 主键列内容
            - 类型: formated_value，支持integer、string和blob
      - Columns
        - 含义: 属性列数组
        - 内部成员
          - Type

- 含义: 属性列类型，包含Put、DeleteOneVersion和DeleteAllVersions
- 类型: string

- ColumnName
  - 含义: 属性列名称
  - 类型: string
- Value
  - 含义: 属性列内容
  - 类型: formated_value，支持integer、boolean、double、string和blob
- Timestamp
  - 含义: 属性列最后修改UTC时间
  - 类型: int64

· 数据示例

```
{
    "Version": "Sync-v1",
    "Records": [
        {
            "Type": "PutRow",
            "Info": {
                "Timestamp": 1506416585740836
            },
            "PrimaryKey": [
                {
                    "ColumnName": "pk_0",
                    "Value": 1506416585881590900
                },
                {
                    "ColumnName": "pk_1",
                    "Value": "2017-09-26 17:03:05.8815909 +0800 CST"
                },
                {
                    "ColumnName": "pk_2",
                    "Value": 1506416585741000
                }
            ],
            "Columns": [
                {
                    "Type": "Put",
                    "ColumnName": "attr_0",
                    "Value": "hello_table_store",
                    "Timestamp": 1506416585741
                },
                {
                    "Type": "Put",
                    "ColumnName": "attr_1",
                    "Value": 1506416585881590900,
                    "Timestamp": 1506416585741
                }
            ]
```

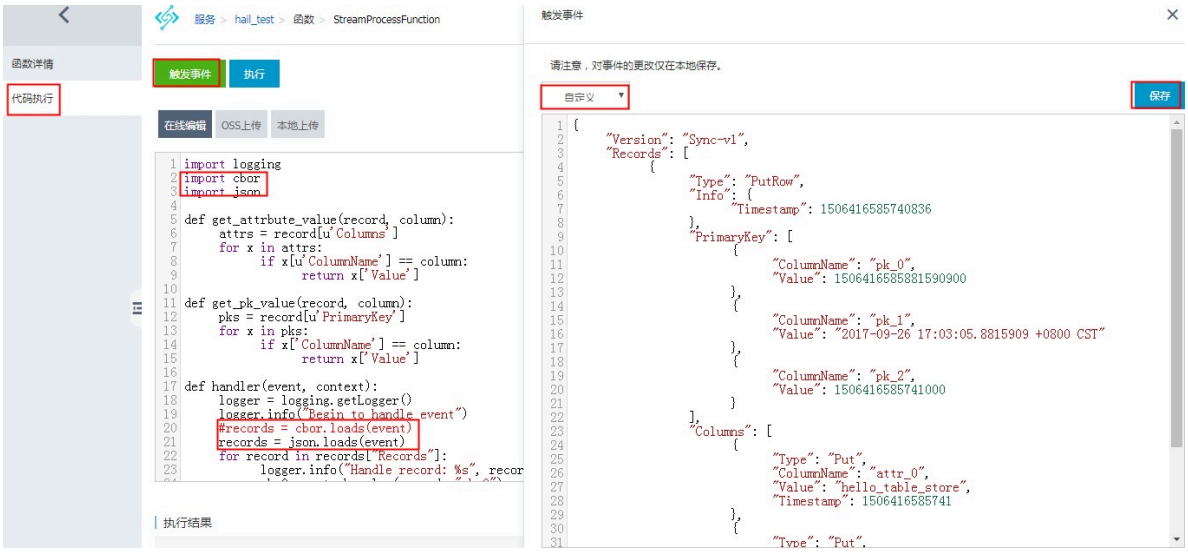
```
            }
        ]
}
```

在线调试

函数计算支持函数的在线调试功能，用户能够自己构建触发的Event，并测试代码逻辑是否符合期望。

由于Tablestore触发函数服务的Event是CBOR格式，该数据格式是一种类似JSON的二进制格式，可以按照如下方法进行在线调试：

1. 在代码中同时import CBOR和JSON。
2. 单击触发事件，选择自定义，将上述数据示例的JSON文件粘贴至编辑框，根据实际情况修改，并保存。
3. 代码中先使用 `records = json.loads(event)` 来处自定义的测试触发事件，单击执行，查看是否符合期望。
4. 当使用 `records=json.loads(event)` 测试OK之后，可以修改为 `records = cbor.loads(event)` 并保存，当Tablestore中有数据写入时，相关的函数逻辑就会触发。



示例代码：

```
import logging
import cbor
import json
def get_attrbute_value(record, column):
    attrs = record[u'Columns']
    for x in attrs:
        if x[u'ColumnName'] == column:
            return x['Value']
def get_pk_value(record, column):
    attrs = record[u'PrimaryKey']
    for x in attrs:
        if x['ColumnName'] == column:
```

```
            return x['Value']
def handler(event, context):
    logger = logging.getLogger()
    logger.info("Begin to handle event")
    #records = cbor.loads(event)
    records = json.loads(event)
    for record in records['Records']:
        logger.info("Handle record: %s", record)
        pk_0 = get_pk_value(record, "pk_0")
        attr_0 = get_attrbute_value(record, "attr_0")
    return 'OK'
```

1.2 使用函数计算做数据清洗

表格存储高并发的写入性能以及低廉的存储成本非常适合物联网、日志、监控数据的存储，您可以将数据写入到表格存储中，同时在函数计算中对新增的数据做简单的清洗，并将清洗之后的数据写回到表格存储表中，并对原始及结果数据提供实时访问。

数据定义

我们假设写入的为日志数据，包括三个基础字段：

字段名称	类型	含义
id	整型	日志id
level	整型	日志的等级，越大表明等级越高
message	字符串	日志的内容

我们需要将 level>1 的日志写入到另外一张数据表中，用作专门的查询。

创建实例及数据表

在表格存储控制台创建表格存储实例（本次以 华东2 distribute-test 为例），并创建源表（source_data）及结果表（result），主键为 id (整型)，由于表格存储是 schemafree 结构，无需预先定义其他属性列字段。

以 source_data 为例，创建如下图：

开启数据源表的Stream功能

触发器功能需要先开启数据表的Stream功能，才能在函数计算中处理写入表格存储中的增量数据。

Stream记录过期时长是指通过 StreamAPI 能够读取到的增量数据的最长时间。

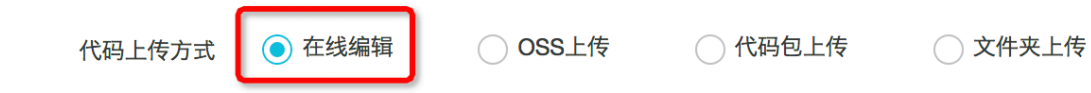
由于触发器只能绑定现有的函数，故先到函数计算的控制台上在同region创建服务及函数。

创建函数计算服务

在函数计算的控制台上创建服务及处理函数，我们继续使用华东2节点。

1. 在华东2节点创建服务。

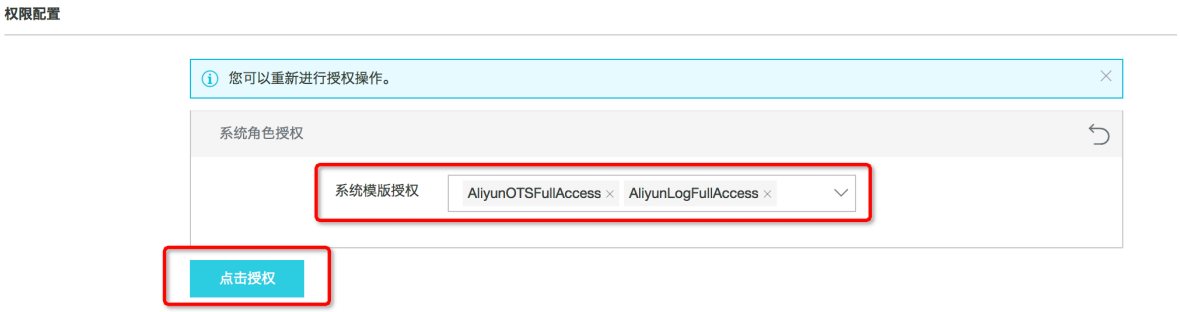
2. 创建函数依次选择空白函数 > 不创建触发器。



- 函数名称为：etl_test，选择 python2.7 环境，在线编辑代码
- 函数入口为：etl_test.handler
- 代码稍后编辑，点击下一步。

3. 进行服务授权。

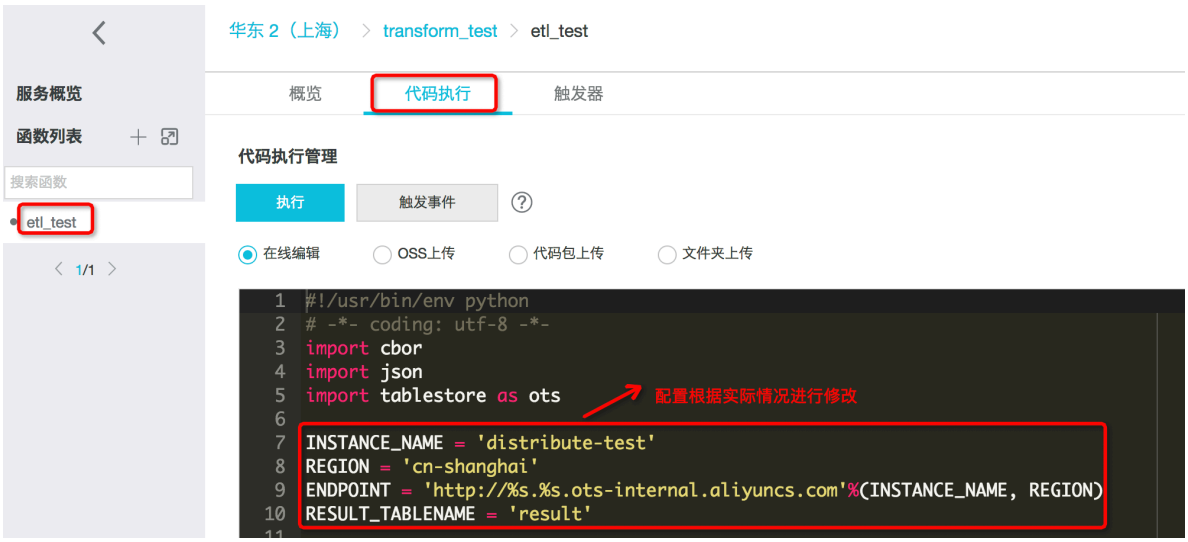
   由于函数计算需要将运行中的日志写入到日志服务中，同时，需要对表格存储的表进行读写，故需要对函数计算进行授权，为方便起见，我们先添加 AliyunOTSFullAccess 与 AliyunLogFullAccess 权限，实际生产中，建议根据权限最小原则来添加权限。



4. 单击授权完成，并创建函数。

5. 修改函数代码。

   创建好函数之后，点击对应的函数—代码执行，编辑代码并保存，其中，INSTANCE_NAME(表格存储的实例名称)、REGION(使用的区域)需要根据情况进行修改：



   使用示例代码如下：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import cbor
import json
import tablestore as ots
INSTANCE_NAME = 'distribute-test'
REGION = 'cn-shanghai'
ENDPOINT = 'http://%s.%s.ots-internal.aliyuncs.com'%(INSTANCE_NAME,
REGION)
RESULT_TABLENAME = 'result'
def _utf8(input):
```

```
    return str(bytearray(input, "utf-8"))
def get_attrbute_value(record, column):
    attrs = record[u'Columns']
    for x in attrs:
        if x[u'ColumnName'] == column:
            return x['Value']
def get_pk_value(record, column):
    attrs = record[u'PrimaryKey']
    for x in attrs:
        if x['ColumnName'] == column:
            return x['Value']
#由于已经授权了AliyunOTSFullAccess权限，此处获取的credentials具有访问表格存储的权限
def get_ots_client(context):
    creds = context.credentials
    client = ots.OTSClient(ENDPOINT, creds.accessKeyId, creds.
accessKeySecret, INSTANCE_NAME, sts_token = creds.securityToken)
    return client
def save_to_ots(client, record):
    id = int(get_pk_value(record, 'id'))
    level = int(get_attrbute_value(record, 'level'))
    msg = get_attrbute_value(record, 'message')
    pk = [(_utf8('id'), id),]
    attr = [(_utf8('level'), level), (_utf8('message'), _utf8(msg
)),]
    row = ots.Row(pk, attr)
    client.put_row(RESULT_TABLENAME, row)
def handler(event, context):
    records = cbor.loads(event)
    #records = json.loads(event)
    client = get_ots_client(context)
    for record in records['Records']:
        level = int(get_attrbute_value(record, 'level'))
        if level > 1:
            save_to_ots(client, record)
        else:
```

```
            print "Level <= 1, ignore."
```

绑定触发器

1. 回到表格存储的实例管理页面，单击表 source_data 后的使用触发器按钮，进入触发器绑定界面，单击使用已有函数计算， 选择刚创建的服务及函数，勾选表格存储发送事件通知的权限，进行确定。

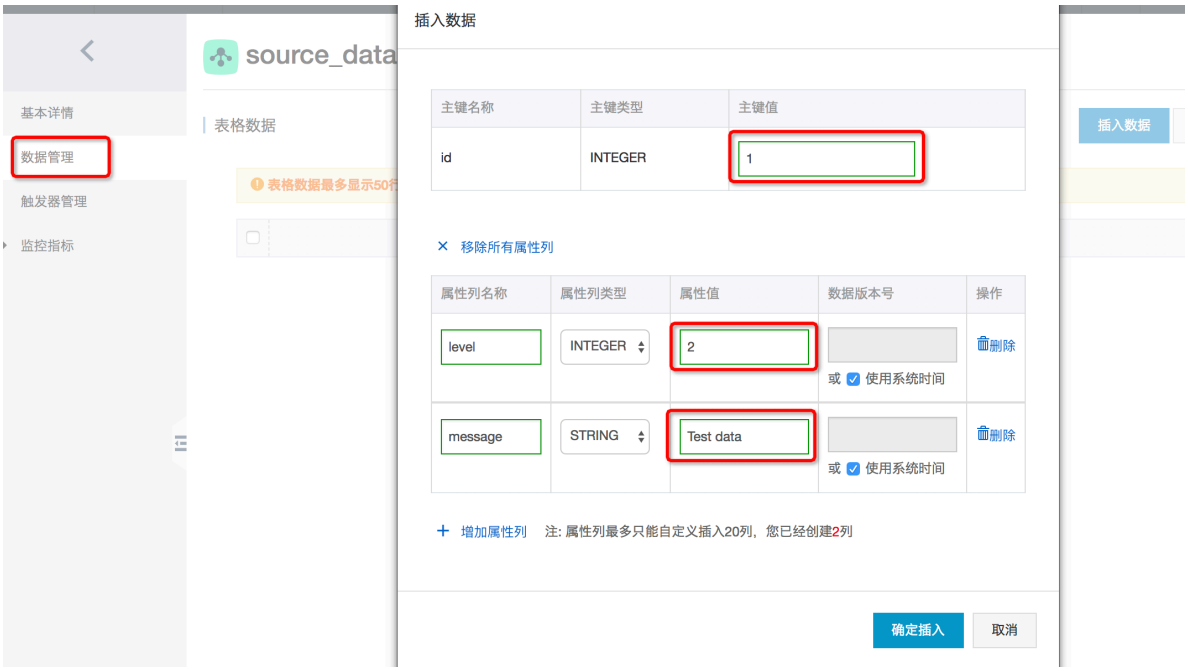

source_data
创建触发器
新建函数计算
使用已有
触发器列表
目前一张数据表只支持创建
函数计算服务名
创建触发
* 选择
* 选择

2. 绑定成功之后，能够看到如下的信息：

运行验证

1. 向 source_data 表中写入数据。

   单击source_data的数据管理页面，然后单击插入数据，如图依次填入id、level及message信息。



2. 在 result 表中查询清洗后的数据 。

   单击result表的数据管理页面，会查询到刚写入到 source_data 中的数据。

   向 soure_data 写入level <=1的数据将不会同步到 result 表中。

2 数据展现

本文主要为您介绍如何使用DataV将表格存储的数据可视化。

背景信息

表格存储的表数据支持接入DataV数据可视化。DataV可以将数据由单一的数字转化为各种动态的可视化图表，根据表数据生成数据看板，实时地将数据展示给需要的用户。

操作步骤

1. 在DataV控制台配置表格存储数据源。
   a) 登录DataV控制台。
   b) 选择我的数据 > 添加数据 。
   c) 单击类型下拉菜单，选择TableStore。填写Tablestore相关信息。

参数	说明
名称	数据源的显示名称，您可以自由命名。
AK ID	拥有Tablestore访问权限的账号的AccessKey ID。AccessKey ID获取方式参见AccessKey。
AK Secret	拥有Tablestore访问权限的账号的AccessKey Secret。AccessKey Secret获取方式参见AccessKey。
外网	Tablestore的#unique_10，需要根据访问的Tablestore实例来填写。

   d) 信息填写完成后，单击确定，完成数据源的添加。新添加的数据源会自动显示在数据源列表中。
2. 在DataV控制台上，单击我的可视化，选择您的项目，单击编辑，进入大屏编辑界面。
3. 单击选择某一组件，在数据面板中，选择数据源类型为TableStore。
4. 单击选择操作下拉列表选择需要的操作，系统支持以下两种操作方式。
   - `getRow`：对应Tablestore的`GetRow` API，详情请参见GetRow API 参考。
   - `getRange`：对应Tablestore的`GetRange` API，详情请参见GetRange API 参考。

5. 在选择操作下的编辑框中输入查询语句。

   - 查询参数必须为JSON对象。
   - 选择getRow操作时，需要根据指定的主键读取单行数据。

     参数格式如下。

     ```
     {
     "table_name": "test",
     "rows": {
     "id": 2
     },
     "columns": [
     "id",
     "test"
     ]
     }
     ```

     - table_name：填写您需要查询的Table名。
     - rows：填写行的过滤条件，将筛选出符合条件的行返回。如果您需要在rows里面添加column作为查询条件，那么所添加的column必须是建立过索引的。
     - columns：填写需要返回的列名。
   - 选择getRange操作，可读取指定主键范围内的数据，参数格式如下。

     ```
     {
     "table_name": "test",
     "direction": "FORWARD",
     "columns": [
     "id",
     "test"
     ],
     "range": {
     "limit": 4,
     "start": {
     "id": "InfMin"
     },
     "end": {
     "id": 3
     }
     }
     ```

```
}
```

- table_name：填写您需要查询的 Table 名。
- direction：查询的顺序。
- columns：填写需要返回的列名。
- limit：读取最多返回的行数。
- start：指定读取时的起始列，返回的结果中包含当前起始列，所列出的 column 必须是已经建立索引的列。
- end：指定读取时的结束列，返回的结果中不包含当前结束列，所列出的 column 必须是已经建立索引的列。

说明:

start和end参数里可以使用InfMin、InfMax表示最小值和最大值 。

6. 单击查看数据响应结果，数据响应成功后即可看到效果。

3 MaxCompute

3.1 使用MaxCompute访问表格存储

本文主要为您介绍如何在同一个云账号下实现表格存储和 MaxCompute 之间的无缝连接。

背景信息

MaxCompute是一项大数据计算服务，它能提供快速、完全托管的 PB 级数据仓库解决方案，使您可以经济并高效地分析处理海量数据。您只需通过一条简单的 DDL 语句，即可在 MaxCompute 上创建一张外部表，建立 MaxCompute 表与外部数据源的关联，提供各种数据的接入和输出能力。MaxCompute 表是结构化的数据，而外部表可以不限于结构化数据。

表格存储与 MaxCompute 都有其自身的类型系统，两者之间的类型对应关系如下表所示。

Tablestore	MaxCompute
STRING	STRING
INTEGER	BIGINT
DOUBLE	DOUBLE
BOOLEAN	BOOLEAN
BINARY	BINARY

准备工作

使用 MaxCompute 访问表格存储前，您需要完成以下准备工作：

1. 开通 MaxCompute 服务。
2. 创建 #unique_15/unique_15_Connect_42_section_jhg_s4g_r2b。
3. 创建 AccessKey。
4. 在 RAM 控制台授权 MaxCompute 访问表格存储的权限。具体参见同账号授权、跨账号授权。

5. 在表格存储控制台创建实例和创建数据表。

   在本示例中，创建的表格存储实例和数据表信息如下：

   - 实例名称：cap1
   - 数据表名称：vehicle_track
   - 主键信息：vid (integer)，gt (integer)
   - 访问域名：`https://cap1.cn-hangzhou.ots-internal.aliyuncs.com`

   > 说明：
   > 使用 MaxCompute 访问表格存储时，建议使用表格存储的私网地址。

   - 设置实例网络类型为允许任意网络访问。

步骤一： 安装并配置客户端

1. 下载 MaxCompute 客户端并解压。

   > 说明：
   > 确保您的机器上已安装 JRE 1.7或以上版本。

2. 编辑 conf/odps_config.ini 文件，对客户端进行配置，如下所示：

```
access_id=********************
access_key=**********************
 # Accesss ID 及 Access Key 是用户的云账号信息，可登录阿里云官网，进入管理控制台 – accesskeys 页面进行查看。
project_name=my_project
 # 指定用户想进入的项目空间。
end_point=https://service.odps.aliyun.com/api
 # MaxCompute 服务的访问链接
tunnel_endpoint=https://dt.odps.aliyun.com
```

```
 # MaxCompute Tunnel 服务的访问链接
log_view_host=http://logview.odps.aliyun.com
 # 当用户执行一个作业后，客户端会返回该作业的 LogView 地址。打开该地址将会看到作业执行的详细信息
https_check=true
 #决定是否开启 HTTPS 访问
```

说明:

odps_config.ini 文件中使用#作为注释，MaxCompute 客户端内使用--作为注释。

3. 运行 bin/odpscmd.bat，输入 `show tables;` 。

   如果显示当前 MaxCompute 项目中的表，则表述上述配置正确。

```
odps@ MCOTStest>show tables;

ALIYUN$document@aliyun-test.com:bank_data
ALIYUN$document@aliyun-test.com:result_table

OK
```

步骤二： 创建外部表

创建一张 MaxCompute 的数据表（ots_vehicle_track）关联到 Tablestore 的某一张表（vehicle_track）。

关联的数据表信息如下。

- 实例名称：cap1
- 数据表名称：vehicle_track
- 主键信息：vid (int); gt (int)
- 访问域名：`https://cap1.cn-hangzhou.ots-internal.aliyuncs.com`

```
CREATE EXTERNAL TABLE IF NOT EXISTS ots_vehicle_track
(
vid bigint,
gt bigint,
longitude double,
latitude double,
distance double ,
speed double,
oil_consumption double
)
STORED BY 'com.aliyun.odps.TableStoreStorageHandler' -- (1)
WITH SERDEPROPERTIES ( -- (2)
'tablestore.columns.mapping'=':vid, :gt, longitude, latitude, distance, speed, oil_consumption', -- (3)
'tablestore.table.name'='vehicle_track' -- (4)
)
```

```
LOCATION 'tablestore://cap1.cn-hangzhou.ots-internal.aliyuncs.com'; --
 (5)
```

参数说明如下：

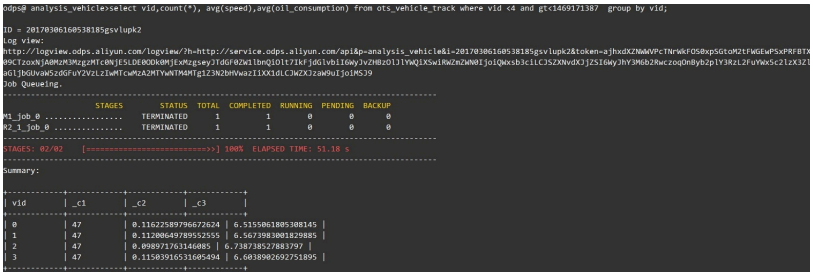
标号	参数	说明
(1)	com.aliyun.odps.TableStore StorageHandler	MaxCompute 内置的处理 Tablestore 数据的 StorageHandler，定义了 MaxCompute 和 Tablestore 的交互，相关逻辑由 MaxCompute 实现。
(2)	SERDEPROPERITES	可以理解为提供参数选项的接口，在使用 TableStoreStorageHandler 时，有两个必须指定的选项，分别是 tablestore.columns.mapping 和 tablestore.table.name。
(3)	tablestore.columns.mapping	必填选项。MaxCompute 将要访问的 Tablestore 表的列，包括主键和属性列。 其中，带:的表示 Tablestore 主键，例如本示例中的 :vid 与 :gt，其他均为属性列。在指定映射的时候，用户必须提供指定 Tablestore 表的所有主键，属性列无需全部提供，可以只提供需要通过 MaxCompute 来访问的属性列。
(4)	tablestore.table.name	需要访问的 Tablestore 表名。 如果指定的 Tablestore 表名错误（不存在），则会报错。MaxCompute 不会主动创建 Tablestore 表。
(5)	LOCATION	指定访问的 Tablestore 的实例信息，包括实例名和 endpoint 等。

步骤三：通过外部表访问 Tablestore 数据

创建外部表后，Tablestore 的数据便引入到了 MaxCompute 生态中，您可通过 MaxCompute SQL 命令来访问 Tablestore 数据。

```
// 统计编号 4 以下的车辆在时间戳 1469171387 以前的平均速度和平均油耗
select vid,count(*),avg(speed),avg(oil_consumption) from ots_vehicl
e_track where vid <4 and gt<1469171387  group by vid;
```

返回类似如下结果：



3.2 同账号授权

本文主要为您介绍如何使用阿里云访问控制的RAM角色实现同账号MaxCompute访问表格存储。

背景信息

RAM角色（RAM role）与RAM用户一样，都是RAM身份类型的一种。RAM角色是一种虚拟用户，没有确定的身份认证密钥，需要被一个受信的实体用户扮演才能正常使用。具体参见#unique_20。

操作步骤

1. 登录RAM控制台。
2. 在左侧导航栏，选择RAM角色管理。
3. 单击新建RAM角色，创建用户角色AliyunODPSDefaultRole。
4. 在左侧导航栏，选择权限管理 > 权限策略管理，进入权限策略管理页面。
5. 单击新建权限策略。
6. 策略名称输入AliyunODPSRolePolicy。配置模式选择脚本配置，然后单击确定。策略内容如下：

```
{
"Version": "1",
"Statement": [
 {
   "Action": [
     "ots:ListTable",
     "ots:DescribeTable",
     "ots:GetRow",
     "ots:PutRow",
     "ots:UpdateRow",
     "ots:DeleteRow",
     "ots:GetRange",
     "ots:BatchGetRow",
     "ots:BatchWriteRow",
     "ots:ComputeSplitPointsBySize"
   ],
   "Resource": "*",
   "Effect": "Allow"
 }
]
}
```

```
--还可自定义其他权限
```

7. 在左侧导航栏，选择RAM角色管理。找到AliyunODPSDefaultRole角色，单击添加权限。

8. 在添加权限页面，选择权限选择自定义权限策略，找到并单击AliyunODPSRolePolicy，单击确定完成角色授权。
9. 使用目标RAM用户扮演AliyunODPSDefaultRole角色。具体参见#unique_21。

完成以上步骤，目标RAM用户的下的MaxCompute即可访问表格存储。

3.3 跨账号授权

本文主要为您介绍不同账号之间如何实现表格存储和 MaxCompute 之间的无缝连接。

> 说明:
> 如需了解同账号下的表格存储与 Maxcompute 对接操作，请参考同账号下使用 MaxCompute 访问表格存储。

准备工作

跨云账号需要两个主账号，账号 A 将访问权限授予账号 B，则运行 MaxCompute 时，账号 B 可以访问账号 A 下的表数据。基本信息如下：

> 说明:
> 以下信息仅为示例，在操作时请替换为实际使用的信息。

项目	表格存储	MaxCompute
主账号名	账号 A	账号 B
UserId	12345	56789

使用 MaxCompute 跨云账号访问表格存储前，您需要完成以下准备工作：

1. 账号 B 开通MaxCompute 服务，并#unique_15/unique_15_Connect_42_section_jhg_s4g_r2b。
2. 账号 A 和 B 分别创建 AccessKey
3. 使用账号 A 登录 RAM 控制台，并在RAM角色管理页面，新建RAM角色。在本示例中，假设创建的角色名称为 AliyunODPSRoleForOtherUser。

4. 在RAM角色列表中，找到AliyunODPSRoleForOtherUser角色，然后单击RAM角色名称，设置策略内容。 策略内容设置如下：

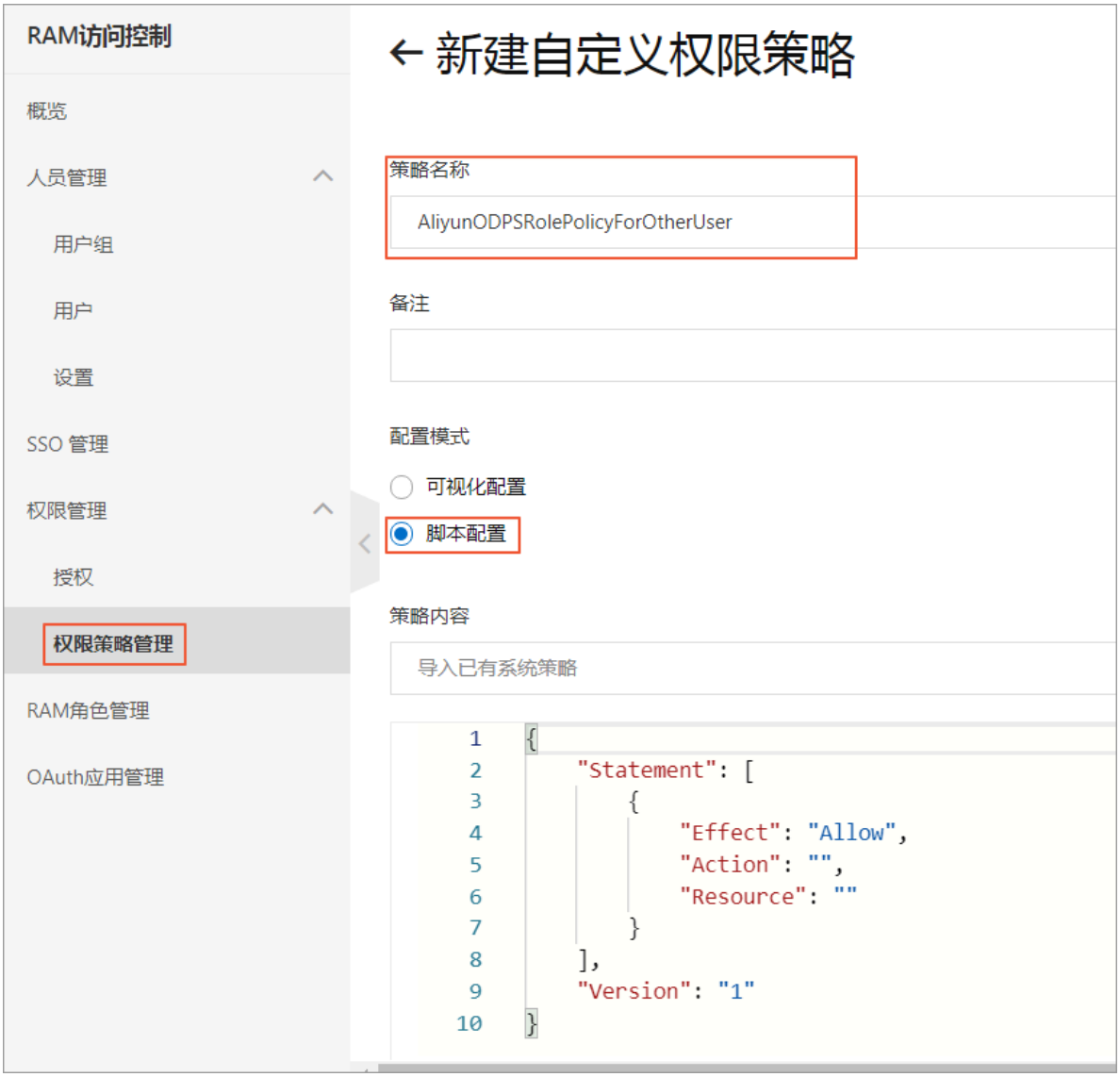
```
{
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "1xxxx@odps.aliyuncs.com"
        ]
      }
    }
  ],
  "Version": "1"
}
```

说明:

请将上述策略内容中的 1xxxx 替换成您的 UID 即可。

5. RAM角色创建后，您可以在基本信息页面查看该角色的ARN。

6. 返回 RAM 控制台首页，进入权限策略管理页面，单击新建权限策略。



7. 在RAM角色管理页面，找到AliyunODPSRoleForOtherUser角色，然后单击添加权限。
8. 在添加权限页面，选择AliyunODPSRolePolicyForOtherUse权限，然后单击确定。
9. 在表格存储控制台创建实例和创建数据表。

   在本示例中，创建的表格存储实例和数据表信息如下：

   - 实例名称：cap1
   - 数据表名称：vehicle_track
   - 主键信息：vid (integer)，gt (integer)
   - 访问域名：`https://cap1.cn-hangzhou.ots-internal.aliyuncs.com`

   说明：

使用 MaxCompute 访问表格存储时，建议使用表格存储的私网地址。

- 设置实例网络类型为允许任意网络访问。

使用 MaxCompute 访问表格存储

跨账号访问的操作与同账号下的访问一样，只是在创建外部表时使用 roleArn。

账号 B 通过 MaxCompute 创建外部表，指定准备工作中创建出来的 roleArn 来访问表格存储。

具体操作步骤请参考同账号授权访问。其中，在步骤 2 创建外部表时，使用如下代码：

```
CREATE EXTERNAL TABLE ads_log_ots_pt_external
(
vid bigint,
gt bigint,
longitude double,
latitude double,
distance double ,
speed double,
oil_consumption double
)
STORED BY 'com.aliyun.odps.TableStoreStorageHandler'
WITH SERDEPROPERTIES (
'tablestore.columns.mapping'=':vid, :gt, longitude, latitude, distance
, speed, oil_consumption',
'tablestore.table.name'='vehicle_track',
'odps.properties.rolearn'='acs:ram::12345:role/aliyunodpsroleforoth
eruser'
)
LOCATION 'tablestore://cap1.cn-hangzhou.ots-internal.aliyuncs.com'
USING 'odps-udf-example.jar'
```

3.4 使用AccessKey访问表格存储

除了授权方式外，您还可以在 MaxCompute 中使用 AccessKey 访问表格存储的数据。

准备工作

获取表格存储资源所属账号的AccessKeyId 和 AccessKeySecret，如果该 AK 是资源所属账号的子账号，那么该子账号至少需要对表格存储相关的资源具有以下权限：

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "ots:ListTable",
        "ots:DescribeTable",
        "ots:GetRow",
        "ots:PutRow",
        "ots:UpdateRow",
        "ots:DeleteRow",
        "ots:GetRange",
        "ots:BatchGetRow",
        "ots:BatchWriteRow",
```

```
        "ots:ComputeSplitPointsBySize"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}

--您也可以自定义其他权限
```

在 MaxCompute 中使用 AccessKey 访问表格存储

同授权方式不同的是，需要在创建外表时在LOCATION中显示写入 AK 信息，其格式为：

```
LOCATION 'tablestore://${AccessKeyId}:${AccessKeySecret}@${InstanceNa
me}.${Region}.ots-internal.aliyuncs.com'
```

假设需要访问的表格存储资源的信息为：

AccessKeyId	AccessKeySecret	实例名称	区域	网络模式
abcd	1234	cap1	cn-hangzhou	内网访问

创建外表的语句为：

```
CREATE EXTERNAL TABLE ads_log_ots_pt_external
(
vid bigint,
gt bigint,
longitude double,
latitude double,
distance double ,
speed double,
oil_consumption double
)
STORED BY 'com.aliyun.odps.TableStoreStorageHandler'
WITH SERDEPROPERTIES (
'tablestore.columns.mapping'=':vid, :gt, longitude, latitude, distance
, speed, oil_consumption',
'tablestore.table.name'='vehicle_track'
)
LOCATION 'tablestore://abcd:1234@cap1.cn-hangzhou.ots-internal.
aliyuncs.com'
```

对数据访问的操作步骤请参考使用MaxCompute访问表格存储中的步骤3.通过外部表访问 Table Store 数据。

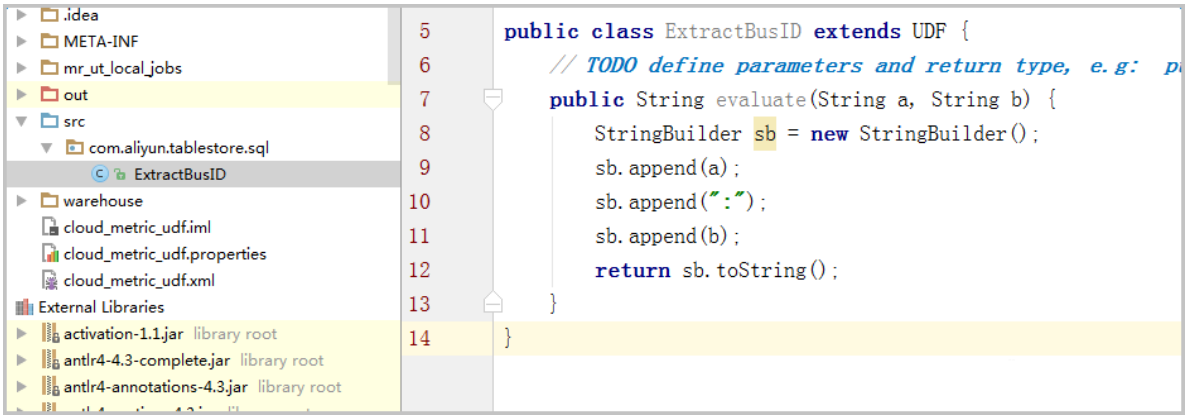
3.5 使用UDF处理数据

如果您在表格存储里面的数据有着独特的结构，希望自定义开发逻辑来处理每一行数据，比如解析特定的 JSON 字符串，可以使用 UDF（User Defined Function，即用户自定义函数）来处理。

操作步骤

1. 参考 MaxCompute Studio 文档，在 IntelliJ 中安装 MaxCompute-Java/MaxCompute-Studio 插件。插件安装完毕，就可以直接开发。

   下图是一个简单的 UDF 定义，将两个字符串连接。MaxCompute 支持更复杂的 UDF，包括自定义窗口执行逻辑等，更多信息请参考开发和调试 UDF。

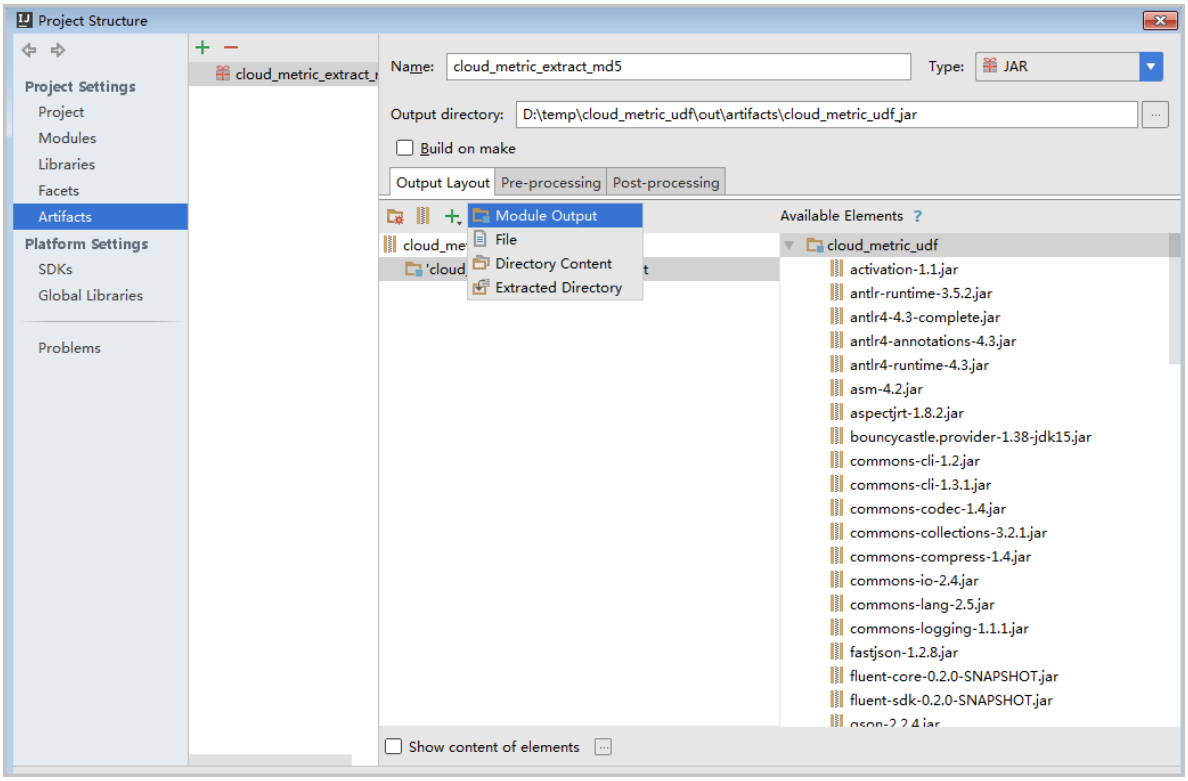


2. 参考MaxCompute Studio 文档，在 IntelliJ 中安装 MaxCompute-Java/MaxCompute-Studio 插件。插件安装完毕，就可以直接开发。

   下图是一个简单的 UDF 定义，将两个字符串连接。MaxCompute 支持更复杂的 UDF，包括自定义窗口执行逻辑等，更多信息请参考开发和调试 UDF。

3. 包之后可以上传到 MaxCompute。

   选择 File > Project Structure > Artifacts，输入Name 和 Output directory 后，单击 + 选择输出模块。打包后通过 ODPS Project Explorer 来上传资源、创建函数，然后就可以在 SQL 中调用。



4. 运行bin/odpscmd.bat。

```
// 我们选出来1行数据，并将name/name传入UDF，返回两个string的累加
select cloud_metric_extract_md5(name, name) as udf_test from
test_table limit 1;
```

返回结果如下：

```
odps@ table_store_sql_engine_dev>select cloud_metric_extract_md5(c,c) as udf_test from cloud_metric_stable limit 1;

ID = 20170302055324953gq1tsau1
Log view:
http://logview.odps.aliyun-inc.com:8080/logview/?h=http://service-corp.odps.aliyun-inc.com/api&p=table_store_sql_eng
055324953gq1tsau1&token=d2l4cGJkSk9VRW1GQkNmNXZCV0JOZWQ4T21zPSxPRFBTX09CTzoxNDE0MDcwMjYwNjg3NzQ1LDE0ODkwMzg4MDUseyJ
FjdGlvbiI6WyJvZHBzOlJlYWQiXSwiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnByb2plY3RzL3RhYmxlX3N0b3JlX3NxbE
c3RhbmNlcy8yMDE3MDMwMjA1NTMyNDk1M2dxMXRzYXUxIl19XSwiVmVyc2lvbiI6IjEifQ==
Job Queueing...
QuotaCPUUsage: 99.99%    QuotaMemUsage: 79.36%
--------------------------------------------------------------------------------
                STAGES        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  BACKUP
M1_job_0 .............    TERMINATED      1          1        0        0       0
R2_1_job_0 ............    TERMINATED      1          1        0        0       0
--------------------------------------------------------------------------------
STAGES: 02/02    [==========================>>] 100%  ELAPSED TIME: 350.08 s
--------------------------------------------------------------------------------
Summary:

+----------+
| udf_test |
+----------+
| code4xx1,0.00,netflow,2512570.00,qps,2989.00,p99RT,95607.60,code5xx,0.00,MaxRT,432553.00,MinRT,0.00,AvgRT,9940.51
```

3.6 常见问题

本文主要为您介绍使用MaxCompute访问表格存储的相关常见问题。

- FAILED: ODPS-0010000:System internal error - fuxi job failed, WorkerPackageNotExist

  原因：需要设置 set odps.task.major.version=unstructured_data。
- FAILED: ODPS-0010000:System internal error - std::exception:Message: a timeout was reached

  原因：一般情况下是表格存储的 endpoint 填写错误，导致 MaxCompute 无法访问。
- logview invalid end_point

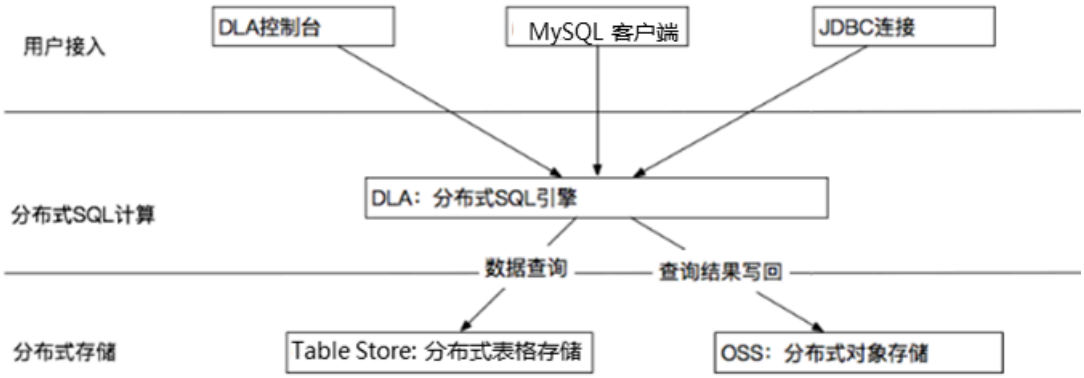
  原因：在执行过程中，会返回 logview URL 地址，如果使用浏览器访问该地址返回错误，可能是配置不对，请检查 MaxCompute 配置。

如果仍未解决问题，请提交工单。

4 Data Lake Analytics

4.1 背景信息

Tablestore 中接入 Data Lake Analytics（简称 DLA）服务的方式，为您提供一种快速的 OLAP（On-Line Analytical Processing） 解决方案。DLA 是阿里云一款通用的 SQL 查询引擎，使用通用的 SQL 语言（兼容 MySQL 5.7 绝大部分查询语法）可在 Tablestore 中进行灵活的数据分析。



如架构图所示，OLAP查询架构涉及阿里云DLA、Tablestore 和 OSS 三款产品。

- DLA：负责分布式 SQL 查询计算。在实际运行过程中将 SQL 查询请求进行任务拆解，产生若干可并行化的子任务，提升数据计算和查询能力。
- Tablestore：数据存储层，用于接收 DLA 的各类子查询任务。如果您在 Tablestore 中已经有存量数据，可以直接在 DLA 上建立映射视图，从而体验 SQL 计算的便捷服务。
- OSS：分布式对象存储系统，主要用于保存查询结果。

如果您要在 Tablestore 中体验分布式 SQL 计算，须开通 Tablestore、DLA 和 OSS 服务。

> 说明:
>
> - 开通OSS服务的主要原因是 DLA 默认将查询结果集数据写入 OSS，因此需要引入一个额外的存储依赖。您仅需开通 OSS 服务，无需预先创建 OSS 存储实例。
> - 目前开服公测的区域是华东2（上海），对应的实例是该 region 内所有的容量型实例。在开通 DLA 服务时，需要先填写公测申请。详情请参考准备工作文档。

4.2 准备工作

如果您要在 Tablestore 中体验分布式 SQL 计算，须开通 Tablestore、Data Lake Analytics 和 OSS 服务。本文主要为您介绍如何开通这些服务。

> 说明:
> 完成Tablestore、Data Lake Analytics 和 OSS 服务接入后，实际查询将会产生相应的费用。在实际查询过程中如果您的账号欠费，则查询失败。

开通 Tablestore 服务

如果您已经开通 Tablestore 服务，并且已创建实例和数据表，则忽略该步骤。

如果您首次使用 Tablestore，可按照如下步骤开通 Tablestore 并创建实例和数据表：

1. #unique_31
2. #unique_32
3. #unique_33
4. #unique_34

开通 OSS 服务

如果您首次使用 OSS，需#unique_35。

开通 Data Lake Analytics 服务

如果您首次使地域用 Data Lake Analytics，需先开通 Data Lake Analytics 服务。开通 DLA 服务后，按照如下步骤申请 DLA 账号：

1. 登录 DLA 管理控制台，选择开通对应地域的 DLA 服务实例（如华东 2 上海区域），然后单击初始化服务。

   > 说明:
   > 不同的地域对应不同的账号，且不同地域之间的DLA账号不能混用。

2. 在云产品开通页面，单击立即开通。

   > 说明:
   > 账号创建完成之后会收到相关邮件（邮箱为阿里云的注册邮箱），内含该region的DLA账号和密码，注意查收。

3. 选择地域，单击右上角的Tablestore授权。

4. 在云资源访问授权页面，单击同意授权，授权 DLA 访问 Tablestore 中的用户实例数据，如下图所示：

4.3 使用DLA服务

开通服务后，可通过控制台、MySQL Client 以及 JDBC 这三种方式接入 DLA 服务并进行 SQL 查询。

Tablestore 和 DLA 元信息映射逻辑

- 库和表等概念映射

Tablestore	DLA
实例（instance）	schema 或 database
表（table）	table
主键列（pk）	column，isPrimaryKey=true，isNullable=false
非主键列（column）	column，isPrimaryKey=false，isNullable=<用户的DDL定义>

- 字段的映射关系

Tablestore	DLA
INTEGER(8bytes)	bigint(8bytes)
STRING	varchar
BINARY	varbinary(目前主键中不支持)
DOUBLE	double

Tablestore	DLA
BOOLEAN	byte

控制台访问 DLA

控制台访问 DLA 步骤如下：

1. 使用邮件中随附的该地域的用户名和密码登录数据库。
2. 为 Tablestore 中的实例表格数据建立映射。

假设您在上海地域已创建一个名为 sh_tpch 的实例，该实例包含表格 test001，表格内包含 2 行测试数据。该实例建立映射的步骤如下：

1. 将 Tablestore 的实例映射成 DLA 的一个 DataBase 实例。建立 DLA 的 Database 映射前，首先需要在 Tablestore 中创建实例，如创建一个名为 sh-tpch 的实例，对应的 endpoint 为 sh-tpch.cn-shanghai.ots.aliyuncs.com。

   完成测试实例创建后，执行下列语句建立 Database 映射：

```
CREATE SCHEMA sh_tpch001 with DBPROPERTIES(LOCATION ='https://sh-tpch.cn-shanghai.ots.aliyuncs.com', catalog='ots', instance ='sh-tpch');
```

> **说明：**
> 使用 MySQL Client 时，可以使用 create database 或 create schema 语句创建 database 映射。但是控制台目前只支持 create schema 语句创建 database 映射。

2. 在 tp_tpch001 的 Database 下建立表格的映射。在建立 DLA 的表格映射前，首先需要在 Tablestore 中创建数据表。

   数据表创建完成后，执行下列语句建立表格映射：

```
CREATE EXTERNAL TABLE test001 (pk0 int NOT NULL , primary key(pk0));
```

> **说明：**
> 建立 DLA 映射表时，指定的 Primary Key 必须与 Tablestore 表格定义 Primary Key 列表一致。Primary Key 用于唯一确定一行的数据，一旦映射表的 Primary Key 列表与 Tablestore 表格的 PK 不一致，可能导致 SQL 查询结果出现非预期错误。

   例如，您的 Tablestore 实例 sh_tpch 中包含 test001 表格，其中只有一列 pk0。使用 show 命令可查看该表已创建成功：

3. 使用select语句执行SQL查询：

   - 查出所有数据：

```
select * from test001;
```

- **执行count统计：**

```
select count(*) from test001;
```

- **执行sum统计：**

```
select sum(pk0) from test001;
```

执行SQL查询时，可以选择同步执行结果，返回满足条件的前 10,000 条记录；如要获取大结果集数据，请选择异步执行，并使用show query_id的方式异步获取结果：

```
show query_task where id = '59a05af7_1531893489231';
```

其他执行语句，请查看如下说明文档：

- create schema语句
- create table语句
- select语句
- show语句
- drop table语句
- drop schema语句

MySQL Client 访问 DLA

您可以使用标准的 MySQL Client 快速接入 DLA 的数据实例，其连接语句为：

```
mysql -h service.cn-shanghai.datalakeanalytics.aliyuncs.com -P 10000 -
u <username> -p <password> -c -A
```

说明:

其他操作语句与控制台访问一致。

JDBC 访问 DLA

您还可以使用标准的 Java API 访问 DLA，其连接语句为：

```
jdbc:mysql://service.cn-shanghai.datalakeanalytics.aliyuncs.com:10000/
```

说明：

其他操作语句与控制台访问一致。

5 Hive/HadoopMR

5.1 环境准备

本文主要为您介绍使用 Hive/HadoopMR 来访问表格存储中的表前的环境准备。

使用 Hive/HadoopMR 来访问表格存储中的表

通过表格存储及 E-MapReduce 官方团队发布的依赖包，可以直接使用 Hive 及 HadoopMR 来访问表格存储中的数据并进行数据分析。

安装 JDK-7+

1. 下载并安装 JDK-7+ 安装包。
   - Linux/MacOS 系统：使用系统自带的包管理器安装
   - Windows 系统：点此下载
2. 按照以下示例进行安装检查。

```
$ java -version
java version "1.8.0_77"
Java(TM) SE Runtime Environment (build 1.8.0_77-b03)
Java HotSpot(TM) 64-Bit Server VM (build 25.77-b03, mixed mode)
```

安装并启动 Hadoop 环境

1. 下载 2.6.0 版本以上的 Hadoop 安装包。（点此下载）
2. 解压并安装，根据实际集群情况安装 Hadoop 服务。
3. 按照如下示例启动 Hadoop 环境。

```
$ bin/start-all.sh
# 检查服务是否成功启动
$ jps
24017 NameNode
24835 Jps
24131 DataNode
24438 ResourceManager
5114 HMaster
24287 SecondaryNameNode
24527 NodeManager
```

4. 在 `/etc/profile` 中添加 Hadoop 路径，并执行 `source /etc/profile` 的命令使配置生效。

```
export HADOOP_HOME=/data/hadoop/hadoop-2.6.0
```

```
export PATH=$PATH:$HADOOP_HOME/bin
```

下载及安装 Hive 环境

1. 下载类型为 bin.tar.gz 的 Hive 安装包。（点此下载）
2. 按照如下示例解压安装包。

```
$ mkdir /home/admin/hive-2.1.0
$ tar -zxvf apache-hive-2.1.0-bin.tar.gz -C /home/admin/
$ mv /home/admin/apache-hive-2.1.0-bin /home/admin/hive-2.1.0/
```

3. 按照如下示例初始化 schema。

```
    # 进入指定的目录
    $ cd /home/admin/hive-2.1.0/

    # 初始化，如果是mysql则derby可以直接替换成mysql
    # 如果执行出错可以删除rm -rf metastore_db/之后重新执行
    $ ./bin/schematool -initSchema -dbType derby
```

4. 按照如下示例启动 Hive 环境。

```
$ ./bin/hive
# 检查服务是否成功启动
hive> show databases;
OK
default
Time taken: 0.207 seconds, Fetched: 1 row(s)
```

下载表格存储的 Java SDK

1. 在 Maven 库中下载 4.1.0 版本以上的 Java SDK 相关依赖包。（点此下载）

说明:

该依赖包会随最新的 Java SDK 发布，请根据最新的 Java SDK 版本下载相关依赖包。

2. 按照如下示例将 SDK 拷贝到 Hive 目录下。

```
$ mv tablestore-4.1.0-jar-with-dependencies.jar /home/admin/hive-2.1
.0/
```

下载阿里云 EMR SDK

点此下载 EMR SDK 依赖包。

说明:

了解更多 EMR 信息请参考这里。

5.2 使用教程

本文主要为您介绍如何使用 Hive/HadoopMR 来访问表格存储中的表。

数据准备

在表格存储中准备一张数据表 pet，`name` 是唯一的一列主键，数据示例如下：

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	
Puffball	Diane	hamster	f	1999-03-30	

说明:

表格中空白的部分不需要写入，因为表格存储是一个 schema-free 的存储结构（#unique_41），没有值也不需要写入`NULL`。

Hive 访问示例

前提条件

按照准备工作准备好 Hadoop、Hive、JDK 环境以及表格存储 JAVA SDK 和 EMR SDK 依赖包。

示例

```
# HADOOP_HOME 及 HADOOP_CLASSPATH 可以添加到 /etc/profile 中
$ export HADOOP_HOME=${你的 Hadoop 安装目录}
$ export HADOOP_CLASSPATH=emr-tablestore-1.4.2.jar:tablestore-4.3.1-
jar-with-dependencies.jar:joda-time-2.9.4.jar
$ bin/hive
hive> CREATE EXTERNAL TABLE pet
  (name STRING, owner STRING, species STRING, sex STRING, birth STRING
, death STRING)
  STORED BY 'com.aliyun.openservices.tablestore.hive.TableStore
StorageHandler'
  WITH SERDEPROPERTIES(
    "tablestore.columns.mapping"="name,owner,species,sex,birth,death")
  TBLPROPERTIES (
    "tablestore.endpoint"="YourEndpoint",
```

```
    "tablestore.access_key_id"="YourAccessKeyId",
    "tablestore.access_key_secret"="YourAccessKeySecret",
    "tablestore.table.name"="pet");
hive> SELECT * FROM pet;
Bowser  Diane   dog     m       1979-08-31      1995-07-29
Buffy   Harold  dog     f       1989-05-13      NULL
Chirpy  Gwen    bird    f       1998-09-11      NULL
Claws   Gwen    cat     m       1994-03-17      NULL
Fang    Benny   dog     m       1990-08-27      NULL
Fluffy  Harold  cat     f       1993-02-04      NULL
Puffball        Diane   hamster f       1999-03-30      NULL
Slim    Benny   snake   m       1996-04-29      NULL
Whistler        Gwen    bird    NULL    1997-12-09      NULL
Time taken: 5.045 seconds, Fetched 9 row(s)
hive> SELECT * FROM pet WHERE birth > "1995-01-01";
Chirpy  Gwen    bird    f       1998-09-11      NULL
Puffball        Diane   hamster f       1999-03-30      NULL
Slim    Benny   snake   m       1996-04-29      NULL
Whistler        Gwen    bird    NULL    1997-12-09      NULL
Time taken: 1.41 seconds, Fetched 4 row(s)
```

参数说明如下：

- WITH SERDEPROPERTIES

  tablestore.columns.mapping（可选）：在默认的情况下，外表的字段名即为表格存储上表的列名（主键列名或属性列名）。但有时外表的字段名和表上列名并不一致（比如处理大小写或字符集相关的问题），这时候就需要指定 tablestore.columns.mapping。该参数为一个英文逗号分隔的字符串，每一项都是表上列名，顺序与外表字段一致。

  > 说明：
  > 空白也会被认为是表上列名的一部分。

- TBLPROPERTIES
  - tablestore.endpoint（必填）：访问表格存储的服务地址，也可以在表格存储控制台上查看这个实例的 endpoint 信息。
  - tablestore.instance（可选）：表格存储的实例名。若不填，则为 tablestore.endpoint 的第一段。
  - tablestore.table.name（必填）：表格存储上对应的表名。
  - tablestore.access_key_id、tablestore.access_key_secret（必填），请参见访问控制。
  - tablestore.sts_token（可选），请参见授权管理。

HadoopMR 访问示例

以下示例介绍如何使用 HadoopMR 程序统计数据表 pet 的行数。

示例代码

- 构建 Mappers 和 Reducers

```
public class RowCounter {
public static class RowCounterMapper
extends Mapper<PrimaryKeyWritable, RowWritable, Text, LongWritable>
 {
    private final static Text agg = new Text("TOTAL");
    private final static LongWritable one = new LongWritable(1);

    @Override
    public void map(
        PrimaryKeyWritable key, RowWritable value, Context context)
        throws IOException, InterruptedException {
        context.write(agg, one);
    }
}

public static class IntSumReducer
extends Reducer<Text,LongWritable,Text,LongWritable> {

    @Override
    public void reduce(
        Text key, Iterable<LongWritable> values, Context context)
        throws IOException, InterruptedException {
        long sum = 0;
        for (LongWritable val : values) {
            sum += val.get();
        }
        context.write(key, new LongWritable(sum));
    }
}
}
```

数据源每从表格存储上读出一行，都会调用一次 mapper 的 map()。前两个参数 PrimaryKeyWritable 和 RowWritable 分别对应这行的主键以及这行的内容。可以通过调用 PrimaryKeyWritable.getPrimaryKey() 和 RowWritable.getRow() 取得表格存储 JAVA SDK 定义的主键对象及行对象。

- 配置表格存储作为 mapper 的数据源。

```
    private static RangeRowQueryCriteria fetchCriteria() {
        RangeRowQueryCriteria res = new     RangeRowQueryCriteria("
YourTableName");
        res.setMaxVersions(1);
        List<PrimaryKeyColumn> lower = new ArrayList<PrimaryKey
Column>();
        List<PrimaryKeyColumn> upper = new ArrayList<PrimaryKey
Column>();
        lower.add(new PrimaryKeyColumn("YourPkeyName", PrimaryKey
Value.INF_MIN));
        upper.add(new PrimaryKeyColumn("YourPkeyName", PrimaryKey
Value.INF_MAX));
        res.setInclusiveStartPrimaryKey(new PrimaryKey(lower));
        res.setExclusiveEndPrimaryKey(new PrimaryKey(upper));
        return res;
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
```

```
        Job job = Job.getInstance(conf, "row count");
        job.addFileToClassPath(new Path("hadoop-connector.jar"));
        job.setJarByClass(RowCounter.class);
        job.setMapperClass(RowCounterMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(LongWritable.class);
        job.setInputFormatClass(TableStoreInputFormat.class);
        TableStoreInputFormat.setEndpoint(job, "https://YourInstance
.Region.ots.aliyuncs.com/");
        TableStoreInputFormat.setCredential(job, "YourAccessKeyId",
 "YourAccessKeySecret");
        TableStoreInputFormat.addCriteria(job, fetchCriteria());
        FileOutputFormat.setOutputPath(job, new Path("output"));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
```

示例代码中使用 job.setInputFormatClass(TableStoreInputFormat.class) 把表格存储设为数据源，除此之外，还需要：

- 把 hadoop-connector.jar 部署到集群上并添加到 classpath 里面。路径为 addFileToClassPath() 指定 hadoop-connector.jar 的本地路径。代码中假定 hadoop-connector.jar 在当前路径。
- 访问表格存储需要指定入口和身份。通过 TableStoreInputFormat.setEndpoint() 和 TableStoreInputFormat.setCredential() 设置访问表格存储需要指定的 endpoint 和 access key 信息。
- 指定一张表用来计数。

说明：

- 每调用一次 addCriteria() 可以在数据源里添加一个 JAVA SDK 定义的 RangeRowQueryCriteria 对象。可以多次调用addCriteria()。RangeRowQueryCriteria 对象与表格存储 JAVA SDK GetRange 接口所用的 RangeRowQueryCriteria 对象具有相同的限制条件。
- 可以利用 RangeRowQueryCriteria 的 setFilter() 和 addColumnsToGet() 在表格存储的服务器端过滤掉不必要的行和列，减少访问数据的大小，降低成本，提高性能。
- 通过添加对应多张表的多个 RangeRowQueryCriteria，可以实现多表的 union。
- 通过添加同一张表的多个 RangeRowQueryCriteria，可以做到更均匀的切分。TableStore-Hadoop Connector 会根据一些策略将用户传入的范围切细。

程序运行示例

```
$ HADOOP_CLASSPATH=hadoop-connector.jar bin/hadoop jar row-counter.jar
...
$ find output -type f
output/_SUCCESS
```

```
output/part-r-00000
output/._SUCCESS.crc
output/.part-r-00000.crc
$ cat out/part-r-00000
TOTAL   9
```

类型转换说明

表格存储支持的数据类型和 Hive/Spark 支持的数据类型不完全相同。

下表列出了从表格存储的数据类型（行）转换到 Hive/Spark 数据类型（列）的支持情况。

	TINYIN	SMALLI	INT	BIGINT	FLOAT	DOUBLI	BOOLEA	STRING	BINARY
INTEGER	可，损失精度	可，损失精度	可，损失精度	可	可，损失精度	可，损失精度			
DOUBLE	可，损失精度	可，损失精度	可，损失精度	可，损失精度	可，损失精度	可			
BOOLEAN							可		
STRING								可	
BINARY									可

6 Spark/SparkSQL

6.1 环境准备

本文主要为您介绍使用 Saprk/Spark SQL 来查询和链接表格存储中的表需要的环境准备。

使用 Saprk/Spark SQL 来查询和链接表格存储中的表

通过表格存储及 E-MapReduce 官方团队发布的依赖包，可以直接使用 Spark 及 Spark SQL 来访问表格存储中的数据并进行数据的查询分析。

下载及安装 Spark/Spark SQL

1. 下载版本号为 1.6.2 的 Spark 安装包，安装包类型为 Pre-built for Hadoop 2.6。（点此下载）
2. 按照如下示例解压安装包。

```
$ cd /home/admin/spark-1.6.2
$ tar -zxvf spark-1.6.2-bin-hadoop2.6.tgz
```

安装 JDK-7+

1. 下载并安装 JDK-7+ 安装包。
   - Linux/MacOS 系统：请用系统自带的包管理器进行安装
   - Windows 系统：点此下载
2. 按照如下示例进行安装检查。

```
$ java -version
java version "1.8.0_77"
Java(TM) SE Runtime Environment (build 1.8.0_77-b03)
Java HotSpot(TM) 64-Bit Server VM (build 25.77-b03, mixed mode)
```

下载表格存储的 Java SDK

1. 在 Maven 库中下载 4.1.0 版本以上的 Java SDK 相关依赖包。（点此下载）

> **说明:**
> 该依赖包会随最新的 Java SDK 发布，请根据最新的 Java SDK 版本下载相关依赖包。

2. 按照如下示例将 SDK 拷贝到 Spark 目录下。

```
$ mv tablestore-4.1.0-jar-with-dependencies.jar /home/admin/spark-1.
6.2/
```

下载阿里云 EMR SDK

下载 EMR SDK 相关的依赖包。（点此下载）

说明:

了解更多 EMR 信息请参见这里。

启动 Spark SQL

```
$ cd /home/admin/spark-1.6.2/
$ bin/spark-sql --master local --jars tablestore-4.3.1-jar-with-
dependencies.jar,emr-tablestore-1.4.2.jar
```

6.2 使用教程

本文主要为您介绍如何使用 Saprk/Spark SQL 来查询和链接表格存储中的表。

数据准备

在表格存储中准备一张数据表 pet，其中`name`是唯一的一列主键。数据示例如下：

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	
Puffball	Diane	hamster	f	1999-03-30	

说明:

表格中空白的部分不需要写入，因为表格存储是一个 schema-free 的存储结构（数据模型），没有值也不需要写入`NULL`。

Spark SQL 访问示例

前提条件

按照准备工作中的步骤准备好 Spark、JDK 环境以及表格存储 Java SDK 和 EMR SDK 的依赖包。

示例

```
$ bin/spark-sql --master local --jars tablestore-4.3.1-jar-with-
dependencies.jar,emr-tablestore-1.4.2.jar
spark-sql> CREATE EXTERNAL TABLE pet
  (name STRING, owner STRING, species STRING, sex STRING, birth STRING
, death STRING)
  STORED BY 'com.aliyun.openservices.tablestore.hive.TableStore
StorageHandler'
  WITH SERDEPROPERTIES(
    "tablestore.columns.mapping"="name,owner,species,sex,birth,death")
  TBLPROPERTIES (
    "tablestore.endpoint"="YourEndpoint",
    "tablestore.access_key_id"="YourAccessKeyId",
    "tablestore.access_key_secret"="YourAccessKeySecret",
    "tablestore.table.name"="pet");
spark-sql> SELECT * FROM pet;
Bowser  Diane   dog     m       1979-08-31      1995-07-29
Buffy   Harold  dog     f       1989-05-13      NULL
Chirpy  Gwen    bird    f       1998-09-11      NULL
Claws   Gwen    cat     m       1994-03-17      NULL
Fang    Benny   dog     m       1990-08-27      NULL
Fluffy  Harold  cat     f       1993-02-04      NULL
Puffball        Diane   hamster f       1999-03-30      NULL
Slim    Benny   snake   m       1996-04-29      NULL
Whistler        Gwen    bird    NULL    1997-12-09      NULL
Time taken: 5.045 seconds, Fetched 9 row(s)
spark-sql> SELECT * FROM pet WHERE birth > "1995-01-01";
Chirpy  Gwen    bird    f       1998-09-11      NULL
Puffball        Diane   hamster f       1999-03-30      NULL
Slim    Benny   snake   m       1996-04-29      NULL
Whistler        Gwen    bird    NULL    1997-12-09      NULL
Time taken: 1.41 seconds, Fetched 4 row(s)
```

参数说明如下：

- WITH SERDEPROPERTIES
  - tablestore.columns.mapping（可选）：在默认情况下，外表的字段名即为表格存储上表的列名（主键列名或属性列名）。但有时外表的字段名和表上列名并不一致（比如处理大小写或字符集相关的问题），这时候就需要指定 tablestore.columns.mapping。该参数为一个英文逗号分隔的字符串，每个分隔之间不能添加空格，每一项都是表上列名，顺序与外表字段一致。

> **说明：**
> 表格存储的列名支持空白字符，所以空白也会被认为是表上列名的一部分。

- TBLPROPERTIES
  - tablestore.endpoint（必填）：访问表格存储的服务地址，也可以在表格存储控制台上查看这个实例的 endpoint 信息。
  - tablestore.instance（可选）：表格存储的实例名。若不填，则为 tablestore.endpoint 的第一段。
  - tablestore.table.name（必填）：表格存储上对应的表名。
  - tablestore.access_key_id、tablestore.access_key_secret（必填），请参见访问控制。
  - tablestore.sts_token（可选），请参见授权管理。

Spark 访问示例

以下示例介绍如何使用 Spark 程序统计数据表 pet 的行数。

```
private static RangeRowQueryCriteria fetchCriteria() {
    RangeRowQueryCriteria res = new RangeRowQueryCriteria("YourTableN
ame");
    res.setMaxVersions(1);
    List<PrimaryKeyColumn> lower = new ArrayList<PrimaryKeyColumn>();
    List<PrimaryKeyColumn> upper = new ArrayList<PrimaryKeyColumn>();
    lower.add(new PrimaryKeyColumn("YourPkeyName", PrimaryKeyValue.
INF_MIN));
    upper.add(new PrimaryKeyColumn("YourPkeyName", PrimaryKeyValue.
INF_MAX));
    res.setInclusiveStartPrimaryKey(new PrimaryKey(lower));
    res.setExclusiveEndPrimaryKey(new PrimaryKey(upper));
    return res;
}

public static void main(String[] args) {
    SparkConf sparkConf = new SparkConf().setAppName("RowCounter");
    JavaSparkContext sc = new JavaSparkContext(sparkConf);

    Configuration hadoopConf = new Configuration();
    TableStoreInputFormat.setCredential(
        hadoopConf,
        new Credential("YourAccessKeyId", "YourAccessKeySecret"));
    TableStoreInputFormat.setEndpoint(
        hadoopConf,
        new Endpoint("https://YourInstance.Region.ots.aliyuncs.com
/"));
    TableStoreInputFormat.addCriteria(hadoopConf, fetchCriteria());

    try {
        JavaPairRDD<PrimaryKeyWritable, RowWritable> rdd = sc.
newAPIHadoopRDD(
            hadoopConf,
            TableStoreInputFormat.class,
            PrimaryKeyWritable.class,
            RowWritable.class);
        System.out.println(
            new Formatter().format("TOTAL: %d", rdd.count()).toString
());
    } finally {
        sc.close();
```

```
    }
}
```

说明:

如果使用 scala，只需把 JavaSparkContext 换成 SparkContext，JavaPairRDD 换成 PairRDD 即可。或者更简单，交给编译器自行做类型推断

运行程序

```
$ bin/spark-submit --master local --jars hadoop-connector.jar row-
counter.jar
TOTAL: 9
```

类型转换说明

表格存储支持的数据类型和 Hive/Spark 支持的数据类型不完全相同。

下表列出了从表格存储的数据类型（行）转换到 Hive/Spark 数据类型（列）时所支持的情况。

	TINYIN	SMALLI	INT	BIGINT	FLOAT	DOUBLI	BOOLEA	STRING	BINARY
INTEGER	可，损失精度	可，损失精度	可，损失精度	可	可，损失精度	可，损失精度			
DOUBLE	可，损失精度	可，损失精度	可，损失精度	可，损失精度	可，损失精度	可			
BOOLEAN							可		
STRING								可	
BINARY									可