Alibaba Cloud Table Store

計算與分析

檔案版本：20180929

目錄

1 函數觸發器

1.1 使用Function Compute

Function Compute（Function Compute）是一個事件驅動的服務，通過Function Compute，使用者無需管理伺服器等運行情況，只需編寫代碼並上傳。

Function Compute準備計算資源，並以Auto Scaling的方式運行使用者代碼，而使用者只需根據實際代碼運行所消耗的資源進行付費。詳細資料請參見*Function Compute####*。

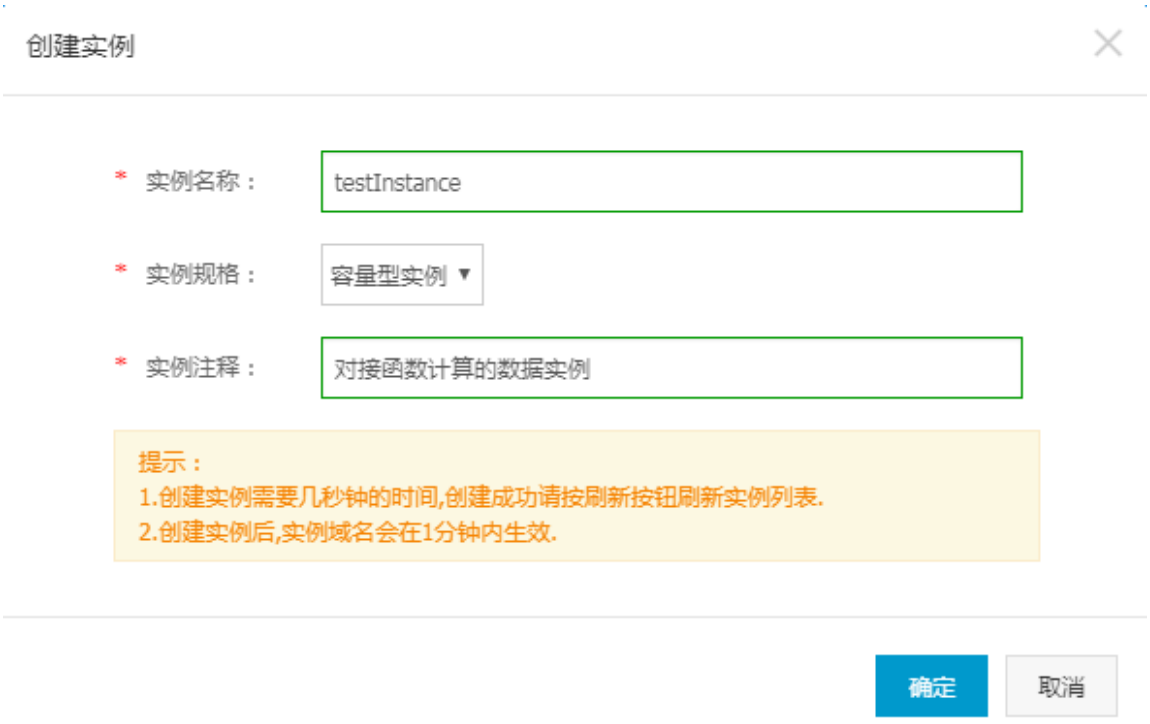
Table Store Stream是用於擷取Table Store表中增量資料的一個資料通道，通過建立Table Store觸發器，能夠實現Table Store Stream和Function Compute的自動對接，讓計算函數中自訂的程式邏輯自動處理Table Store表中發生的資料修改。詳細資料請參見*Table Store Stream*。

本節介紹如何使用Function Compute對錶格儲存增量資料進行即時計算。

配置Table Store觸發器

您可以通過控制台建立Table Store觸發器來處理Table Store資料表的即時資料流。

1. 建立開通了Stream流的資料表。
   a. 在*Table Store###*建立執行個體。

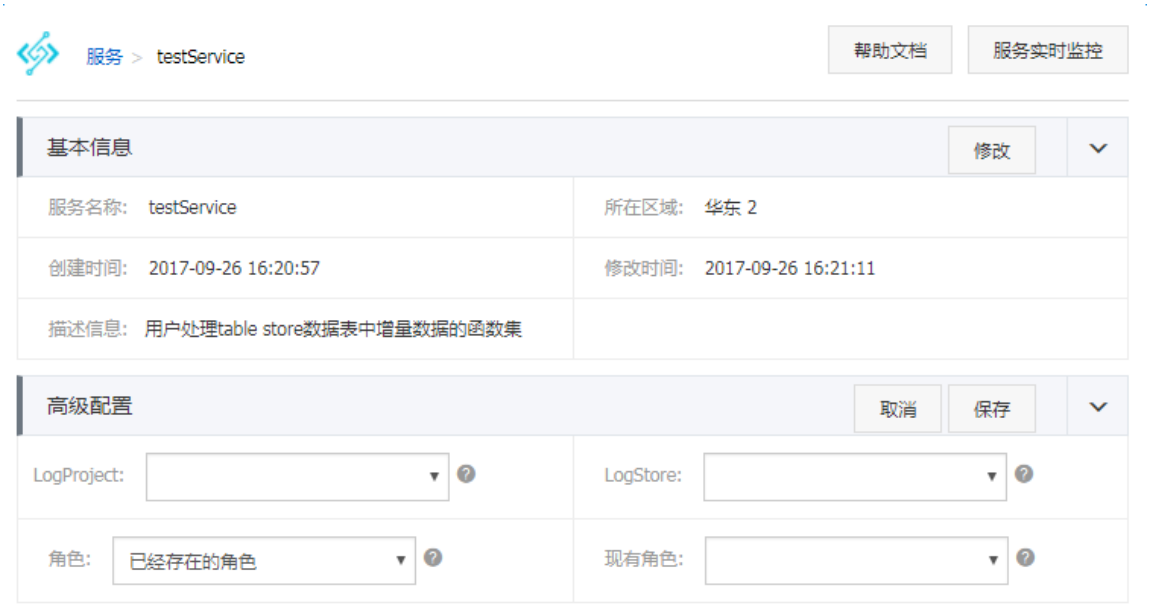


   b. 在該執行個體下建立資料表，並且開啟資料表的Stream功能。

2. 建立Function Compute的函數。

   a. 在*Function Compute###*建立服務（Service）。

b. 在Service的進階配置中，您可以佈建服務的角色，用於授權函數進行日誌收集，以及在計算函數中繼續訪問使用者的其他資源。詳細資料請參見*Function Compute#####*。



c. 在新建立的Service下單擊建立函數。

d. 在函數模板頁面選擇 空白函數。

e. 在觸發器配置頁面選擇不建立任何觸發器 ，然後單擊下一步。

f. 配置函數資訊。

Table Store觸發器會使用*CBOR##*將增量資料編碼為Function Compute的Event，並調用使用者函數。下圖樣本函數就是將編碼後的Event重新解碼和列印到日誌中心，您可以在解碼資料後進行任意資料處理。

3. 建立和測試Table Store觸發器。

   a. 在*Table Store###*新建立的資料表下，選擇使用已有**Function Compute**來建立觸發器。

b. 建立過程中需要授權Table Store發送事件通知所需的許可權。

勾選完畢後，實際可以在*RAM###*查看到自動建立的授權角色AliyunTableStoreStreamNotificationRole。

資料處理

- 資料格式

  Table Store觸發器使用*CBOR##*對增量資料進行編碼構成Function Compute的Event。增量資料的具體資料格式如下：

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

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

- 成員定義
  - Version
    - 含義: payload版本號碼，目前為Sync-v1
    - 類型: string
  - Records
    - 含義: 資料表中的增量資料行數組
    - 內部成員:
      - Type
        - 含義: 資料行類型，包含PutRow、UpdateRow和DeleteRow
        - 類型: string
      - Info
        - 含義: 資料行基本資料
        - 內部成員:
          - Timestamp
          - 含義: 該行的最後修改UTC時間
          - 類型: int64
      - PrimaryKey
        - 含義: 主鍵列數組
        - 內部成員
          - ColumnName
            - 含義: 主鍵列名稱
            - 類型: string
          - Value

        - 含義: 主鍵列內容
        - 類型: formated_value，支援integer、string和blob
    - Colunms
        - 含義: 屬性列數組
        - 內部成員
            - Type
                - 含義: 屬性列類型，包含Put、DeleteOneVersion和DeleteAllVersions
                - 類型: string
            - ColumnName
                - 含義: 屬性列名稱
                - 類型: string
            - Value
                - 含義: 屬性列內容
                - 類型: formated_value，支援integer、boolean、double、string和blob
            - Timestamp
                - 含義: 屬性列最後修改UTC時間
                - 類型: int64

- 資料樣本

```
{
    "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": [
                {
```

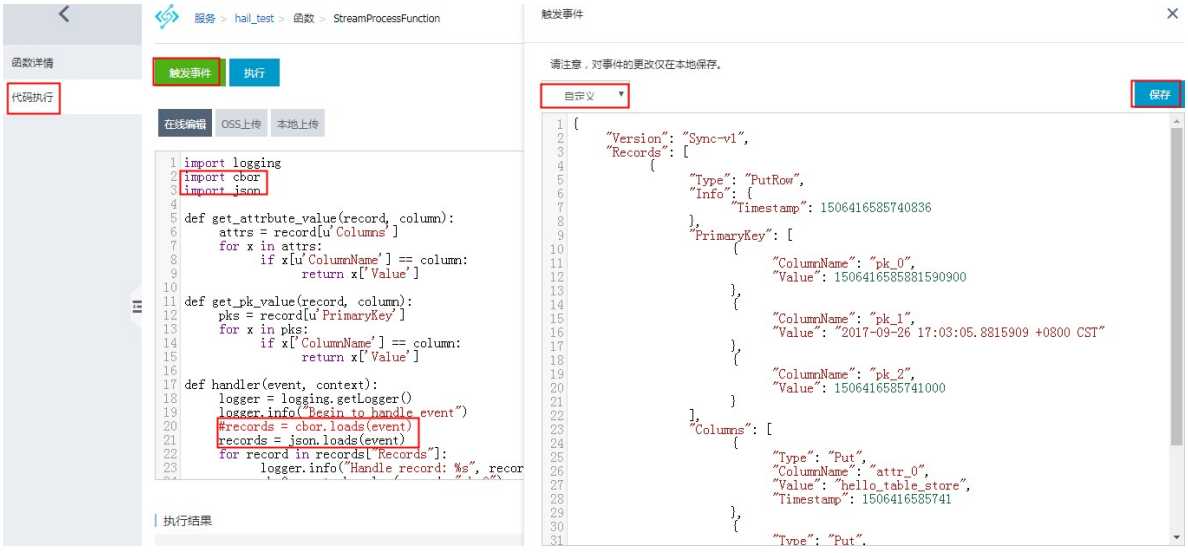
```
                    "Type": "Put",
                    "ColumnName": "attr_0",
                    "Value": "hello_table_store",
                    "Timestamp": 1506416585741
                },
                {
                    "Type": "Put",
                    "ColumnName": "attr_1",
                    "Value": 1506416585881590900,
                    "Timestamp": 1506416585741
                }
            ]
        }
    ]
}
```

線上調試

Function Compute支援函數的線上調試功能，使用者能夠自己構建觸發的Event，並測試代碼邏輯是否符合期望。

由於Table Store觸發函數服務的Event是CBOR格式，該資料格式是一種類似JSON的二進位格式，可以按照如下方法進行線上調試：

1. 在代碼中同時import cbor和json。
2. 單擊觸發事件，選擇自訂，將上述資料樣本的json檔案粘貼至編輯框，根據實際情況修改，並儲存。
3. 代碼中先使用 `records = json.loads(event)` 來處自訂的測試觸發事件，單擊執行，查看是否符合期望。
4. 當使用 `records=json.loads(event)` 測試OK之後，可以修改為 `records = cbor.loads(event)` 並儲存，當Table Store中有資料寫入時，相關的函數邏輯就會觸發。



範例程式碼：

```
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Function Compute做資料清洗

經過了上一節的介紹，下面我們來完成一個使用Function Compute對錶格儲存中的資料做簡單清洗的情境。

Table Store高並發的寫入效能以及低廉的儲存成本非常適合物聯網、日誌、監控資料的儲存，我們可以將資料寫入到Table Store中，同時在Function Compute中對新增的資料做簡單的清洗，並將清洗之後的資料寫回到Table Store的結果表中，並對未經處理資料及結果資料提供即時訪問。

資料定義

我們假設寫入的為日誌資料，包括三個基礎欄位：

欄位名稱	類型	含義
id	整型	日誌id
level	整型	日誌的等級，越大表明等級越高
message	字串	日誌的內容

我們需要將 level>1 的日誌寫入到另外一張資料表中，用作專門的查詢。

建立執行個體及資料表

在*Table Store###*建立Table Store執行個體（本次以 華東2 distribute-test 為例），並建立源表（source_data）及結果表（result），主鍵為 id (整型)，由於Table Store是 schemafree 結構，無需預先定義其他屬性欄欄位。

以 source_data 為例，建立如下圖：

開啟資料來源表的**Stream**功能

觸發器功能需要先開啟資料表的*Stream##*，才能在Function Compute中處理寫入Table Store中的增量資料。

Stream記錄到期時間長度是指通過 StreamAPI 能夠讀取到的增量資料的最長時間。

由於觸發器只能綁定現有的函數，故先到Function Compute的控制台上在同region建立服務及函數。

- 建立Function Compute服務

  在*Function Compute####*上建立服務及處理函數，我們繼續使用華東2節點。

  1. 在華東2節點建立服務。

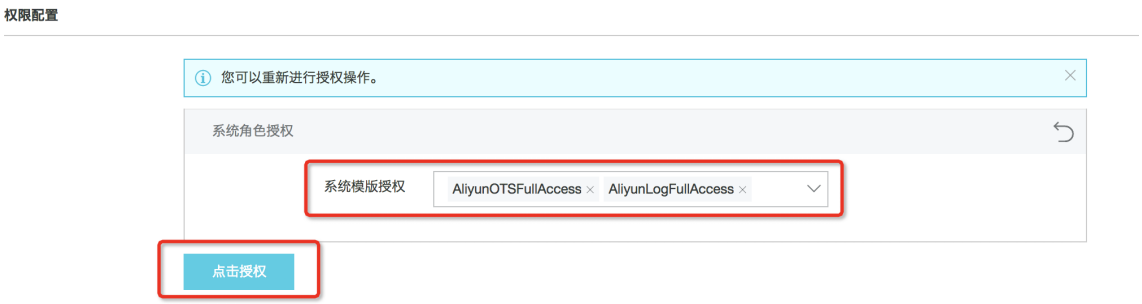
  2. 建立函數依次選擇空白函數 > 不建立觸發器。


* 所在服务
transform_test
新建服务
* 函数名称
etl_test
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
描述信息
对表格存储中的数据做简单的清洗
* 运行环境
python2.7
代码上传方式
在线编辑
OSS上传
代码包上传
文件夹上传

- 函數名稱為：etl_test，選擇 python2.7 環境，線上編輯代碼
- 函數入口為：etl_test.handler
- 代碼稍後編輯，點擊下一步。

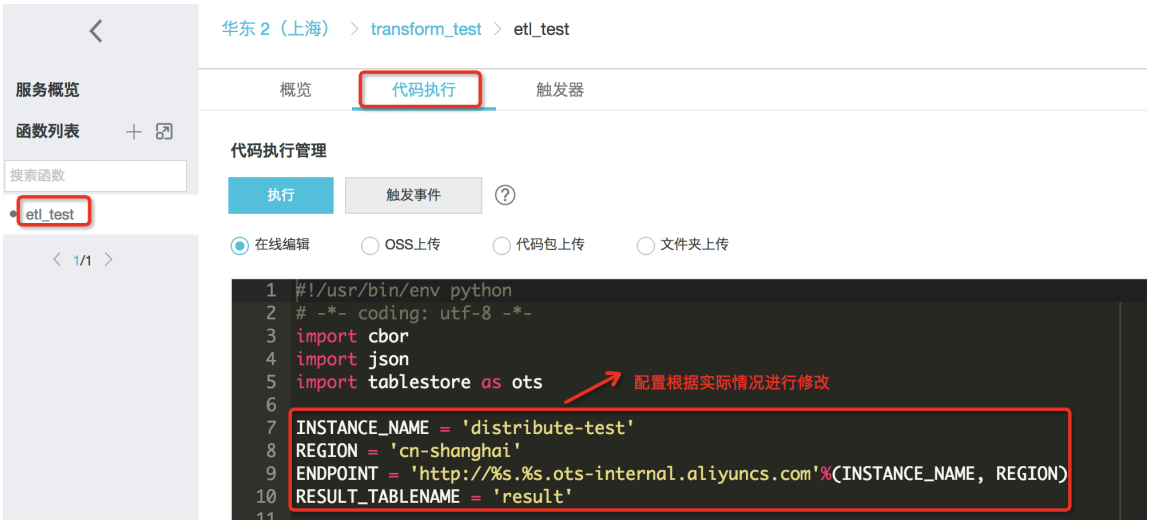
3. 進行服務授權

   由於Function Compute需要將運行中的日誌寫入到Log Service中，同時，需要對錶格儲存的表進行讀寫，故需要對Function Compute進行授權，為方便起見，我們先添加 AliyunOTSFullAccess 與 AliyunLogFullAccess 許可權，實際生產中，建議根據許可權最小原則來添加許可權。



4. 單擊授權完成，並建立函數。
5. 修改函數代碼。

   建立好函數之後，點擊對應的函數—代碼執行，編輯代碼並儲存，其中，INSTANCE_NAME(Table Store的執行個體名稱)、REGION(使用的地區)需要根據情況進行修改：



使用範例程式碼如下：

```
#!/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']
#由於已經授權了AliyunOTSFullAccess許可權，此處擷取的credentials具有訪問
Table Store的許可權
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."
```

綁定觸發器

1. 回到Table Store的執行個體管理頁面，單擊表 source_data 後的使用觸發器按鈕，進入觸發器綁定介面，單擊使用已有**Function Compute**， 選擇剛建立的服務及函數，勾選**Table Store**發送事件通知的許可權， 進行確定。

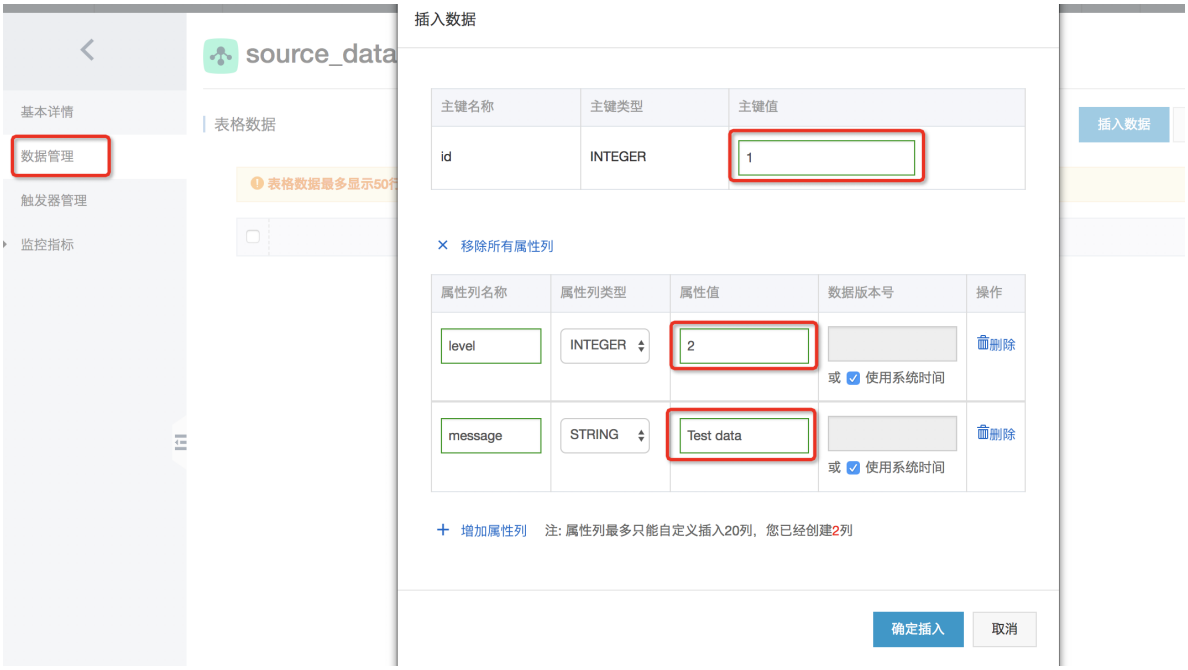

source_data
创建触发器
新建函数计算
使用已有
触发器列表
目前一张数据表只支持创建
函数计算服务名
创建触发
* 选择
* 选择

2. 綁定成功之後，能夠看到如下的資訊：

運行驗證

1. 向 source_data 表中寫入資料。

   單擊source_data的資料管理頁面，然後單擊插入資料，如圖依次填入id、level及message資訊。



2. 在 result 表中查詢清洗後的資料

   單擊result表的資料管理頁面，會查詢到剛寫入到 source_data 中的資料。

   向 soure_data 寫入level <=1的資料將不會同步到 result 表中。


result
基本详情
数据管理
触发器管理
监控指标
表格数据
插入数据
查询数据
更新数据
删除数据
表格数据最多显示50行。
id(主键)
level
message
1
2
Test data
版本
值
类型
1525319981646
2
INTEGER
共有1条，每页显示：10条
1

2 MaxCompute

2.1 使用MaxCompute訪問Table Store

背景資訊

本文介紹如何在同一個雲帳號下實現Table Store和 MaxCompute 之間的無縫串連。

*MaxCompute*是一項大資料計算服務，它能提供快速、完全託管的 PB 級資料倉儲解決方案，使您可以經濟並高效地分析處理海量資料。您只需通過一條簡單的 DDL 語句，即可在 MaxCompute 上建立一張外部表格，建立 MaxCompute 表與外部資料源的關聯，提供各種資料的接入和輸出能力。MaxCompute 表是結構化的資料，而外部表格可以不限於結構化資料。

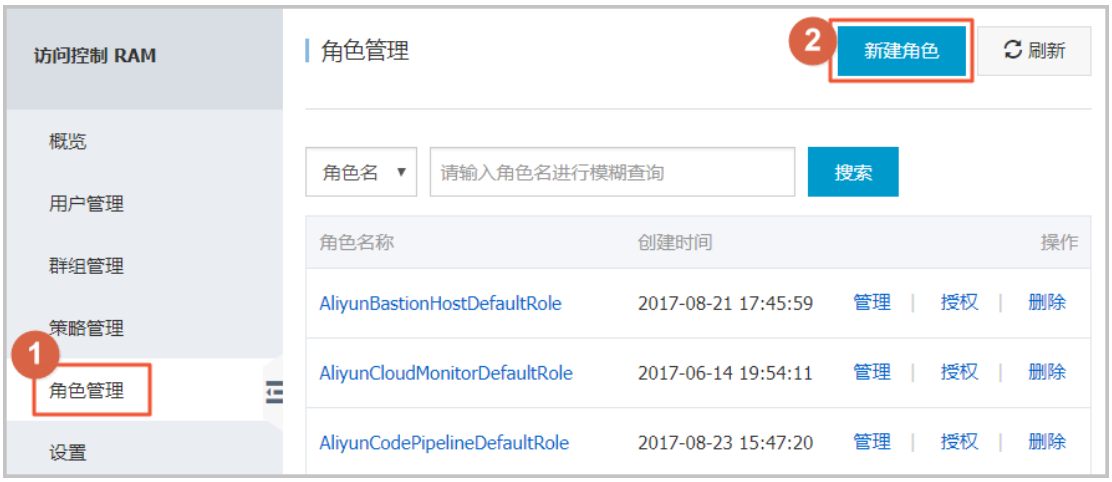
Table Store與 MaxCompute 都有其自身的類型系統，兩者之間的類型對應關係如下表所示。

Table Store	MaxCompute
STRING	STRING
INTEGER	BIGINT
DOUBLE	DOUBLE
BOOLEAN	BOOLEAN
BINARY	BINARY

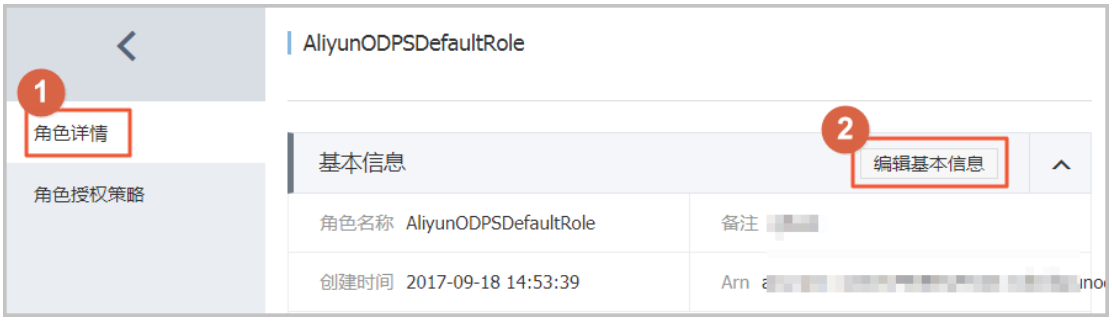
準備工作

使用 MaxCompute 訪問Table Store前，您需要完成以下準備工作：

1. 開通*MaxCompute##*。
2. 建立*AccessKey*。
3. 在 RAM 控制台授權 MaxCompute 訪問Table Store的許可權。
   - 方法一：登入阿里雲帳號##########。
   - 方法二：進行手動授權。步驟如下：
     1. 登入 *RAM ###*。
     2. 在角色管理頁面，建立使用者角色 AliyunODPSDefaultRole。



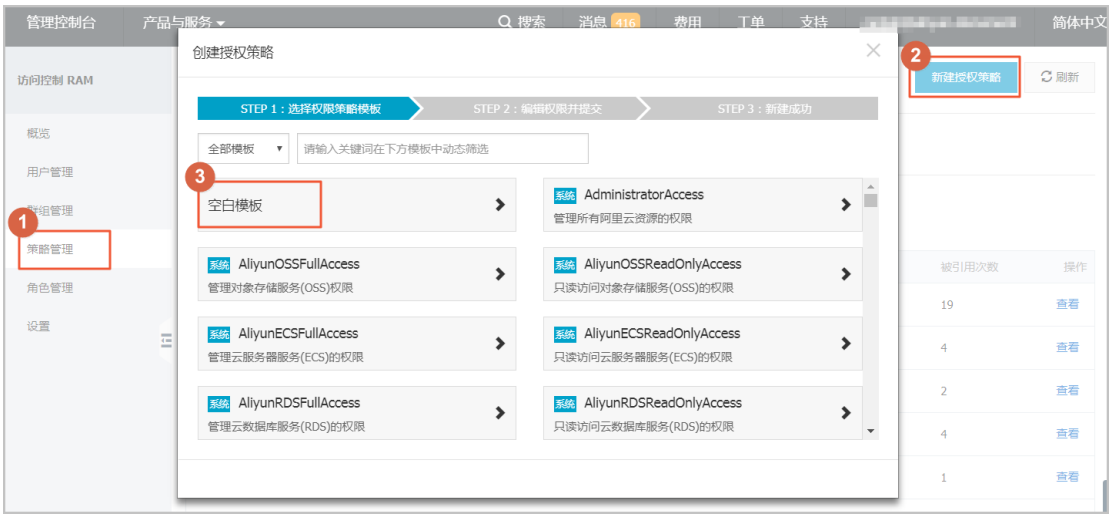
3. 在角色詳情頁面，設定策略內容。



策略內容如下：

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

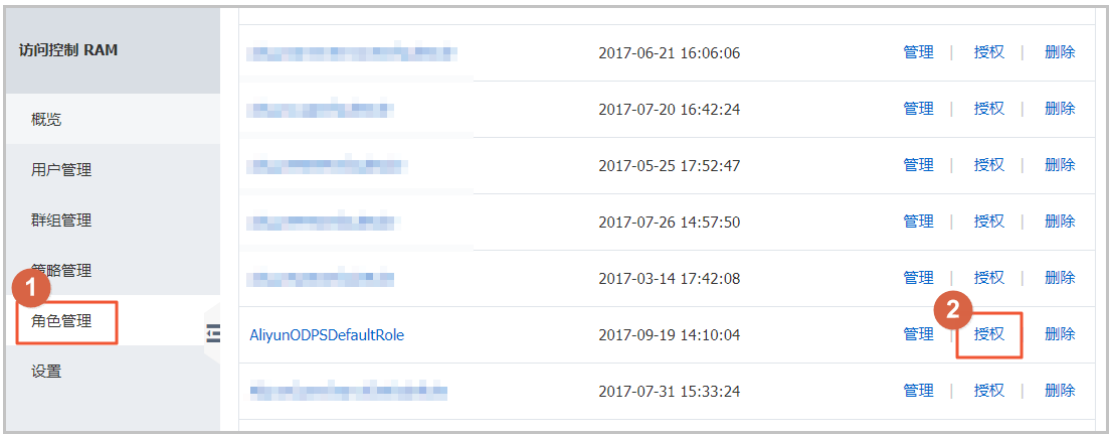
4. 在策略管理頁面，建立授權策略 AliyunODPSRolePolicy。



策略內容如下：

```
{
"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"
 }
]
}
--還可自訂其他許可權
```

5. 在角色管理頁面，將 AliyunODPSRolePolicy 許可權授權給 AliyunODPSDefaultRole 角色。



4. 在Table Store控制台######和#####。

   在本樣本中，建立的Table Store執行個體和資料表資訊如下：

   - 執行個體名稱：cap1
   - 資料表名稱：vehicle_track
   - 主鍵資訊：vid (integer)，gt (integer)
   - 訪問網域名稱：`https://cap1.cn-hangzhou.ots-internal.aliyuncs.com`

     > 说明：
     > 使用 MaxCompute 訪問Table Store時，建議使用Table Store的私網地址。

   - 設定執行個體網路類型為允許任意網路訪問。

步驟 1. 安裝並配置用戶端

1. 下載 *MaxCompute ###*並解壓。

> 说明：
> 確保您的機器上已安裝 JRE 1.7或以上版本。

2. 編輯 conf/odps_config.ini 檔案，對用戶端進行配置，如下所示：

```
access_id=********************
access_key=**********************
 # Accesss ID 及 Access Key 是使用者的雲帳號資訊，可登入阿里雲官網，進入管理主控台 — accesskeys 頁面進行查看。
project_name=my_project
 # 指定使用者想進入的項目空間。
end_point=https://service.odps.aliyun.com/api
 # MaxCompute 服務的訪問連結
tunnel_endpoint=https://dt.odps.aliyun.com
 # MaxCompute Tunnel 服務的訪問連結
log_view_host=http://logview.odps.aliyun.com
 # 當使用者執行一個作業後，用戶端會返回該作業的 LogView 地址。開啟該地址將會看到作業執行的詳細資料
https_check=true
 #決定是否開啟 HTTPS 訪問
```

> 说明：
> odps_config.ini 檔案中使用#作為注釋，MaxCompute 用戶端內使用--作為注釋。

3. 運行 bin/odpscmd.bat，輸入 `show tables;`。

   如果顯示當前 MaxCompute 項目中的表，則表述上述配置正確。

```
odps@ MCOTStest>show tables;

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

OK
```

步驟 2. 建立外部表格

建立一張 MaxCompute 的資料表（ots_vehicle_track）關聯到 Table Store 的某一張表（vehicle_track）。

關聯的資料表資訊如下：

- 執行個體名稱：cap1
- 資料表名稱：vehicle_track
- 主鍵資訊：vid (int); gt (int)

- 訪問網域名稱：`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)
```

參數說明如下：

標號	參數	說明
(1)	com.aliyun.odps.TableStore StorageHandler	MaxCompute 內建的處理 Table Store 資料的 StorageHandler，定義了 MaxCompute 和 Table Store 的互動，相關邏輯由 MaxCompute 實現。
(2)	SERDEPROPERITES	可以理解為提供參數選項的介面，在使用 TableStore StorageHandler 時，有兩個必須指定的選項，分別是 tablestore.columns.mapping 和 tablestore.table.name。
(3)	tablestore.columns.mapping	必填選項。MaxCompute 將要訪問的 Table Store 表的列，包括主鍵和屬性列。 其中，帶 : 的表示 Table Store 主鍵，例如本樣本中的 :vid 與 :gt，其他均為屬性列。在指定映射的時候，使用者必須提供指定 Table Store 表的所有主鍵，屬性列無需全部提供，可以只提供需要通過 MaxCompute 來訪問的屬性列。

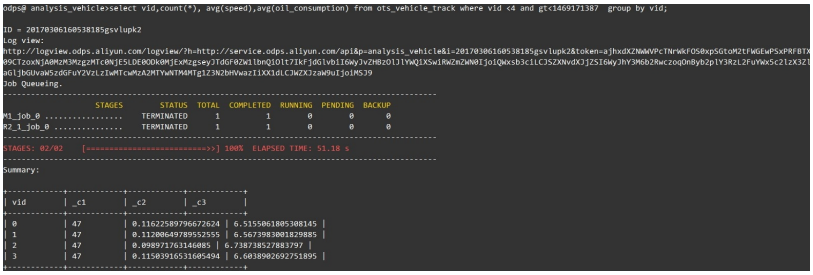
標號	參數	說明
（4）	tablestore.table.name	需要訪問的 Table Store 表名。如果指定的 Table Store 表名錯誤（不存在），則會報錯。MaxCompute 不會主動建立 Table Store 表。
（5）	LOCATION	指定訪問的 Table Store 的執行個體資訊，包括執行個體名和 endpoint 等。

步驟 3. 通過外部表格訪問 **Table Store** 資料

建立外部表格後，Table Store 的資料便引入到了 MaxCompute 生態中，您可通過 MaxCompute SQL 命令來訪問 Table Store 資料。

```
// 統計編號 4 以下的車輛在時間戳記 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;
```

返回類似如下結果：



2.2 跨帳號授權

本文檔介紹不同帳號之間如何?Table Store和 MaxCompute 之間的無縫串連。如需瞭解同帳號下的 Table Store與 Maxcompute 對接操作，請參考*###### MaxCompute ##Table Store*。

準備工作

跨雲帳號需要兩個主帳號，帳號 A 將存取權限授予帳號 B，則運行 MaxCompute 時，帳號 B 可以訪問帳號 A 下的表資料。基本資料如下：

說明：
以下資訊僅為樣本，在操作時請替換為實際使用的資訊。

項目	**Table Store**	**MaxCompute**
主帳號名	帳號 A	帳號 B

項目	Table Store	MaxCompute
UserId	12345	56789

使用 MaxCompute 跨雲帳號訪問Table Store前，您需要完成以下準備工作：

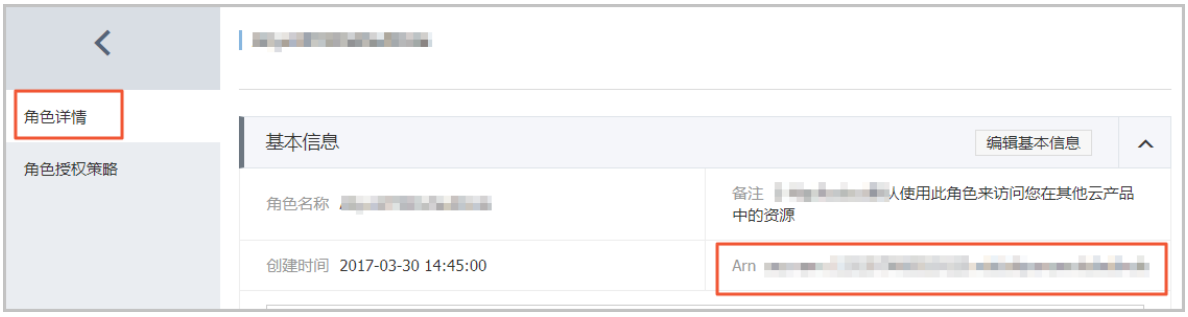
1. 帳號 B 開通*MaxCompute ##*，並建立*MaxCompute ##*。
2. 帳號 A 和 B 分別 *## AccessKey*。
3. 使用帳號 A 登入 *RAM ###*，並在角色管理頁面，建立使用者角色。

   在本樣本中，假設建立的角色名稱為 AliyunODPSRoleForOtherUser。
4. 進入該角色，在角色詳情頁面單擊編輯基本資料，設定策略內容。 策略內容設定如下：

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

> 说明：
> 請將上述策略內容中的 1xxxx 替換成您的 UID 即可。

5. 角色建立後，您可以在角色詳情頁面查看該角色的 roleArn：



6. 返回 RAM 控制台首頁，進入策略管理頁面，建立授權策略。

   在本樣本中，假設授權策略名稱為AliyunODPSRolePolicyForOtherUser。


管理控制台
产品与服务
访问控制 RAM
概览
用户管理
群组管理
1
策略管理
角色管理
设置
创建授权策略
STEP 1：选择权限策略模板
全部模板
请输入关键词在
3
空白模板
系统 AliyunOSSFullAccess
管理对象存储服务(OSS)权限
系统 AliyunECSFullAccess
管理云服务器服务(ECS)的权限
系统 AliyunRDSFullAccess
管理云数据库服务(RDS)的权限

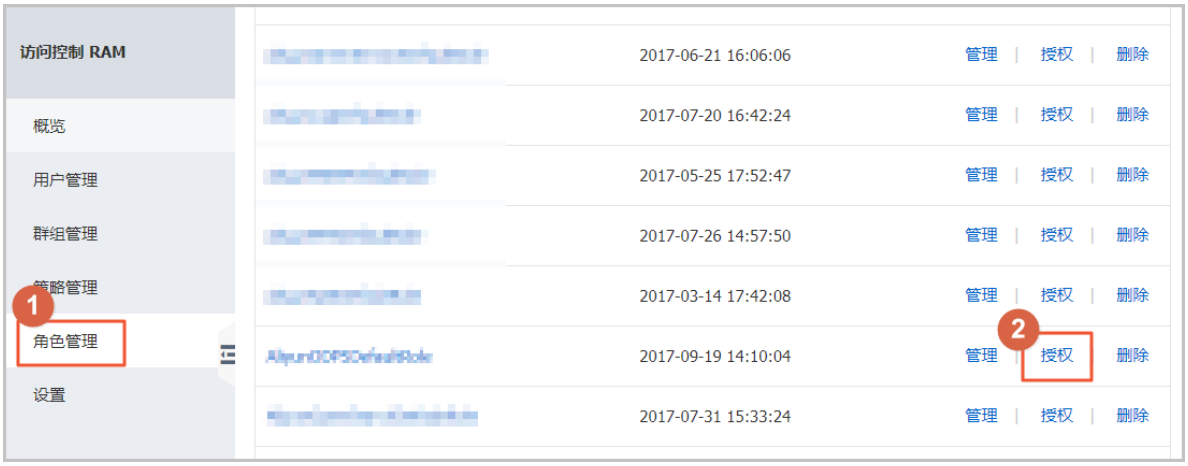
策略內容如下：

```
{
"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"
 }
]
}
```

说明：

您也可以自訂其他許可權，如 CreateTable 等。

7. 在角色管理頁面，將 AliyunODPSRolePolicyForOtherUser 許可權授權給 AliyunODPSRoleForOtherUser 角色。



8. 在Table Store控制台######和#####。

在本樣本中，建立的Table Store執行個體和資料表資訊如下：

- 執行個體名稱：cap1
- 資料表名稱：vehicle_track
- 主鍵資訊：vid (integer)，gt (integer)

- 訪問網域名稱：`https://cap1.cn-hangzhou.ots-internal.aliyuncs.com`

  说明：

  使用 MaxCompute 訪問Table Store時，建議使用Table Store的私網地址。

- 設定執行個體網路類型為允許任意網路訪問。

使用 **MaxCompute** 訪問**Table Store**

跨帳號訪問的操作與同帳號下的訪問一樣，只是在建立外部表格時使用 roleArn。

帳號 B 通過 MaxCompute 建立外部表格，指定####中建立出來的 roleArn 來訪問Table Store。

具體操作步驟請參考#######。其中，在步驟 2 建立外部表格時，使用如下代碼：

```
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'
```

2.3 使用AccessKey訪問Table Store

除了授權方式外，您還可以在 MaxCompute 中使用 AccessKey 訪問Table Store的資料。

準備工作

擷取Table Store資源所屬帳號的*AccessKeyId # AccessKeySecret*，如果該 AK 是資源所屬帳號的子帳號，那麼該子帳號至少需要對錶格儲存相關的資源具有以下許可權：

```
{
  "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"
 }
  ]
}

--您也可以自訂其他許可權
```

在 **MaxCompute** 中使用 **AccessKey** 訪問**Table Store**

同授權方式不同的是，需要在建立外表時在LOCATION中顯示寫入 AK 資訊，其格式為：

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

假設需要訪問的Table Store資源的資訊為：

AccessKeyId	**AccessKeyS ecret**	執行個體名稱	地區	網路模式
abcd	1234	cap1	cn-hangzhou	內網訪問

建立外表的語句為：

```
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'
```

對資料訪問的操作步驟請參考*##MaxCompute##Table Store*中的步驟3.通過外部表格訪問 Table Store 資料。

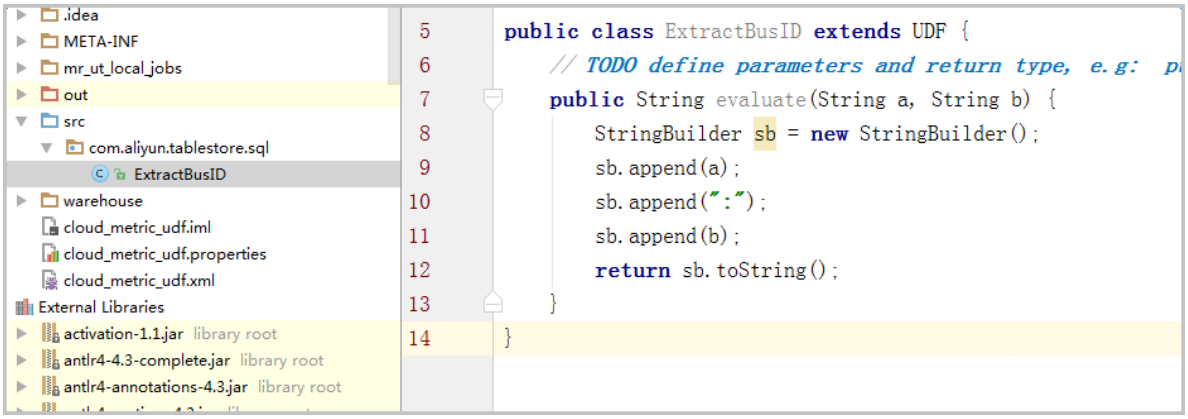
2.4 使用 UDF 處理資料

如果您在Table Store裡面的資料有著獨特的結構，希望自訂開發邏輯來處理每一行資料，比如解析特定的 json 字串，可以使用 UDF（User Defined Function，即使用者自訂函數）來處理。

操作步驟

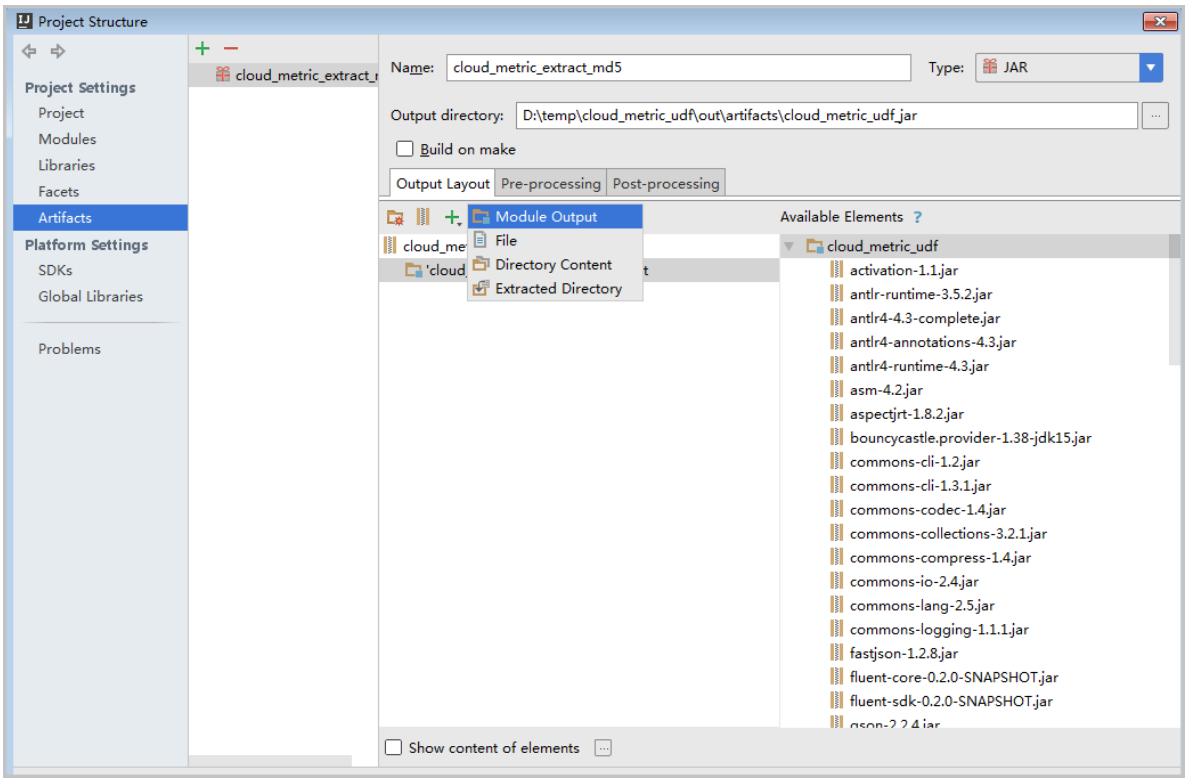
1. 參考*MaxCompute Studio* 文檔，在 IntelliJ 中安裝 MaxCompute-Java/MaxCompute-Studio 外掛程式。外掛程式安裝完畢，就可以直接開發。

   下圖是一個簡單的 UDF 定義，將兩個字串串連。MaxCompute 支援更複雜的 UDF，包括自訂視窗執行邏輯等，更多資訊請參考*##### UDF*。



2. 打包之後可以上傳到 MaxCompute。

   選擇 **File** > **Project Structure** > **Artifacts**，輸入Name 和 Output directory 後，單擊 **+** 選擇輸出模組。打包後通過 ODPS Project Explorer 來上傳資源、建立函數，然後就可以在 SQL 中調用。



3. 運行 bin/odpscmd.bat。

```
// 我們選出來1行資料，並將name/name傳入UDF，返回兩個string的累加
select cloud_metric_extract_md5(name, name) as udf_test from
test_table limit 1;
```

返回結果如下：

```
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
```


2.5 常見問題

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

原因：需要設定 set odps.task.major.version=unstructured_data。

- FAILED: ODPS-0010000:System internal error - std::exception:Message: a timeout was reached

  原因：一般情況下是Table Store的 endpoint 填寫錯誤，導致 MaxCompute 無法訪問。

- logview invalid end_point

  原因：在執行過程中，會返回 logview URL 地址，如果使用瀏覽器訪問該地址返回錯誤，可能是配置不對，請檢查 MaxCompute 配置。

如果仍未解決問題，請####。

3 Hive/HadoopMR

3.1 環境準備

使用 **Hive/HadoopMR** 來訪問**Table Store**中的表

通過*Table Store*及 *E-MapReduce* 官方團隊發布的依賴包，可以直接使用 Hive 及 HadoopMR 來訪問Table Store中的資料並進行資料分析。

安裝 **JDK-7+**

1. 下載並安裝 JDK-7+ 安裝包。
   - Linux/MacOS 系統：使用系統內建的包管理器安裝
   - Windows 系統：####
2. 按照以下樣本進行安裝檢查。

```
$ 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)
```

安裝並啟動 **Hadoop** 環境

1. 下載 2.6.0 版本以上的 Hadoop 安裝包。（####）
2. 解壓並安裝，根據實際叢集情況安裝 Hadoop 服務。
3. 按照如下樣本啟動 Hadoop 環境。

```
$ bin/start-all.sh
# 檢查服務是否成功啟動
$ jps
24017 NameNode
24835 Jps
24131 DataNode
24438 ResourceManager
5114 HMaster
24287 SecondaryNameNode
24527 NodeManager
```

4. 在 `/etc/profile` 中添加 Hadoop 路徑，並執行 `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
```

下載及安裝 **Hive** 環境

1. 下載類型為 bin.tar.gz 的 Hive 安裝包。（####）

2. 按照如下樣本解壓安裝包。

```
$ 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. 按照如下樣本初始化 schema。

```
 # 進入指定的目錄
 $ cd /home/admin/hive-2.1.0/

 # 初始化,如果是mysql則derby可以直接替換成mysql
 # 如果執行出錯可以刪除rm -rf metastore_db/之後重新執行
 $ ./bin/schematool -initSchema -dbType derby
```

4. 按照如下樣本啟動 Hive 環境。

```
$ ./bin/hive
# 檢查服務是否成功啟動
hive> show databases;
OK
default
Time taken: 0.207 seconds, Fetched: 1 row(s)
```

下載**Table Store**的 **Java SDK**

1. 在 Maven 庫中下載 4.1.0 版本以上的 Java SDK 相關依賴包。（####）

> 说明：
> 該依賴包會隨最新的 Java SDK 發布，請根據最新的 *Java SDK* ##下載相關依賴包。

2. 按照如下樣本將 SDK 拷貝到 Hive 目錄下。

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

下載阿里雲 **EMR SDK**

EMR SDK 依賴包。

> 说明：
> 瞭解更多 EMR 資訊請參考###

3.2 使用教程

資料準備

在Table Store中準備一張資料表 pet，`name` 是唯一的一列主鍵，資料樣本如下：

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寫入，因為Table Store是一個 schema-free 的儲存結構（####），沒有值也不需要寫入NULL。

Hive 訪問樣本

前提條件

按照####準備好 Hadoop、Hive、JDK 環境以及Table Store JAVA SDK 和 EMR SDK 依賴包。

樣本

```
# HADOOP_HOME 及 HADOOP_CLASSPATH 可以添加到 /etc/profile 中
$ export HADOOP_HOME=${你的 Hadoop 安裝目錄}
$ 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)
```

參數說明如下：

- WITH SERDEPROPERTIES

  tablestore.columns.mapping（可選）：在預設的情況下，外表的欄位名即為Table Store上表的列名（主鍵列名或屬性列名）。但有時外表的欄位名和表上列名並不一致（比如處理大小寫或字元集相關的問題），這時候就需要指定 tablestore.columns.mapping。該參數為一個英文逗號分隔的字串，每一項都是表上列名，順序與外表欄位一致。

  > 说明：
  > 空白也會被認為是表上列名的一部分。

- TBLPROPERTIES
  - tablestore.endpoint（必填）：訪問Table Store的####，也可以在Table Store控制台上查看這個執行個體的 endpoint 資訊。
  - tablestore.instance（可選）：Table Store的####名。若不填，則為 tablestore.endpoint 的第一段。
  - tablestore.table.name（必填）：Table Store上對應的表名。
  - tablestore.access_key_id、tablestore.access_key_secret（必填），請參見####。
  - tablestore.sts_token（可選），請參見####。

HadoopMR 訪問樣本

以下樣本介紹如何使用 HadoopMR 程式統計資料表 pet 的行數。

範例程式碼

- 構建 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));
    }
}
}
```

資料來源每從Table Store上讀出一行，都會調用一次 mapper 的 map()。前兩個參數 PrimaryKeyWritable 和 RowWritable 分別對應這行的主鍵以及這行的內容。可以通過調用 PrimaryKeyWritable.getPrimaryKey() 和 RowWritable.getRow() 取得Table Store JAVA SDK 定義的主鍵對象及行對象。

- 配置Table Store作為 mapper 的資料來源。

```
 private static RangeRowQueryCriteria fetchCriteria() {
     RangeRowQueryCriteria res = new  RangeRowQueryCriteria("
YourTableName");
     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) 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);
    }
```

範例程式碼中使用 job.setInputFormatClass(TableStoreInputFormat.class) 把Table Store設為資料來源，除此之外，還需要：

- 把 hadoop-connector.jar 部署到叢集上並添加到 classpath 裡面。路徑為 addFileToClassPath() 指定 hadoop-connector.jar 的本地路徑。代碼中假定 hadoop-connector.jar 在當前路徑。
- 訪問Table Store需要指定入口和身份。通過 TableStoreInputFormat.setEndpoint() 和 TableStoreInputFormat.setCredential() 設定訪問Table Store需要指定的 endpoint 和 access key 資訊。
- 指定一張表用來計數。

> 说明：
>
> - 每調用一次 addCriteria() 可以在資料來源裡添加一個 JAVA SDK 定義的 RangeRowQueryCriteria 對象。可以多次調用addCriteria()。RangeRowQueryCriteria 對象與Table Store JAVA SDK GetRange 介面所用的 RangeRowQueryCriteria 對象具有相同的限制條件。
> - 可以利用 RangeRowQueryCriteria 的 setFilter() 和 addColumnsToGet() 在Table Store的伺服器端過濾掉不必要的行和列，減少訪問資料的大小，降低成本，提高效能。
> - 通過添加對應多張表的多個 RangeRowQueryCriteria，可以實現多表的 union。
> - 通過添加同一張表的多個 RangeRowQueryCriteria，可以做到更均勻的切分。TableStore-Hadoop Connector 會根據一些策略將使用者傳入的範圍切細。

程式運行樣本

```
$ 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
```

```
T0TAL    9
```

類型轉換說明

Table Store支援的資料類型和 Hive/Spark 支援的資料類型不完全相同。

下表列出了從Table Store的資料類型（行）轉換到 Hive/Spark 資料類型（列）的支援情況。

	TINYINT	SMALLIN	INT	BIGINT	FLOAT	DOUBLE	BOOLEA	STRING	BINARY
INTEGER	可，損失精度	可，損失精度	可，損失精度	可	可，損失精度	可，損失精度			
DOUBLE	可，損失精度	可，損失精度	可，損失精度	可，損失精度	可，損失精度	可			
BOOLEAN							可		
STRING								可	
BINARY									可

4 Spark/SparkSQL

4.1 環境準備

使用 **Saprk/Spark SQL** 來查詢和連結資料表格儲存中的表

通過*Table Store*及 *E-MapReduce* 官方團隊發布的依賴包，可以直接使用 Spark 及 Spark SQL 來訪問Table Store中的資料並進行資料的查詢分析。

下載及安裝 **Spark/Spark SQL**

1. 下載版本號碼為 1.6.2 的 Spark 安裝包，安裝包類型為 Pre-built for Hadoop 2.6。（####）
2. 按照如下樣本解壓安裝包。

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

安裝 **JDK-7+**

1. 下載並安裝 JDK-7+ 安裝包。
   - Linux/MacOS 系統：請用系統內建的包管理器進行安裝
   - Windows 系統：####
2. 按照如下樣本進行安裝檢查。

```
$ 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)
```

下載**Table Store**的 **Java SDK**

1. 在 Maven 庫中下載 4.1.0 版本以上的 Java SDK 相關依賴包。（####）

   说明：
   該依賴包會隨最新的 Java SDK 發布，請根據最新的 *Java SDK* ##下載相關依賴包。
2. 按照如下樣本將 SDK 拷貝到 Spark 目錄下。

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

下載阿里雲 **EMR SDK**

下載 EMR SDK 相關的依賴包。（####）

说明：

瞭解更多 EMR 資訊請參見##。

啟動 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
```

4.2 使用教程

資料準備

在Table Store中準備一張資料表 pet，其中name是唯一的一列主鍵。資料樣本如下：

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寫入，因為Table Store是一個 schema-free 的儲存結構（####），沒有值也不需要寫入NULL。

Spark SQL 訪問樣本

前提條件

按照####中的步驟準備好 Spark、JDK 環境以及Table Store Java SDK 和 EMR SDK 的依賴包。

樣本

```
$ 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)
```

參數說明如下：

- WITH SERDEPROPERTIES
  - tablestore.columns.mapping（可選）：在預設情況下，外表的欄位名即為Table Store上表的列名（主鍵列名或屬性列名）。但有時外表的欄位名和表上列名並不一致（比如處理大小寫或字元集相關的問題），這時候就需要指定 tablestore.columns.mapping。該參數為一個英文逗號分隔的字串，每個分隔之間不能添加空格，每一項都是表上列名，順序與外表欄位一致。

    说明：

    Table Store的列名支援空白字元，所以空白也會被認為是表上列名的一部分。
- TBLPROPERTIES
  - tablestore.endpoint（必填）：訪問Table Store的####，也可以在Table Store控制台上查看這個執行個體的 endpoint 資訊。
  - tablestore.instance（可選）：Table Store的####名。若不填，則為 tablestore.endpoint 的第一段。
  - tablestore.table.name（必填）：Table Store上對應的表名。
  - tablestore.access_key_id、tablestore.access_key_secret（必填），請參見####。

- tablestore.sts_token（可選），請參見####。

Spark 訪問樣本

以下樣本介紹如何使用 Spark 程式統計資料表 pet 的行數。

```
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 換成 SparkContext，JavaPairRDD 換成 PairRDD 即可。或者更簡單，交給編譯器自行做類型推斷

運行程式

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

類型轉換說明

Table Store支援的資料類型和 Hive/Spark 支援的資料類型不完全相同。

下表列出了從Table Store的資料類型（行）轉換到 Hive/Spark 資料類型（列）時所支援的情況。

	TINYINT	SMALLIN	INT	BIGINT	FLOAT	DOUBLE	BOOLEA	STRING	BINARY
INTEGER	可，損失精度	可，損失精度	可，損失精度	可	可，損失精度	可，損失精度			
DOUBLE	可，損失精度	可，損失精度	可，損失精度	可，損失精度	可，損失精度	可			
BOOLEAN							可		
STRING								可	
BINARY									可