

Alibaba Cloud Tablestore

計算と分析

Document Version20190909

目次

1 関数トリガー	1
1.1 Function Compute の紹介.....	1
1.2 データクレンジングへの Function Compute の使用.....	9
2 MaxCompute	17
2.1 1つのアカウントを使用して MaxCompute の Table Store へのアクセスを許可 する.....	17
2.2 AccessKeys を使用した MaxCompute の Table Store へのアクセス許可.....	23
2.3 UDF を使用したデータ処理.....	25
2.4 一般的なエラーのトラブルシューティング.....	27
3 Hive と HadoopMR	28
3.1 環境の準備.....	28
3.2 チュートリアル.....	30
4 Spark と SparkSQL	36
4.1 環境の準備.....	36
4.2 チュートリアル.....	37

1 関数トリガー

1.1 Function Compute の紹介

Function Compute は、サーバーの動作条件などを管理しなくても、ユーザーがコードを書いてアップロードできるようにするイベント駆動型サービスです。

ここでは、Function Compute を使用して、Table Store データテーブル内の増分データに対してリアルタイム計算を実行する方法を紹介します。

Function Compute は、ユーザーコードを実行するために適切な量のコンピューティングリソースを準備し、自動スケーリングします。ユーザーは自分のコードを実行するために必要なリソースに対してのみ支払います。詳細については、「[Function Compute とは](#)」を参照ください。

Table Store Stream は、Table Store データテーブルの増分データを取得するために使用されるデータチャンネルです。Table Store トリガーを作成することで、Table Store Stream と Function Compute を自動的にドッキングすることができます。これにより、コンピューティング機能内のカスタムプログラムロジックは、Table Store データテーブル内の変更されたデータを自動的に処理することができます。詳細については、「[Table Store Stream](#)」を参照ください。

ここでは、Function Compute を使用して、Table Store データテーブル内の増分データに対してリアルタイム計算を実行する方法を紹介します。

Table Store トリガーの設定

コンソールを使用して、Table Store データテーブルからのリアルタイムデータストリームを処理する Table Store トリガーを作成できます。

1. ストリーミングを有効にしてデータテーブルを作成します。

a. Table Store コンソールでインスタンスを作成します。

Create Instance

* Instance Name: testInstance

* Instance Type: Capacity

* Instance Description: test instance for FunctionCompute

Note:

1. It takes you several seconds to create an instance. Wait a moment and then click the Refresh button.

2. After the instance is created, its access URL will take effect within 1 minutes.

OK Cancel

b. このインスタンスの下にデータテーブルを作成し、ストリーミングを有効にします。

Create Table

* Table Name: streamDataTable

* Time To Live: -1 seconds
The minimum TTL is 86400 seconds (one day) or -1 (no expiry).

* Max Versions: 1
Data versions must be non-zero values.

* Max Version Offset: 86400 seconds
The difference between the version number of the write data and the write time must be within the value of Max Version Offset.

* Primary Key: id String Partition Key
Each table can have a maximum of 4 Primary Keys. You have created 1.
To ensure best use of Reserved Throughput and to improve load balancing, hashing or similar methods should be used to evenly distribute data in the Partition key space. For more details, please refer to "Best Practices".
+ Add a Primary Key

Enable Stream Note: See pricing.

* Stream Records Expiration Time: 12

Refresh Bind VPC

OK Cancel

2. Function Compute 関数を作成します。

a. Function Compute コンソールでサービスを作成します。

The screenshot shows a 'Create Service' dialog box with the following details:

- Service Name:** testService
- Region:** China East 2 (Shanghai)
- Description:** Process the data from tablestore
- Advanced Settings:** A toggle switch is currently turned off.

Below the Service Name field, there are three rules:

1. Only letters, numbers, underscores (_), and hyphens (-) are allowed.
2. The name cannot start with a number or hyphen.
3. The name has to be between 1 to 128 characters in length.

Below the Region field, there is a note: "Services in the same region can communicate with each other over the intranet. The region cannot be changed after the service is created."

- b. サービスの詳細設定では、ログを収集し、コンピューティング機能内のユーザーの他のリソースに引き続きアクセスするために、認証機能に対するサービスのロールを設定できます。詳細については、「[ユーザー権限](#)」を参照ください。

The screenshot shows the details page for a service named 'testService' in the 'China East 2 (Shanghai)' region. The page includes the following sections:

- Usage:** Shows 'Invocations (This Month)' as 0.0 and 'Resource Usage (This Month)' as 0 GB-S. A note indicates that usage data is updated hourly.
- Basic Configurations:** Lists the service name, region, created time (06/12/2018, 13:23:51), last modified time (06/12/2018, 13:23:51), and description (Process the data from tablestore).
- Advanced Configurations:** Shows 'Log Project' and 'Service Role' as LogStore.

- c. 新しく作成されたサービスの下で [新規関数] をクリックします。
- d. 「機能テンプレート」ページで [空白の関数] をクリックします。
- e. 「トリガー設定」ページで、[トリガーを作成しない] をクリックして、[次へ] をクリックします。
- f. 機能情報を設定します。

Table Store トリガーでは **CBOR 形式** を使用して、増分データを Function Compute イベントとしてエンコードし、ユーザー関数を呼び出します。次の関数例は、イベントをデ

コードしてログセンターに出力します。データがデコードされた後に、必要に応じてデータを処理できます。

Basic Management Configuration

Function Information

* Service Name: [Create Service](#)

* Function Name:

1. Only letters, numbers, underscores (_), and hyphens (-) are allowed.
2. It cannot start with a number or hyphen.
3. The name must be 1 to 128 characters in length.

Function Description:

* Runtime:

Code Configuration

Function Code In-line Edit Import from OSS Upload Zip File Upload Folder

```
1 # -*- coding: utf-8 -*-
2 import logging
3 import cbor
4
5 def handler(event, context):
6     logger = logging.getLogger()
7     records = cbor.loads(event)
8     logger.info('TableStore stream records sample %s', records)
9     return 'hello world'
```

Previous Next

* Function Handler:

Handler is defined in the format of "[File name].[Method name]". Handler "index.handler" implies that index.py file contains a method called "handler". Follow this [link](#) for more information. [Documents](#)

* Memory:

* Timeout: seconds

Previous Next

3. Table Store トリガーを作成してテストします。
 - a. **Table Store コンソール**で新しく作成されたデータテーブルの下で、**[既存の関数を使用]**をクリックして、トリガーを作成します。
 - b. 作成中に、イベント通知を送信するように Table Store を認証する必要があります。

クリックすると、自動的に作成された認証ロール

AliyunTableStoreStreamNotificationRole を **RAM コンソール**で確認できます。

Create Trigger (Use Existing FC Function)

* FC Service: testService

* FC Function: streamProcessFunction

* Trigger Name: testTrigger

Table Store has been granted the permission to send event notifications.

OK Cancel

情報処理

- ・ データフォーマット

Table Store トリガーでは **CBOR 形式**を使用しています。増分データをエンコードして関数計算イベントを形成します。増分データの具体的なデータ形式は次のとおりです。

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

```
}
```

- ・ **メンバー定義**

- **バージョン**

- **説明:** ペイロードのバージョン番号は、現在は Sync-v1 です。

- **タイプ:** 文字列

- **記録**

- **説明:** データテーブル内の増分データ行配列です

- **以下を含みます。**

- **タイプ**

- **説明:** データ行タイプ。PutRow、UpdateRow または DeleteRow です。

- **タイプ:** 文字列

- **情報**

- **説明:** データ行に関する基本情報です。

- **以下を含みます。**

- **タイムスタンプ**

- **説明:** 行の最終更新時刻 (UTC) です。

- **タイプ:** int64

- **プライマリキー**

- **説明:** プライマリキー列の配列

- **以下を含みます。**

- **ColumnName**

- **説明:** プライマリキー列名です

- **型:** 文字列

- **値**

- **説明:** プライマリキー列の内容です。

- **型:** Formated_value です。整数、文字列、または BLOB です。

- **カラム**

- **説明:** 属性カラム配列

- **以下を含みます。**

- **タイプ**

■ 説明: 属性列タイプ。Put、DeleteOneVersion または DeleteAllVersions のいずれかです。

■ 型: 文字列

■ ColumnName

■ 説明: 属性列名

■ 型: 文字列

■ 値

■ 説明: 属性列の内容

■ タイプ: Formated_value です。整数、ブール値、ダブル、文字列または BLOB です。

■ タイムスタンプ

■ 説明: 属性列の最終更新日時 (UTC) です。

■ タイプ: int64

・ データ例

```
{
  " Version ": " Sync - v1 ",
  " Records ": [
    {
      " Type ": " PutRow ",
      " Info ": {
        " Timestamp ": 1506416585 740836
      },
      " PrimaryKey ": [
        {
          " ColumnName ": " pk_0 ",
          " Value ": 1506416585 881590900
        },
        {
          " ColumnName ": " pk_1 ",
          " Value ": " 2017 - 09 - 26 17 : 03 : 05 .
8815909 + 0800 CST "
        },
        {
          " ColumnName ": " pk_2 ",
          " Value ": 1506416585 741000
        }
      ],
      " Columns ": [
        {
          " Type ": " Put ",
          " ColumnName ": " attr_0 ",
          " Value ": " hello_tabl e_store ",
          " Timestamp ": 1506416585 741
        },
        {
          " Type ": " Put ",
          " ColumnName ": " attr_1 ",
          " Value ": 1506416585 881590900 ,

```

```

    " Timestamp ": 1506416585 741
  }
}
}
}
}

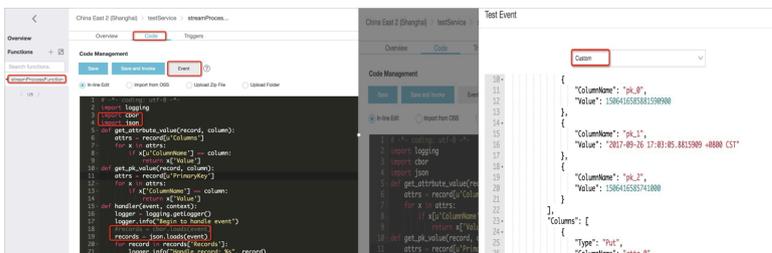
```

オンラインデバッグ

Function Compute は関数のオンラインデバッグをサポートします。ユーザーはトリガーされたイベントを作成してコードロジックをテストできます。

Table Store トリガー機能サービスのイベントは CBOR 形式であるため、データ形式は JSON に似たバイナリ形式であり、次のようにオンラインでデバッグできます。

1. コードに CBOR と JSON を同時にインポートします。
2. [トリガーイベント] をクリックし、 [カスタム] をクリックします。上記のデータ例から JSON ファイルを編集ボックスに貼り付けます。要件に基づいて変更して保存します。
3. コードでは、最初に `" records = json . loads (event)"` を使用して、カスタマイズしたテストトリガーイベントを処理します。その後、 [実行] をクリックして、コードをテストします。
4. `" records = json . loads (event)"` のテストが成功したら、コードを `" records = cbor . loads (event)"` に変更して、保存することができます。その後、データが Table Store に書き込まれると、関連する関数ロジックがトリガーされます。



サンプルコード

```

import logging
import cbor
import json
def get_attrbu te_value ( record , column ):
    attrs = record [ u ' Columns ' ]
    for x in attrs :
        if x [ u ' ColumnName ' ] == column :
            return x [ ' Value ' ]
def get_pk_val ue ( 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_attribute_value ( record , " attr_0 ")
return ' OK '
```

1.2 データクレンジングへの Function Compute の使用

このドキュメントでは、Function Compute を使用して Table Store 内のデータをクレンジングする方法について説明します。

Table Store の高度な同時書き込みパフォーマンスと低ストレージコストにより、ログの保存やデータの監視のような、モノのインターネット (IoT) アプリケーションに最適です。データを Table Store に書き込み、同時に Function Compute に新しく追加されたデータの単純なクレンジングを実行し、クレンジングされたデータを Table Store の結果テーブルに書き戻すことができます。その間、元のデータと結果データにリアルタイムでアクセスできます。

データ定義

書き込まれるデータが 3 つのフィールドを含むログデータである前提とします。

フィールド名	タイプ	意味
id	Integer 型	ログ ID
level	Integer 型	ログレベル (数値が大きいほどレベルが高い)
message	String 型	ログ内容

専用クエリを実行するには、レベル 1 を超えるログを別のデータテーブルに書き込む必要があります。

インスタンスとデータテーブルの作成

Table Store コンソール (今回は例として East China Node 2 の分散テストを使用) で Table Store インスタンスを作成し、ソーステーブル (source_data) と結果テーブル (result) を作成します。プライマリーキーは id (整数) です。Table Store はスキーマフリー構造を使用するため、他の属性列フィールドを事前定義する必要はありません。

例として source_data を取り上げ、次の図のように作成します。

Create Table
×

* Table Name:

* Time To Live: seconds
The minimum TTL is 86400 seconds (one day) or -1 (no expiry).

* Max Versions:
Data versions must be non-zero values.

* Max Version Offset: seconds
The difference between the version number of the write data and the write time must be within the value of Max Version Offset.

* Primary Key:
 Partition Key
Each table can have a maximum of 4 Primary Keys. You have created 1.
To ensure best use of Reserved Throughput and to improve load balancing, hashing or similar methods should be used to evenly distribute data in the Partition key space. For more details, please refer to "Best Practices".
[+ Add a Primary Key](#)

Enable Stream Note: See [pricing](#).

データソーステーブルのストリーミング機能を有効にする

トリガー関数では、Table Store に書き込まれた増分データを Function Compute で処理する前に、データテーブルの **ストリーム関数** が有効化されている必要があります。

Table Name	Time To Live	Max Versions	Max Version Offset	Stream Status	Monitor	Actions
result	-1	1	86400	Disabled	📊	Manage Enable Stream Use Trigger Modify Attributes Delete
source_data	-1	1	86400	Disabled	📊	Manage Enable Stream Use Trigger Modify Attributes Delete

ストリームレコードの有効期限は、増分データがストリーム API を通じて読み取られる最大時間です。

トリガーは既存の関数のみをバインドできるため、最初に Function Compute コンソールで同じリージョンにサービスと関数を作成します。

Function Compute サービスの作成

次の手順では、例として East China Node 2 を使用して、**Function Compute コンソール**でサービスと処理関数を作成する方法について説明します。

1. East China Node 2 でサービスを作成します。

Create Service

* Service Name

1. Only letters, numbers, underscores (_), and hyphens (-) are allowed.
2. The name cannot start with a number or hyphen.
3. The name has to be between 1 to 128 characters in length.

Region **China East 2 (Shanghai)**

Services in the same region can communicate with each other over the intranet. The region cannot be changed after the service is created.

Description

Advanced Settings

2. 関数を作成して、[空白の関数] > [トリガーを作成しない] をクリックします。

Function Information

* Service Name Create Service

* Function Name

1. Only letters, numbers, underscores (_), and hyphens (-) are allowed.
2. It cannot start with a number or hyphen.
3. The name must be 1 to 128 characters in length.

Function Description

* Runtime

Code Configuration

Function Code In-line Edit Import from OSS Upload Zip File Upload Folder

* Function Handler

Handler is defined in the format of "[File name],[Method name]". Handler "index.handler" implies that index.py file contains a method called "handler". Follow this [link](#) for more information. [Documents](#)

* Memory

* Timeout seconds

Previous Next

- ・ 関数名は "etl_test" です。Python 2.7 環境を選択して、コードをオンラインで編集してください。
- ・ 関数のエントリは、" etl_test.handler" です。
- ・ コードは後で編集されます。次に [次へ] をクリックします。

3. サービス認証

Function Compute は実行中のログをログサービスに書き込むと同時に Table Store データテーブルの読み取りと書き込みを行うため、Function Compute には特定の権限が必要です。便宜のために、最初に "AliyunOTSFullAccess" および "AliyunLogFullAccess" 許可を追加してください。実際の使用では、最小特権の原則に基づいて権限を追加することを推奨します。

Service Role Information (authorizes FC to access other resources)

Service Name transform_test

ⓘ Hover over role policies. Add the recommended policies to your existing role if needed. ✕

> Role(acs:ram::5373096288841079:role/test)Permissions granted or to be granted.

Permission Configuration

4. [認証を完了する] をクリックし、関数を作成します。

5. 機能コードの修正

関数を作成した後、対応する [関数] - [コード実行] をクリックします。その後、コードを編集し保存します。必要に応じて、"INSTANCE_NAME" (Table Store のインスタンス名) と "REGION" (使用されているリージョン) を変更します。

The screenshot shows the AWS Lambda console interface for a function named 'etl_test' in the 'China East 2 (Shanghai)' region. The 'Code' tab is selected, and the code editor displays the following Python code:

```

1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 import cbor
4 import json
5 import tablestore as ots
6
7 INSTANCE_NAME = 'distribute-test'
8 REGION = 'cn-shanghai'
9 ENDPOINT = 'http://%.s.ots-internal.aliyuncs.com'%(INSTANCE_NAME, REGION)
10 RESULT_TABLENAME = 'result'
11
12
13 def _utf8(input):
14     return str(bytearray(input, "utf-8"))
15
16 def get_attribute_value(record, column):
17     attrs = record[u'Columns']
18     for x in attrs:
19         if x[u'ColumnName'] == column:
20             return x['Value']
21
22

```

A red box highlights lines 7-10, and a red arrow points to it with the text "This parameters should be modified".

サンプルコード:

```

#!/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.ots-internal.aliyuncs.com'
'%(INSTANCE_NAME, REGION)
RESULT_TABLENAME = 'result'
def _utf8(input):
    return str(bytearray(input, "utf-8"))
def get_attribute_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']
# AliyunOTSFullAccess 許可が付与されているため、ここで取得した認証情報は
# Table Store へのアクセスを許可されています。
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_attribute_value ( record , ' level '))
    msg = get_attribute_value ( record , ' message ' )
    pk = [ ( _utf8 ( ' id '), id ),]
    attr = [ ( _utf8 ( ' level '), level ), ( _utf8 ( ' message '),
    _utf8 ( msg )),]
    row = ots . Row ( pk , attr )
    client . put_row ( RESULT_TABLE_NAME , 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_attribute_value ( record , ' level '))
        if level > 1 :
            save_to_ots ( client , record )
        else :
            print " Level <= 1 , ignore ."

```

トリガーのバインド

1. Table Store のインスタンス管理ページに戻り、source_data テーブルの後ろにある [トリガーを使用] ボタンをクリックして、トリガーバインディングインターフェイスに入ります。[既存の関数を使用] をクリックし、新しく作成したサービスと関数を選択して、Table Store のイベント通知を送信許可を確認のためにチェックします。

Table Name	Time To Live	Max Versions	Max Version Offset	Stream Status	Monitor	Actions
result	-1	1	86400	Disabled		Manage Enable Stream Use Trigger Modify Attributes Delete
source_data	-1	1	86400	Enabled		Manage Disable Stream Use Trigger Modify Attributes Delete

source_data

Create Trigger

Create Function Compute

Trigger List

Only one trigger is supported

FC Service Name

Create Trigger (Use Existing FC Function)

* FC Service: transform_test

* FC Function: etl_test

* Trigger Name: tablestore_etl

Table Store has been granted the permission to send event notifications.

OK Cancel

2. バインドが成功すると、以下の情報が表示されます。

source_data

Create Trigger

Create Function Compute Use Existing Function Compute Refresh

Trigger List

Only one trigger is supported in a table.

FC Service Name	FC Function Name	Trigger Name	Actions
transform_test	etl_test	tablestore_etl	Edit/Test Delete

実行の確認

1. source_data テーブルにデータを書き込みます。

source_data の「データエディタ」ページで、[挿入] をクリックし、id、レベルおよびメッセージ情報を次の順序で入力します。

source_data

Details

Data Editor

Table Data

Table can display up to 100 rows.

Trigger

Data Monitor

Insert

Name	Type	Primary Key Value
id	INTEGER	1

Remove all attribute columns

Name	Type	Value	Version	Actions
level	INTEGER	2		Delete
		or <input checked="" type="checkbox"/> System time		
message	STRING	Test data		Delete
		or <input checked="" type="checkbox"/> System time		

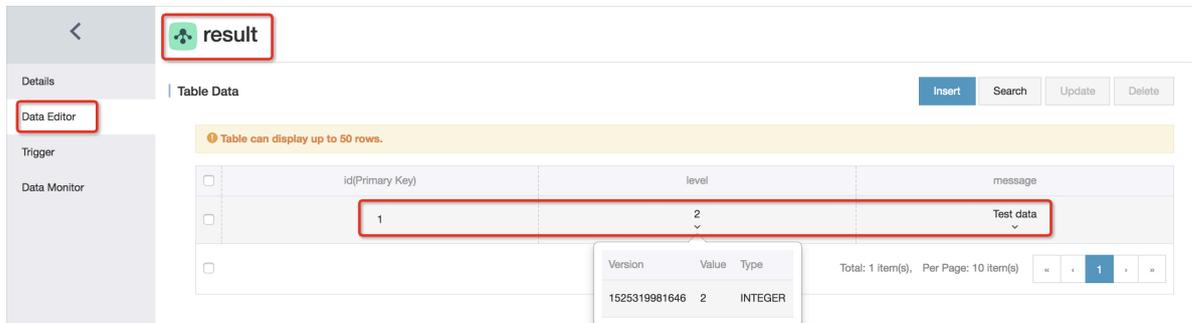
+ Add Column Attribute custom inserts available: 20 columns. Already created: 2 columns.

Insert Cancel

2. 結果テーブルからクレンジングされたデータのクエリを実行します。

結果テーブルの「データエディタ」ページをクリックします。ここで、`source_data` に新しく書き込まれたデータのクエリを実行できます。

`source_data` に書き込まれたレベル 1 以下のデータは結果テーブルと同期されません。



The screenshot shows the 'Data Editor' page for a table named 'result'. The table has three columns: 'id(Primary Key)', 'level', and 'message'. A single row of data is displayed with 'id' value 1, 'level' value 2, and 'message' value 'Test data'. A tooltip is visible over the 'level' cell, showing 'Version: 1525319981646', 'Value: 2', and 'Type: INTEGER'. The interface includes a sidebar with 'Data Editor' selected, and a top navigation bar with 'result' and a back arrow. The table data is displayed in a grid with a yellow warning banner above it stating 'Table can display up to 50 rows.'.

id(Primary Key)	level	message
1	2	Test data

Version: 1525319981646 | Value: 2 | Type: INTEGER

Total: 1 item(s), Per Page: 10 item(s)

2 MaxCompute

2.1 1つのアカウントを使用して MaxCompute の Table Store へのアクセスを許可する

背景情報

このドキュメントでは、1つの Alibaba Cloud アカウントで Table Store と MaxCompute 間のシームレスな接続を確立する方法について説明します。

ビッグデータコンピューティングサービスとして、[MaxCompute](#) は、高速かつ完全にホストされた PB レベルのデータウェアハウスソリューションを提供し、大量のデータを経済的かつ効果的に分析し処理することができます。単純な DDL ステートメントを使用して MaxCompute 上に外部テーブルを作成し、MaxCompute テーブルを外部データソースに関連付けてさまざまなデータアクセスおよび出力機能を提供できます。MaxCompute テーブルには構造化データのみを含めることができ、外部テーブルには構造化データまたは非構造化データを含めることができます。

Table Store と MaxCompute の両方に独自の型システムがあり、次の表にそれらのマッピングを示します。

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

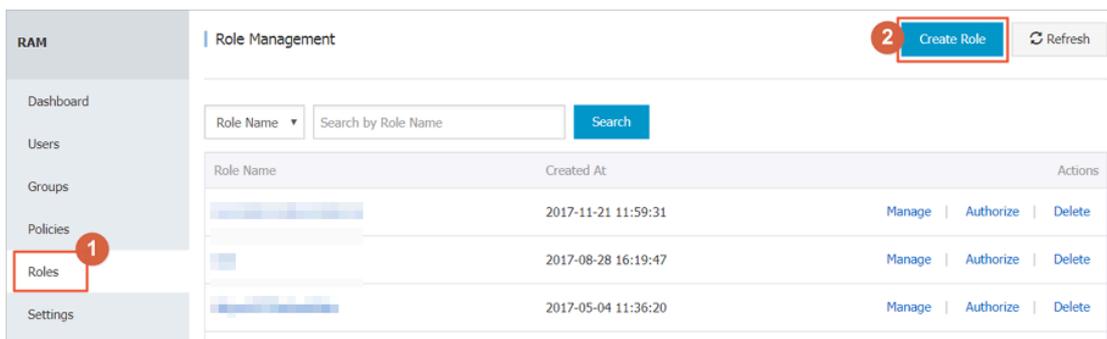
準備

MaxCompute を使用して Table Store にアクセスする前の準備

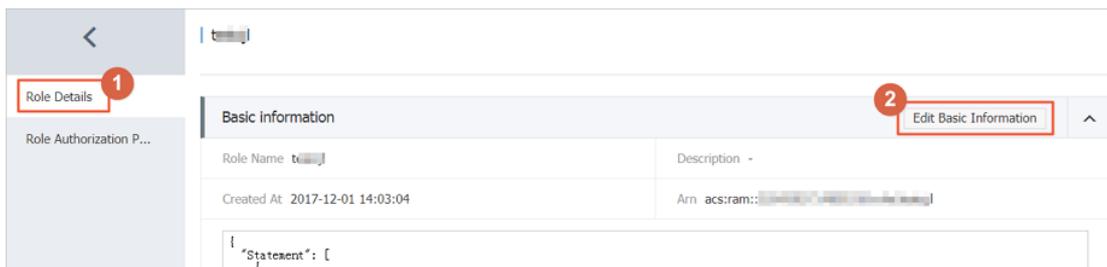
1. [MaxCompute サービス](#)をアクティベートします。
2. [MaxCompute プロジェクト](#)を作成します。
3. [AccessKey](#)を作成します。

4. RAM コンソールで、MaxCompute に Table Store へのアクセスを許可します。

- ・ 方法1: Alibaba Cloud アカウントでログインし、[ここをクリックしてクイック認証](#)します。
- ・ 方法2: 以下のステップを使用して手動許可を実行してください。
 - a. RAM コンソールにログインします。
 - b. 「ロール」 ページで、ユーザーロール AliyunODPSDefaultRole を作成します。



- c. 「ロールの詳細」 ページで、ポリシーの内容を設定します。

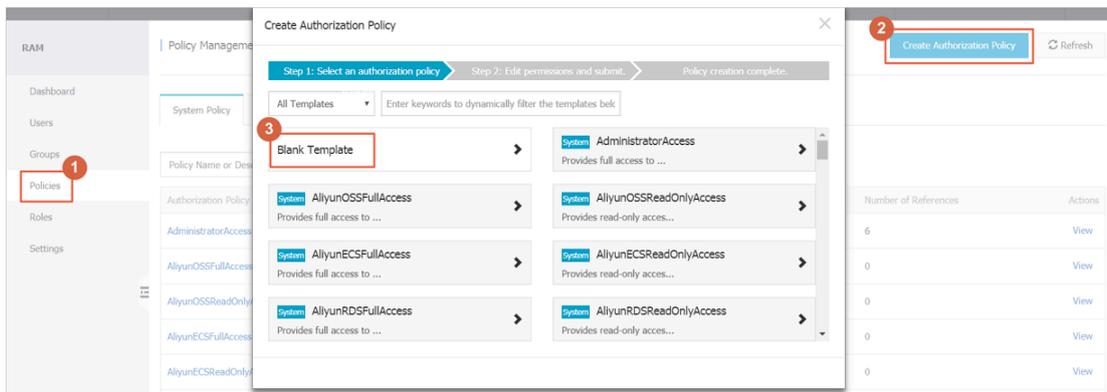


ポリシーの内容は次のように設定されています。

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

}

- d. 「ポリシー」 ページで、認証ポリシー AliyunODPSRolePolicy を作成します。

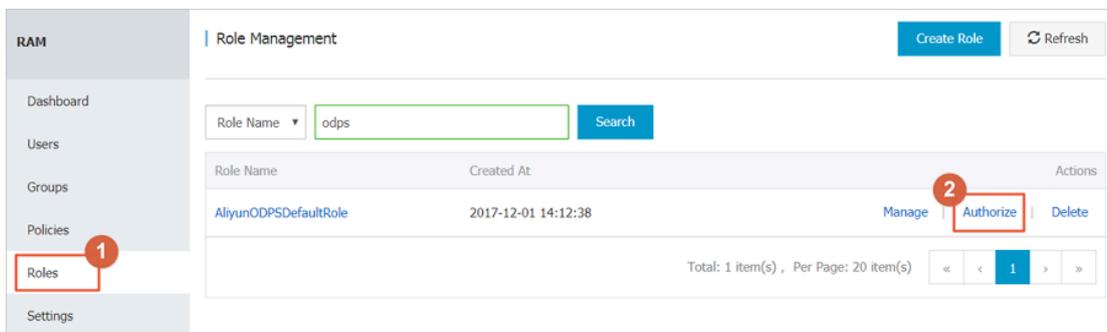


ポリシーの内容は次のように設定されています。

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

アクセス許可をカスタマイズすることもできます。

- e. 「ロール」 ページで、ロール AliyunODPSDefaultRole に AliyunODPSRolePolicy 許可を付与します。



5. Table Store コンソールで、インスタンスを作成し、テーブルを作成します。

この例では、Table Store インスタンスとデータテーブルの詳細は次のとおりです。

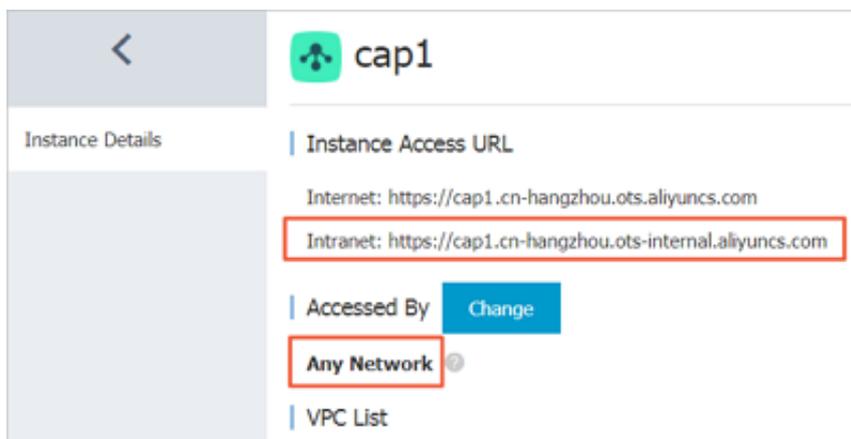
- ・ インスタンス名: cap1
- ・ データテーブル名: vehicle_track
- ・ プライマリキー情報: vid (整数)、gt (整数)
- ・ エンドポイント: 「 https :// cap1 . cn - hangzhou . ots - internal . aliyuncs . com 」



注:

MaxCompute を使用して Table Store にアクセスするときは、Table Store のプライベートネットワークアドレスを使用することを推奨します。

- ・ インスタンスのネットワークタイプを 任意のネットワーク に設定します。



ステップ1 クライアントのインストールと設定

1. MaxCompute クライアントをダウンロードして解凍します。



注:

JRE 1.7 以降のバージョンがコンピュータにインストールされていることを確認してください。

2. " conf / odps_conf_i g . ini " ファイルを編集して、クライアントを設定します。

```
access_id =*****
access_key =*****
# Alibaba Cloud アカウントの AccessID と AccessKey は Alibaba
  Cloud の公式 Web サイトの「管理コンソール」の「AccessKeys」ページで
  確認できます。
project_na me = my_project
# アクセスしたいプロジェクトスペースを指定します。
end_point = https :// service . odps . aliyun . com / api
# MaxCompute サービスのアクセスリンク。
tunnel_end point = https :// dt . odps . aliyun . com
```

```
# MaxCompute Tunnel サービスのアクセスリンク。
log_view_host = http://logview.odps.aliyun.com
# ジョブを実行した後、クライアントはそのジョブの LogView アドレスを返しま
す。アドレスにアクセスして、実行されたジョブの詳細を確認してください。
https_check = true
# HTTPS アクセスを有効にするかどうかを決定します。
```



注:

"odps_config.ini" ファイルでは、`"#"` が注釈として使用されます。MaxCompute クラ
イアントでは、`"--"` が注釈として使用されます。

3. `" bin / odpscmd . bat "` を実行して、`"show tables;"` と入力します。

現在の MaxCompute プロジェクトの表の表示は、前の設定が正しいことを示しています。

```
odps@ MCOTStest>show tables;

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

OK
```

ステップ 2 External Store の作成

MaxCompute データテーブル (ots_vehicle_track) を作成し、それを Table Store のテーブル (vehicle_track) に関連付けます。

この例では、関連付けられたテーブルは次のように詳述されています。

- ・ インスタンス名: cap1
- ・ テーブル名: vehicle_track
- ・ プライマリキー情報: vid (int)、gt (int)
- ・ エンドポイント: 「 `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_tr ack ' -- ( 4 )
)
LOCATION ' tablestore :// cap1 . cn - hangzhou . ots - internal .
aliyun . com '; -- ( 5 )
```

パラメーターは次のように記述されます。

番号	パラメーター	説明
(1)	com.aliyun.odps. TableStoreStorageHandler	StorageHandler は、Table Store データを処理するために MaxCompute を内蔵しています。これは MaxCompute と Table Store の間の相互作用を定義し、関連するロジックは MaxCompute によって実装されています。
(2)	SERDEPROPERITES	パラメーターオプションを提供するインターフェイスです。TableStore StorageHandler を使用する場合は、tablestore.columns.mapping と tablestore.table.name の 2 つのオプションを指定する必要があります。
(3)	tablestore.columns. mapping	必須オプションです。MaxCompute がアクセスする Table Store の列は、プライマリキーと属性の列を含みます。Table Store では、":" を含む列 (この例では :vid や :gt) はプライマリキー列で、その他は属性列です。 マッピングを指定するときは、指定された Table Store テーブルのすべてのプライマリキー列と、MaxCompute がアクセスする属性列を指定する必要があります。

番号	パラメーター	説明
(4)	tablestore.table.name	アクセスする Table Store テーブルの名前。指定された Table Store テーブル名が正しくない (存在しない) 場合、エラーが報告されず。MaxCompute は Table Store テーブルを事前に作成しません。
(5)	LOCATION	インスタンス名とエンドポイントを含む、アクセスする Table Store のインスタンス情報です。

ステップ 3 外部テーブルを介した Table Store のデータへのアクセス

外部テーブルが作成されると、Table Store データが MaxCompute エコシステムに導入され、MaxCompute SQL コマンドを使用してアクセスできるようになります。

```
// タイムスタンプ 1469171387 より前に VID が 4 未満の車両の平均速度とオイル消費量に関する統計を提供します。
select vid , count (*), avg ( speed ), avg ( oil_consumption )
from ots_vehicle_track where vid < 4 and gt < 1469171387
group by vid ;
```

次のような結果が返されます。

```
odps@ analysis_vehicle>select vid,count(*), avg(speed),avg(oil_consumption) from ots_vehicle_track where vid <4 and gt<1469171387 group by vid;
log view:
https://logview.odps.aliyun.com/logview/?http://service.odps.aliyun.com/api/analysis_vehicle/1-20170306160538185gsvlupk2&token=a3hxdK2MwVPCThskFD5hp5Gcm2tF6dEwP8FR8t
9KCTzoxj49R2M3Hzg2HTG0NjESLDE0008MjEwzgsay31d6f82d11bnQ10171KfJdglvblI6wyJv2Bz2011YwQ1xsw1R6z2w08Jjo1Qkx3c1LCSZMvd3J2516wyJhY3M6b2acc0m0y2p1Y3R2L2Fuw5c21x3Z
d011604w45z6Fuy2Vz1z1aMTcWtAZHTYwFMHTg1Z3N2Bhwaz1XK1dL3M2X3zawhuJjo1M539
job_queueing:
-----
STAGES      STATUS  TOTAL COMPLETED  RUNNING  PENDING  BACKUP
R1_job_0 ..... TERMINATED  1      1      0      0      0
R2_job_0 ..... TERMINATED  1      1      0      0      0
STAGES: 02/00 [=====] 100% ELAPSED TIME: 33.18 s
Summary:
-----
| vid | _c1 | _c2 | _c3 |
|----|----|----|----|
| 0 | 47 | 0.11622589796672624 | 6.5155861805388145 |
| 1 | 47 | 0.1128854978952555 | 6.5673938381827885 |
| 2 | 47 | 0.80837316346685 | 6.738738327683797 |
| 3 | 47 | 0.11583916531685494 | 6.6038962692751895 |
```

2.2 AccessKeys を使用した MaxCompute の Table Store へのアクセス許可

アカウント認証に加えて、MaxCompute の AccessKeys を使用して Table Store のデータにアクセスすることができます。

準備

ターゲットの Table Store リソースを所有するアカウントの AccessKey 認証情報 (つまり、[AccessKeyId](#) および [AccessKeySecret](#)) を取得します。AccessKey が Resource Access

Management (RAM) ユーザー用である場合、Table Store リソースを操作するには、RAM ユーザーに少なくとも次の許可が付与されている必要があります。

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

-- 他の権限も定義できます。

AccessKeys を使用して MaxCompute に Table Store へのアクセスを許可する

アカウント認証とは異なり、外部テーブルを作成するときは AccessKey 情報を " LOCATION " 節の中で特定する必要があります。

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

以下の情報が MaxCompute がアクセスしなければならないものであると仮定します。

AccessKeyId	AccessKeySecret	インスタンス名	リージョン	ネットワークモード
abcd	1234	cap1	cn-hangzhou	イントラネットアクセス

外部テーブルを作成するステートメントは次のとおりです。

```
CREATE EXTERNAL TABLE ads_log_ot_s_pt_external
(
  vid bigint ,
  gt bigint ,
  longitude double ,
  latitude double ,
  distance double ,
  speed double ,
  oil_consumption double
```

```

)
STORED BY ' com . aliyun . odps . TableStore StorageHan dler '
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 . aliuncs . com '

```

データへのアクセスの詳細については、「[1つのアカウントでMaxComputeがTable Storeにアクセスできるようにする](#)」のステップ3「外部テーブルを介したTable Storeのデータへのアクセス」をご参照ください。

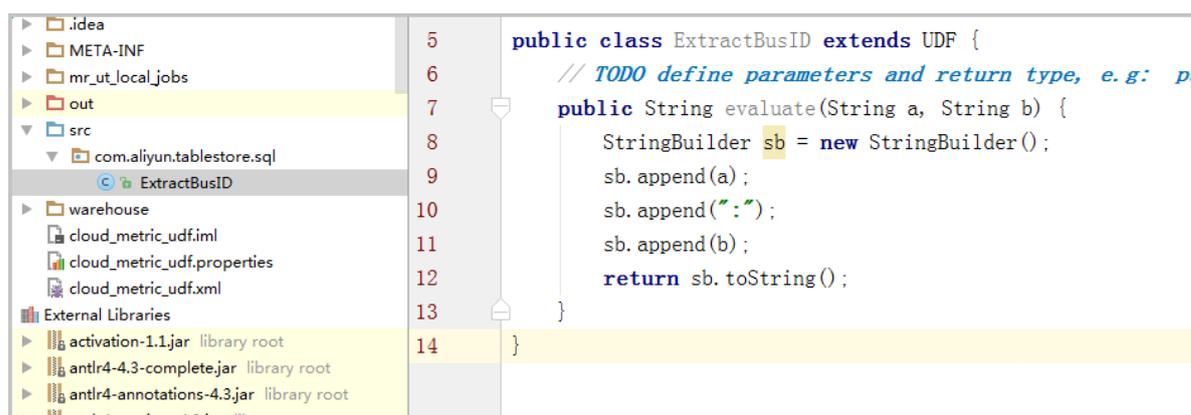
2.3 UDFを使用したデータ処理

Table Storeに格納されているデータが一意に構造化されていて、各行のデータを処理するための開発ロジックを定義したい場合(たとえば、特定のJSON文字列の解析)、ユーザー定義関数(UDF)を使用できます。

手順

1. [MaxCompute Studio](#)に従って、IntelliJにMaxCompute-Java / MaxCompute-Studioプラグインをインストールします。プラグインをインストールすると開発を開始できます。

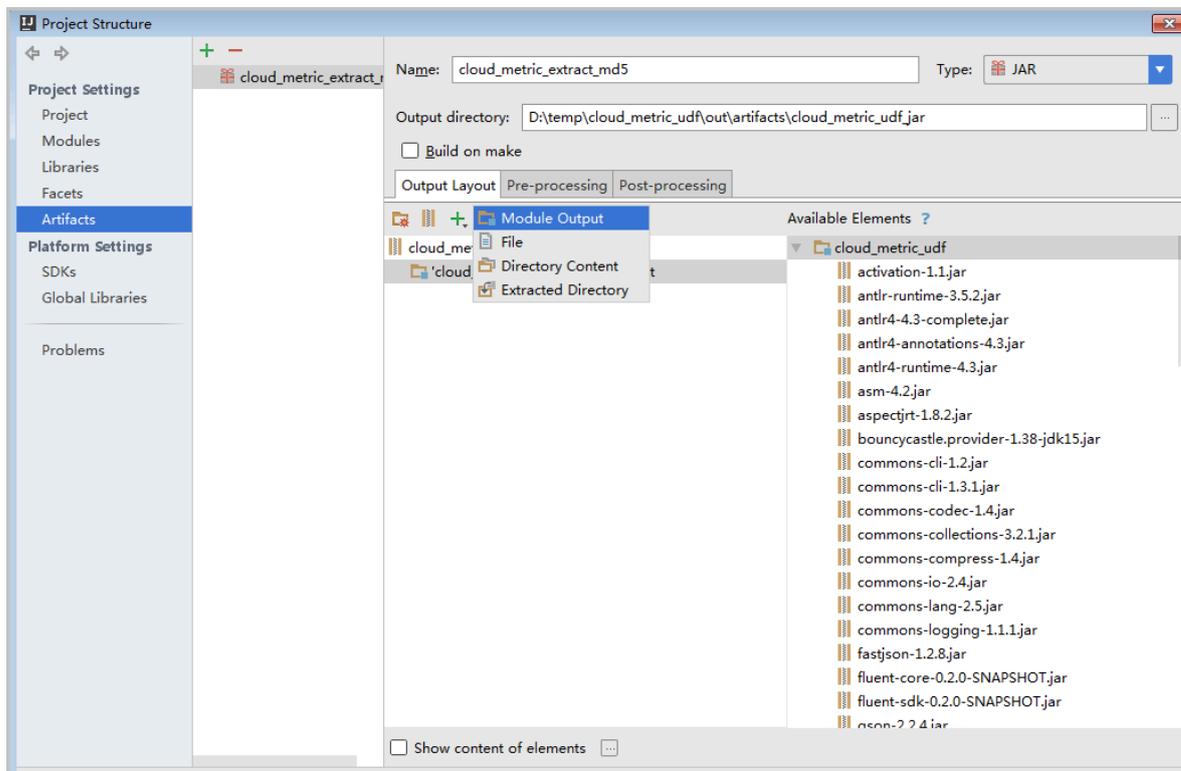
次の図は、2つの文字列を接続する単純なUDF定義を示しています。MaxComputeは、ユーザー定義のウィンドウ実行ロジックなど、より複雑なUDFをサポートします。詳細については、「[UDFの開発とデバッグ](#)」をご覧ください。



2. パッケージ化したら、リソースをMaxComputeにアップロードします。

[ファイル] > [プロジェクト構造] > [成果物] を選択します。名前と出力ディレクトリを入力し、アウトプットモジュールを選択するために[+]をクリックします。パッケージ化したら、

リソースをアップロードし、ODPS Project Explorer を使用して関数を作成したら、それを SQL で呼び出すことができます。



3. " bin / odpscmd . bat " を実行します。

```
// Select a line of data , and pass name / name into
the UDF . 2 つの文字列の連結が返されます。
select cloud_metric_extract_md5 ( name , name ) as
udf_test from test_table limit 1 ;
```

返された結果の例は、次のように表示されます。

```
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
2055324953gq1tsau1&token=d214cGJkSk9VRWlGQkNmNXZCV0J0ZWQ4T21zPSxPRFBTX09CTzoxNDE0MDcwMjYwNjg3NzQ1LDE0ODkwMzg4MDUseyJ
-fjdG1vbiI6WyJvZHBzO1JlYWQiXSwiRWZmZWNOIjoiQWxsY3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnByb2p1Y3RzL3RhYmNlcy8yMDE3MDMwMjA1NTMyNDk1M2dxdXRzYXUxIl19XSwiVmVyc2l2b2I6IjEifQ==
Job Queueing...
QuotaCPUUsage: 99.99%   QuotaMemUsage: 79.36%
-----
          STAGES          STATUS  TOTAL  COMPLETED  RUNNING  PENDING  BACKUP
R1_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
```

2.4 一般的なエラーのトラブルシューティング

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

解決策: "set odps.task.major.version = unstructured_data" を設定します。

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

解決策: Table Store のエンドポイントが正しくないため、MaxCompute にアクセスできません。正しいエンドポイントを入力してください。

- ・ エラー: logview invalid end_point

解決策: ログビュー URL は実行中に返されます。ブラウザを使用してアドレスにアクセスしたときにエラーが返された場合は、設定が正しくないことが原因である可能性があります。

MaxCompute の設定を確認してください。

問題が解決しない場合は、チケットを起票し、[アフターサービスの技術サポートセンター](#)にお問い合わせください。

3 Hive と HadoopMR

3.1 環境の準備

Hive と HadoopMR を使用してテーブルストアテーブルにアクセスする

Table Store および **E-MapReduce** がリリースした依存関係パッケージを利用することで、Hive と HadoopMR を使用して Table Store 内のデータに直接アクセスすることができます。

Table Store および E-MapReduce がリリースした依存関係パッケージを利用することで、Hive と HadoopMR を使用して Table Store 内のデータに直接アクセスすることができます。

JDK-7+ のインストール

1. JDK-7+ の関連インストールパッケージをダウンロードしてインストールします。
 - ・ Linux /MacOS は、パッケージインストールマネージャーを使用してください。
 - ・ Windows は、[クリックしてダウンロード](#)してください。
2. 以下のようにインストール状況を確認してください。

```
$ 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)
```

Hadoop のインストール

1. **Hadoop** をダウンロード (バージョン 2.6.0 以降)
2. インストールパッケージを解凍し、クラスターに Hadoop をインストールします。
3. 以下のように Hadoop を実行します。

```
$ bin / start - all . sh
# Check the hadoop service
$ jps
24017 NameNode
24835 Jps
24131 DataNode
24438 ResourceMa nager
5114 HMaster
24287 SecondaryN ameNode
24527 NodeManage r
```

4. Hadoop のパスを `/ etc / profile` に追加します。設定を有効にするために、`" source / etc / profile "` を実行します。

```
export HADOOP_HOME=/data/hadoop/hadoop-2.6.0
export PATH=$PATH:$HADOOP_HOME/bin
```

Hive のインストール

1. `bin.tar.gz` タイプの [Hive](#) をダウンロードします。
2. 次のようにインストールパッケージを解凍します。

```
$ 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. 次のようにスキーマを初期化します。

```
# Enter the specified directory
$ cd /home/admin/hive-2.1.0/

# Initialization, Derby can be replaced directly
with mysql if it is MySQL
# If an error occurs, you can delete it by
running rm -rf metastore_db/ and execute again.
$ ./bin/schematool -initSchema -dbType derby
```

4. 次のように Hive を実行します。

```
$ ./bin/hive
# check hive
hive> show databases ;
OK
default
Time taken : 0.207 seconds , Fetched : 1 row(s)
```

Table Store 用の Java SDK のダウンロード

1. [Java SDK](#) 依存関係パッケージ (バージョン 4.1.0 以降) をダウンロードします。



注:

SDK 依存関係パッケージは [Java SDK](#) で更新されます。最新の Java SDK に従って依存パッケージをダウンロードしてください。

2. 次のように SDK を Hive ディレクトリにコピーします。

```
$ mv tablestore - 4 . 1 . 0 - jar - with - dependenci es . jar
/ home / admin / hive - 2 . 1 . 0 /
```

EMR 依存パッケージのダウンロード

1. [Alibaba Cloud EMR 依存関係パッケージ](#)をダウンロードします。



注：

EMR の詳細については、[こちら](#)をクリックしてください。

2. `emr - sdk_2 . 10 - 1 . 3 . 0 - 20161025 . 065936 - 1 . jar` ファイルの名前を変更します。

```
mv emr - sdk_2 . 10 - 1 . 3 . 0 - 20161025 . 065936 - 1 . jar /
home / admin / hive - 2 . 1 . 0 / emr - sdk_2 . 10 - 1 . 3 . 0 -
SNAPSHOT . jar
```

3.2 チュートリアル

データ準備

Table Store 内のテーブルに "pet" という名前を付け、次のデータをインポートします。名前列が唯一のプライマリーキーです。

名前	管理者	種類	性別	誕生日	死亡日
Fluffy	Harold	ネコ	メス	1993-02-04	
Claws	Gwen	ネコ	オス	1994-03-17	
Buffy	Harold	イヌ	メス	1989-05-13	
Fang	Benny	イヌ	オス	1990-08-27	
Bowser	Diane	イヌ	オス	1979-08-31	1995-07-29
Chirpy	Gwen	トリ	メス	1998-09-11	
Whistler	Gwen	トリ		1997-12-09	
Slim	Benny	ヘビ	オス	1996-04-29	
Puffball	Diane	ハムスター	メス	1999-03-30	



注：

([データモデル](#)トピックに従って) Table Store はスキーマフリーなので、空白のセルには何も入力する必要はありません (`NULL` など)。

Hive によるアクセス例

準備

Hadoop、Hive、JDK のための環境と Table Store SDK および EMR の依存関係パッケージを**必須条件**として準備します。

例

```
# HADOOP_HOME と HADOOP_CLASSPATH を / etc / profile に追加できます。
$ export HADOOP_HOME=${Your Hadoop Path}
$ export HADOOP_CLASSPATH=emr-tablestore-1.4.2.jar:tablestore-4.3.1-jar-with-dependencies.jar:jodatime-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 =" YourAccess KeyId ",
    " tablestore . access_key _secret =" YourAccess KeySecret ",
    " 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 )
```

パラメータ説明

- WITH SERDEPROPERTIES

tablestore.columns.mapping (オプション): デフォルトでは、外部テーブルのフィールド名 (Hiveの規約に従って小文字で表記) は、Table Store の列名 (プライマリキーまたは属性

列の名前)と同じです。ただし、大文字と小文字が区別されるか文字セットが原因で、名前が異なる場合があります。この場合は、`tablestore.columns.mapping`を指定する必要があります。このパラメータはカンマ区切りの文字列です。コンマの前後に空白を追加することはできません。各項目は列名であり、順序は外部表のフィールド名と同じです。



注:

Table Store は空白文字を含む列名をサポートします。つまり、空白は列名の一部と見なされます。

・ TBLPROPERTIES

- `tablestore.endpoint` (必須): [エンドポイント](#)です。Table Store コンソールでインスタンスのエンドポイント情報を表示できます。
- `tablestore.instance` (オプション): [インスタンス名](#)です。指定されていない場合は、`tablestore.endpoint`の最初のフィールドです。
- `tablestore.table.name` (必須): Table Store 内のテーブル名。
- `tablestore.access_key_id`、`tablestore.access_key_secret` (必須): 「[アクセス制御](#)」をご参照ください。
- `tablestore.sts_token` (オプション): 「[セキュリティトークン](#)」をご参照ください。

HadoopMR によるアクセス例

次の例は、HadoopMR を使用して pet 内の行を数える方法を示しています。

コード例

・ マッパーとリデューサーの構築

```
public class RowCounter {
    public static class RowCounter Mapper
    extends Mapper < PrimaryKey Writable , RowWritabl e , Text ,
    LongWritab le > {
        private final static Text agg = new Text (" TOTAL
    ");
        private final static LongWritab le one = new
    LongWritab le ( 1 );

        @ Override
        public void map (
            PrimaryKey Writable key , RowWritabl e value ,
    Context context )
            throws IOExceptio n , Interrupte dException {
            context . write ( agg , one );
        }
    }

    public static class IntSumRedu cer
    extends Reducer < Text , LongWritab le , Text , LongWritab le > {
```

```

    @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 )) ;
    }
}
}

```

HadoopMR は pet から行を取得するたびに、マッパーの map() を呼び出します。最初の2つのパラメーター PrimaryKeyWritable と RowWritable は、それぞれ行のプライマリキーとこの行の内容に対応しています。PrimaryKeyWritable.getPrimaryKey() および RowWritable.getRow() を呼び出すことで、Table Store JAVA SDK によって定義されたプライマリキーオブジェクトと行オブジェクトを取得できます。

- マッパーのデータソースとしてテーブルストアの設定

```

    private static RangeRowQueryCriteria fetchCriteria
    () {
        RangeRowQueryCriteria res = new RangeRowQuery
        Criteria (" YourTableName ");
        res . setMaxVersions ( 1 );
        List < PrimaryKey Column > lower = new ArrayList <
        PrimaryKey Column > ();
        List < PrimaryKey Column > upper = new ArrayList <
        PrimaryKey Column > ();
        lower . add ( new PrimaryKey Column (" YourPkeyName
        ", PrimaryKey Value . INF_MIN ));
        upper . add ( new PrimaryKey Column (" 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 ( RowCounter Mapper . class );
        job . setCombinerClass ( IntSumReducer . class );
        job . setReducerClass ( IntSumReducer . class );
        job . setOutputKeyClass ( Text . class );
        job . setOutputValueClass ( LongWritable . class );
        job . setInputFormatClass ( TableStore InputFormat .
        class );
        TableStore InputFormat . setEndPoint ( job , " https
        :// YourInstance . Region . ots . aliyuncs . com /");
    }
}

```

```

        TableStore InputFormat . setCredential ( job , "
YourAccess KeyId ", " YourAccess KeySecret ");
        TableStore InputFormat . addCriteria ( job ,
fetchCriteria ());
        FileOutput Format . setOutputP ath ( job , new Path ("
output "));
        System . exit ( job . waitForCom pletion ( true ) ? 0 :
1 );
    }

```

上記の例では、`job.setInputFormatClass(TableStoreInputFormat.class)` を使用して Table Store をデータソースとして設定しています。例を完了するには、次の手順も必要です。

- `hadoop-connector.jar` をクラスタに展開して、それをクラスパスに追加します。
`hadoop-connector.jar` のローカルパスは `addFileToClassPath()` で指定されています。
コード例では、`hadoop-connector.jar` が現在のパスにあると仮定しています。
- Table Store にアクセスするときにエンドポイントとアクセスキーを指定します。エンドポイントとアクセスキーは `TableStoreInputFormat.setEndpoint()` と `TableStoreInputFormat.setCredential()` を使用して設定できます。
- カウントするテーブルを指定してください。



注:

- `TableStoreInputFormat.addCriteria()` は複数呼び出すことができます。呼び出しごとに `RangeRowQueryCriteria` オブジェクトが追加されます。
- `setFilter()` および `addColumnstoGet()` を設定して、サーバー側で不要な行と列をフィルタリングし、コストを削減し、Table Store のパフォーマンスを向上させます。
- それらをマージするために `RangeRowQueryCriteria`s を複数のテーブルに追加します。
- 分割を調整するために、1つのテーブルに複数の `RangeRowQueryCriteria`s を追加します。TableStore-Hadoop Connector は、指定された要件に基づいてユーザーの入力範囲を分割できます。

Host プログラムの実行

```

$ 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

```

データ型変換

Table Store と Hive/Spark は異なるデータ型のセットをサポートします。

次の表は、Table Store (行) から Hive (列) へのデータ型変換のサポートを示しています。

	TINYINT	SMALLINT	INT	BIGINT	FLOAT	DOUBLE	BOOLEAN	STRING	BINARY
INTEGER	はい (精度は限られます)	はい (精度は限られます)	はい (精度は限られます)	はい	はい (精度は限られます)	はい (精度は限られます)			
DOUBLE	はい (精度は限られます)	はい							
BOOLEAN							はい		
STRING								はい	
BINARY									はい

4 Spark と SparkSQL

4.1 環境の準備

Spark または Spark SQL で Table Store テーブルにアクセスする

Spark と Spark SQL を使用して、Table Store 内のデータに [Table Store](#) と [E-MapReduce](#) によってリリースされた依存関係パッケージを使用して直接アクセスすることができます。

Spark/Spark SQLをインストール

1. 次の要件に準拠する [Spark インストールパッケージ](#) をダウンロードします。

- ・ リリースバージョン: 1.6.2
- ・ パッケージタイプ: Hadoop 2.6 用にビルド済み
- ・ ダウンロードタイプ: 直接ダウンロード

2. 次のようにインストールパッケージを解凍します。

```
$ cd / home / admin / spark - 1 . 6 . 2
$ tar - zxvf spark - 1 . 6 . 2 - bin - hadoop2 . 6 . tgz
```

JDK-7+ のインストール

1. JDK-7+ のインストールパッケージをダウンロードしてインストールします。

- ・ Linux/MacOS: パッケージインストーラマネージャを使用してください。
- ・ Windows: [クリックしてダウンロード](#) します。

2. 以下のようにインストール状況を確認してください。

```
$ java - version
java version " 1 . 8 . 0_77 "
Java ( TM ) SE Runtime Environmen t ( build 1 . 8 . 0_77 -
b03 )
Java HotSpot ( TM ) 64 - Bit Server VM ( build 25 . 77 -
b03 , mixed mode )
```

Table Store 用の Java SDK のダウンロード

1. [Java SDK 依存関係パッケージ](#) (バージョン 4.1.0 以上) をダウンロードしてください。



注:

SDK 依存関係パッケージは [Java SDK](#) とともに更新されます。最新の Java SDK に従って依存パッケージをダウンロードしてください。

2. 次のように SDK を Spark ディレクトリにコピーします。

```
$ mv tablestore - 4 . 1 . 0 - jar - with - dependencies . jar
/ home / admin / spark - 1 . 6 . 2 /
```

EMR 依存パッケージのダウンロード

- ・ [Alibaba Cloud EMR 依存関係パッケージ](#)をダウンロードします。



注:

MNS の詳細については、[こちら](#)をクリックしてください。

- ・ `emr - sdk_2 . 10 - 1 . 3 . 0 - 20161025 . 065936 - 1 . jar` ファイルの名前を変更します。

```
mv emr - sdk_2 . 10 - 1 . 3 . 0 - 20161025 . 065936 - 1 . jar /
home / admin / spark - 1 . 6 . 2 / emr - sdk_2 . 10 - 1 . 3 . 0 -
SNAPSHOT . jar
```

Spark SQLの実行

```
$ 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 チュートリアル

データ準備

Table Store 内のテーブルに "pet" という名前を付け、データをインポートします。名前 列が唯一のプライマリーキーです。

名前	オーナー	種類	性別	誕生日	死亡日
Fluffy	Harold	ネコ	メス	1993-02-04	
Claws	Gwen	ネコ	オス	1994-03-17	
Buffy	Harold	イヌ	メス	1989-05-13	
Fang	Benny	イヌ	オス	1990-08-27	
Bowser	Diane	イヌ	オス	1979-08-31	1995-07-29
Chirpy	Gwen	トリ	メス	1998-09-11	

名前	オーナー	種類	性別	誕生日	死亡日
Whistler	Gwen	トリ		1997-12-09	
Slim	Benny	ヘビ	オス	1996-04-29	
Puffball	Diane	ハムスター	メス	1999-03-30	



注:

データモデルに従うと Table Store はスキーマフリーなので、空白のセルには (NULL など) 何も入れる必要はありません。

Spark SQL によるアクセスの例

準備

前提条件として、Spark、JDK の環境および Table Store SDK と EMR の依存関係パッケージを準備します。

例

```
$ bin / spark - sql -- master local -- jars tablestore - 4 . 3
. 1 - jar - with - dependenci es . 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 . openservic es . tablestore . hive .
TableStore StorageHan dler '
WITH SERDEPROPE RTIES (
" tablestore . columns . mapping =" name , owner , species , sex
, birth , death ")
TBLPROPERT IES (
" tablestore . endpoint =" YourEndpoi nt " ,
" tablestore . access_key _id =" YourAccess KeyId " ,
" tablestore . access_key _secret =" YourAccess KeySecret " ,
" 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 )

```

パラメーターの説明

・ WITH SERDEPROPERTIES

- `tablestore.columns.mapping` (オプション): デフォルトでは、外部テーブルのフィールド名 (Hive のルールに従って小文字で表記) は、Table Store の列名 (プライマリー列または属性列の名前) と同じです。ただし、大文字と小文字が区別されるか文字セットが原因で、名前が異なる場合があります。この場合は、`tablestore.columns.mapping` を指定する必要があります。このパラメーターはカンマ区切りの文字列です。コンマの前後に空白を追加することはできません。各項目は列名であり、順序は外部表のフィールド名と同じです。



注:

Table Store は、空白文字を含む列名をサポートします。したがって、空白スペースは列名の一部と見なされます。

・ TBLPROPERTIES

- `tablestore.endpoint` (必須): **エンドポイント**です。Table Store コンソールでインスタンスのエンドポイント情報を表示できます。
- `tablestore.instance` (オプション): **インスタンス名**です。指定されていない場合は、`tablestore.endpoint` の最初のフィールドです。
- `tablestore.table.name` (必須): Table Store 内のテーブル名。
- `tablestore.access_key_id` および `tablestore.access_key_secret` (必須): 「**アクセス制御**」をご参照ください。
- `tablestore.sts_token` (オプション): 「**セキュリティトークン**」をご参照ください。

Spark によるアクセスの例

次の例では、`pet` 内の行数を Spark によって数える方法を示します。

```

private static RangeRowQueryCriteria fetchCriteria () {
    RangeRowQueryCriteria res = new RangeRowQueryCriteria (
        "YourTableName");
    res.setMaxVersions ( 1 );
    List < PrimaryKey Column > lower = new ArrayList <
        PrimaryKey Column > ();
    List < PrimaryKey Column > upper = new ArrayList <
        PrimaryKey Column > ();
    lower.add ( new PrimaryKey Column (" YourPKeyName ",
        PrimaryKey Value . INF_MIN ));
}

```

```
upper . add ( new PrimaryKey Column ( " YourPkeyNa me ",
PrimaryKey Value . INF_MAX ));
res . setInclusi veStartPri maryKey ( new PrimaryKey ( lower
));
res . setExclusi veEndPrima ryKey ( new PrimaryKey ( upper
));
return res ;
}

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

    Configurati on hadoopConf = new Configurati on ();
    TableStore InputForma t . setCredent ial (
        hadoopConf ,
        new Credential ( " YourAccess KeyId ", " YourAccess
KeySecret " ));
    TableStore InputForma t . setEndpoi nt (
        hadoopConf ,
        new Endpoint ( " https :// YourInstan ce . Region . ots .
aliyuncs . com /" ));
    TableStore InputForma t . addCriteri a ( hadoopConf ,
fetchCrite ria ());

    try {
        JavaPairRDD < PrimaryKey Writable , RowWritabl e > rdd
= sc . newAPIHado opRDD (
            hadoopConf ,
            TableStore InputForma t . class ,
            PrimaryKey Writable . class ,
            RowWritabl e . class );
        System . out . println (
            new Formatter (). format ( " TOTAL : % d ", rdd . count
()). toString ());
    } finally {
        sc . close ();
    }
}
```



注:

scala を使用する場合は、JavaSparkContext を SparkContext に置き換え、JavaPairRDD を PairRDD に置き換えます。

プログラムの実行

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

データ型変換

Table Store と Hive/Spark は異なる種類のデータ型をサポートします。

次の表は、Table Store (行) から Hive (列) へのデータ型変換のサポートを示しています。

	TINYINT	SMALLINT	INT	BIGINT	FLOAT	DOUBLE	BOOLEAN	STRING	BINARY
INTEGER	はい (精度は限られます)	はい (精度は限られます)	はい (精度は限られます)	はい	はい (精度は限られます)	はい (精度は限られます)			
DOUBLE	はい (精度は限られます)	はい							
BOOLEAN							はい		
STRING								はい	
BINARY									はい