MaxCompute

Quick Start

Quick Start

After the user has been added into a project and granted with corresponding privileges, next he can oeprate MaxCompute. As the operation objects of MaxCompute (input and output) are tables, we must create tables and partitions before processing data.

Create Table

Command format:

```
CREATE TABLE [IF NOT EXISTS] table_name
[(col_name data_type [COMMENT col_comment], ...)]
[COMMENT table_comment]
[PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
[LIFECYCLE days]
[AS select_statement]

CREATE TABLE [IF NOT EXISTS] table_name
LIKE existing_table_name
```

Description:

- The table name and column name are both case insensitive.
- Exception will be thrown if a same name table has already existed. User should specify the option [if not exists] to skip the error. If the option [if not exists] is specified, no matter whether there is a same name table, even if the source table structure and the target table structure are inconsistent, all return success. The Meta information of existing table will not change.
- The data types will only support: bigint, double, boolean, datetime and string.
- The table name and column name cannot have special characters. It can only begin with a letter and include a-z, A-Z, digits and underline_. The name length cannot exceed 128 bytes.
- Use ‘Partitioned by’ to specify the partition and at present only string and bigint are supported. The partition value can not have a double byte characters (such as Chinese), must begin with a letter a-z or A-Z, followed by letter or numbe. The name length cannot exceed 128 bytes. Allowed characters including: space ‘ ’, colon ‘:’, underlined symbol ‘_’, ’$’,’#’, point ‘.’, exclamation point ‘!’ and’ ‘@’. Other characters are taken as undefined characters, such as ‘\t’, ‘\n’, ‘/’ and so on. If using partition fields in partition table, a full table scan is no need when adding partition, updating data in partition and readiy.

- The comment content is the effective string which length does not exceed 1024 bytes.
- Lifecycle indicates the lifecycle of the table. Unit is ‘day’.The statement ‘create table like’ will not copy the lifecycle attribute from source table.
- At present, the partition hierarchy cannot exceed 6 levels. The maximum partition number of a table can be configured in a certain project. The default maximum number is 60000.

Notes:Tables in MaxCompute support partition and lifecycle. For more details of creating table, please refer to CREATE TABLE. For partition operation, please refer to Add/Remove Partition. For lifecycle operation, please refer to Modify Lifecycle for a Table.

An example to create table:

```
create table test1 (key string); -- create a no-partition table.
create table test2 (key bigint) partitioned by (pt string, ds string); --Create a partition table.
create table test3 (key boolean) partitioned by (pt string, ds string) lifecycle 100; -- Create a table with lifecycle.

create table test4 like test3; -- Except the lifecycle properity, other properties of test3 (field type, partition type)
were completely consistent with test4.

create table test5 as select * from test2;
-- This operation will create test5, but the partition and lifecycle information will not be copied to the object table.
-- This operation will copy the data of test2 to the table test5.
```

Here we introduce an instance to create a table: suppose that we need to create a table named user, which includes the following information:

- user_id: bigint, user identifier, to identify a user.
- gender: bigint type, sex (0, unknown; 1, male; 2, female)
- age: bigint, the age of user

It should be partitioned by region and dt and the lifycycle is 365 days.The sentence to create this table is shown as follows:

```
CREATE TABLE user (
user_id BIGINT, gender BIGINT COMMENT '0 unknow,1 male, 2 Female', age BIGINT)
PARTITIONED BY (region string, dt string) LIFECYCLE 365;
```

Describer Table

```
desc <table_name>;
```

For example, get information from test3:

```
desc test3;
```

Get information from test4:

```
desc test4;
```

Except the lifecycle properity, other properties of test3 (field type, partition type) were completely consistent with test4. For more introductions of describing table, please refer to Describe Table.

If the user is to view the information of test2, the two fields ‘pt’, ‘ds’ will only exist as two common columns, rather than the table partitions.

Drop Table

```
DROP TABLE [IF EXISTS] table_name;
```

For example, delete the table ‘test2’:

```
drop table test2;
```

For more introductions, please refer to DROP TABLE.

Add Partition

After we create a partition table and need to import data into different partitions, we should create the partition. The statement format is shown as follows:

```
    alter table table_name add [if not exists] partition partition_spec

partition_spec:

: (partition_col1 = partition_col_value1, partition_col2 = partiton_col_value2, ...)
```

As shown in last example, we need to add the partitions (region is ‘hangzhou’ and dt is ‘20150923’) for the table user. The statement is shown as follows:

```
Alter table user add if not exists partition(region='hangzhou',dt='20150923');
```

Drop Partition

Statement Format:

```
  alter table table_name drop [if exists] partition_spec;

partition_spec:

: (partition_col1 = partition_col_value1, partition_col2 = partiton_col_value2, ...)
```

For example, we must delete the partitions (region is 'hangzhou' and dt is '20150923'). The statement is shown as follows:

```
Alter table user drop if exists partition(region='hangzhou',dt='20150923');
```

MaxCompute provides two data import and export methods: using Tunnel Operation on the console directly or using TUNNEL written with java.

Tunnel Commands

Data Preparation

Suppose that we hava prepared the local file wc_example.txt and its corresponding contents are shown as follows:

```
I LOVE CHINA!
MY NAME IS MAGGIE.I LIVE IN HANGZHOU!I LIKE PLAYING BASKETBALL!
```

Here we save the file into the directory: D:\odps\odps\bin.

Create a ODPS Table

As we need to import the data mentioned above into a MaxCompute table, here we need to create a table at first:

```
CREATE TABLE wc_in (word string);
```

Execute Tunnel Command

After the input table is created successfully, you can import the data on MaxCompute console through tunnel command, as follows:

```
tunnel upload D:\odps\odps\bin\wc_example.txt wc_in;
```

After the running is successful, check the records in the table wc_in, as follows:

```
odps@ $odps_project>select * from wc_in;

ID = 20150918110501864g5z9c6
Log view:
http://webconsole.odps.aliyun-inc.com:8080/logview/?h=http://service-corp.odps.aliyun-
inc.com/api&p=odps_public_dev&i=20150918
QWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnByb2plY3RzL29kcHNfcHVibGljX2Rldi9pbnN0YW5jZXMvMjAxN
TA5MTgxMTA1MDE4NjRnNXo5YzYiXX1dLC
+------+
| word |
+------+
| I LOVE CHINA! |
| MY NAME IS MAGGIE.I LIVE IN HANGZHOU!I LIKE PLAYING BASKETBALL! |
+------+
```

Note:

- For more details of Tunnel commands, please refer to Tunnel Operation.

Tunnel SDK

On how to use SDK tunnel to upload data, the following simple scenario will be introduced.Scenario description: Upload data into ODPS, where the project is “odps_public_dev”, the table name is “tunnel_sample_test” and the partitions are ” pt=20150801,dt=” hangzhou”.

create a table and add corresponding partitions:

```
CREATE TABLE IF NOT EXISTS tunnel_sample_test(
id STRING,
name STRING)
PARTITIONED BY (pt STRING, dt STRING); --Create a table.
ALTER TABLE tunnel_sample_test
ADD IF NOT EXISTS PARTITION (pt='20150801',dt='hangzhou'); --Add the partitions.
```

Create the program directory structure of UploadSample, as follows:

```
|---pom.xml
|---src
|---main
|---java
```

```
|---com
|---aliyun
|---odps
|---tunnel
|---example
|---UploadSample.java
```

UploadSample : tunnel source file.pom.xml : maven program file.

Write UploadSample program, as follows:

```
package com.aliyun.odps.tunnel.example;
import java.io.IOException;
import java.util.Date;
import com.aliyun.odps.Column;
import com.aliyun.odps.Odps;
import com.aliyun.odps.PartitionSpec;
import com.aliyun.odps.TableSchema;
import com.aliyun.odps.account.Account;
import com.aliyun.odps.account.AliyunAccount;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.RecordWriter;
import com.aliyun.odps.tunnel.TableTunnel;
import com.aliyun.odps.tunnel.TunnelException;
import com.aliyun.odps.tunnel.TableTunnel.UploadSession;
public class UploadSample {
private static String accessId = "####";
private static String accessKey = "####";
private static String tunnelUrl = "http://dt-corp.odps.aliyun-inc.com";

private static String odpsUrl = "http://service-corp.odps.aliyun-inc.com/api";

private static String project = "odps_public_dev";
private static String table = "tunnel_sample_test";
private static String partition = "pt=20150801,dt=hangzhou";

public static void main(String args[]) {
Account account = new AliyunAccount(accessId, accessKey);
Odps odps = new Odps(account);
odps.setEndpoint(odpsUrl);
odps.setDefaultProject(project);
try {
TableTunnel tunnel = new TableTunnel(odps);
tunnel.setEndpoint(tunnelUrl);
PartitionSpec partitionSpec = new PartitionSpec(partition);
UploadSession uploadSession = tunnel.createUploadSession(project,
table, partitionSpec);

System.out.println("Session Status is : "
+ uploadSession.getStatus().toString());

TableSchema schema = uploadSession.getSchema();
RecordWriter recordWriter = uploadSession.openRecordWriter(0);
Record record = uploadSession.newRecord();
for (int i = 0; i < schema.getColumns().size(); i++) {
Column column = schema.getColumn(i);
switch (column.getType()) {
case BIGINT:
record.setBigint(i, 1L);
break;
case BOOLEAN:
record.setBoolean(i, true);
break;
case DATETIME:
record.setDatetime(i, new Date());
break;
case DOUBLE:
record.setDouble(i, 0.0);
break;
case STRING:
record.setString(i, "sample");
break;
default:
throw new RuntimeException("Unknown column type: "
+ column.getType());
}
}
for (int i = 0; i < 10; i++) {
recordWriter.write(record);
}
recordWriter.close();
uploadSession.commit(new Long[]{0L});
System.out.println("upload success!");

} catch (TunnelException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
}
}
```

Note: here we ignored the configuration of accessId and accesskey. In actual operation, please change your own accessId and accessKey.

The configuration of pom.xml is shown as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.aliyun.odps.tunnel.example</groupId>
<artifactId>UploadSample</artifactId>
<version>1.0-SNAPSHOT</version>
<dependencies>
<dependency>
```

```
<groupId>com.aliyun.odps</groupId>
<artifactId>odps-sdk-core-internal</artifactId>
<version>0.20.7</version>
</dependency>
</dependencies>
<repositories>
<repository>
<id>alibaba</id>
<name>alibaba Repository</name>
<url>http://mvnrepo.alibaba-inc.com/nexus/content/groups/public/</url>
</repository>
</repositories>
</project>
```

Compile and run:Compile the program UploadSample:

```
mvn package
```

Run the program UploadSample. Here we use Eclipse to import maven project: right-click on the java program and click <Import->Maven->Existing Maven Projects> and the setting is shown as follows:Right-click on ‘UploadSample.java’ and click <Run As->Run Configurations>, as follows:Click <Run>. After running successfully, the console shows as follows:

```
Session Status is : NORMAL
upload success!
```

Check running result:Input the following sentence on the console:

```
select * from tunnel_sample_test;
```

The result is shown as follows:

```
+----+------+----+----+
| id | name | pt | dt |
+----+------+----+----+
| sample | sample | 20150801 | hangzhou |
| sample | sample | 20150801 | hangzhou |
| sample | sample | 20150801 | hangzhou |
| sample | sample | 20150801 | hangzhou |
| sample | sample | 20150801 | hangzhou |
| sample | sample | 20150801 | hangzhou |
| sample | sample | 20150801 | hangzhou |
| sample | sample | 20150801 | hangzhou |
| sample | sample | 20150801 | hangzhou |
| sample | sample | 20150801 | hangzhou |
+----+------+----+----+
```

Notes:

- As an independent service in MaxCompute, Tunnel has exclusive access port provided for users. At the same time, there are multiple Tunnel services deployed for internal MaxCompute service, to meet the different needs of the production segment and pressure. To prevent the confusion caused by multiple sets of access addresses, MaxCompute provides the routing function of access ports. The detailed access policies are showns as follows:
- If you access Tunnel service from the production network, you just need to specify MaxCompute EndPoint(http://service.odps.aliyun-inc.com/api). You do not need to configure Tunnel EndPoint, which supports automatic routing accoring to MaxCompute endpoint.
- If you access Tunnel service from office network, you just need to specify MaxCompute EndPoint (http://service-corp.odps.aliyun-inc.com/api). You do not need to configure Tunnel EndPoint, which supports automatic routing accoring to MaxCompute endpoint.

Fluentd Import Scheme

In addition to MaxCompute Console and Java SDK, data can also be imported through Fluentd.

Fluentd is an open source software, used to collect a variety of source logs (including Log Application, Log Access and Log Sys). It allows user to select corresponding plug-in to filter the log data and save the data into different data processing clients, including MySQL, Oracle, MongoDB, Hadoop, Treasure Data, AWS Services, Google Services and ODPS. Fluentd is known for its compact and flexible, allowing users to customize data sources, filter processing andtarget terminals.Currently in this software, there are 300+ plug-ins runing on the Fluentd architecture, and these plug-ins are all open source. MaxCompute also opened data imported plug-in in this software.

Equirement Preparation

In order to import data into MaxCompute through this software, we need prepare the following environments:

- Ruby 2.1.0 or updated
- Gem 2.4.5 or updated
- Fluentd-0.10.49 or check the latest version from Fluentd Official Website. Fluentd provides different versions for different OS. For the details, refer to Fluentd Articles.
- Protobuf-3.5.1 or updated (Ruby protobuf)

Install Import Plug-in

Next you can install MaxCompute Fluentd import plug-in by any of the following two ways.Method 1:

install it through ruby gem.

```
$ gem install fluent-plugin-aliyun-odps
```

ODPS has released this plug-in into GEM, which name is ‘fluent-plugin-aliyun-odps’. You just need to install it through ‘gem install’ command. (During using gem course, you may encounter that gem can not be accessed. Now you can search ‘change gem sourse’ from internet to solve this problem.)

Method 2: install it through the pulg-in source code

```
$ gem install protobuf
$ gem install fluentd --no-ri --no-rdoc
$ git clone https://github.com/aliyun/aliyun-odps-fluentd-plugin.git
$ cp aliyun-odps-fluentd-plugin/lib/fluent/plugin/* {YOUR_FLUENTD_DIRECTORY}/lib/fluent/plugin/ -r
```

In the commands above, the second command is used to install fluentd. If you have already installed it, you can ignore this command. The source code of MaxCompute Fluentd plug-in is located on github. After clone, put it into ‘plugin’ directory of Fluentd.

Use Plug-in

To import data using Fluentd, the most important is to configure the ‘conf’ file of Fluentd. For more details of conf file, refer to Fluentd Configuration File Introduction.

Example 1: import Nginx log. The configuration in ‘source’ of Conf are shown as follows:

```
<source>
type tail
path /opt/log/in/in.log
pos_file /opt/log/in/in.log.pos
refresh_interval 5s
tag in.log
format /^(?<remote>[^ ]*) - - \[(?<datetime>[^\]]*)\] "(?<method>\S+)(?: +(?<path>[^\"]*?)(?: +\S*)?)?"
(?<code>[^ ]*) (?<size>[^ ]*) "-" "(?<agent>[^\"]*)"$/
time_format %Y%b%d %H:%M:%S %z
</source>
```

Fluentd monitors whether the specified file content has changed by ‘tail’. For more tail configuration, refer to Fluentd Articles.The configuration of match, as follows:

```
<match in.**>
type aliyun_odps
aliyun_access_id ************
aliyun_access_key *********
aliyun_odps_endpoint http://service.odps.aliyun.com/api
aliyun_odps_hub_endpoint http://dh.odps.aliyun.com
```

```
buffer_chunk_limit 2m
buffer_queue_limit 128
flush_interval 5s
project projectforlog
<table in.log>
table nginx_log
fields remote,method,path,code,size,agent
partition ctime=${datetime.strftime('%Y%m%d')}
time_format %d/%b/%Y:%H:%M:%S %z
</table>
</match>
```

The data will be imported into the table nginx_log in the project ‘projectforlog’. The column ‘datatime’ in soruce data will be taken as the partition. The plug-in will create a partition automatically when meeting different values.

Example 2: import the data of MySqL. When importing the data in MySQL, we need install ‘fluent-plugin-sql’ as the data source.

```
$ gem install fluent-plugin-sql
```

Configure ‘source’ in ‘conf’:

```
<source>
type sql
host 127.0.0.1
database test
adapter mysql
username xxxx
password xxxx
select_interval 10s
select_limit 100
state_file /path/sql_state
<table>
table test_table
tag in.sql
update_column id
</table>
</source>
```

This example is to select data from test_table, read 100 records each 10 seconds. When selecting, the ID column is taken as the primary key (ID field is the self enhancement). For more descriptions of ‘fluent-plugin-sql’, refer to Fluentd SQL Plug-in Description.

The configuration of ‘match’ is shown as follows:

```
<match in.**>
type aliyun_odps
aliyun_access_id ************
aliyun_access_key *********
aliyun_odps_endpoint http://service.odps.aliyun.com/api
```

```
aliyun_odps_hub_endpoint http://dh.odps.aliyun.com
buffer_chunk_limit 2m
buffer_queue_limit 128
flush_interval 5s
project your_projectforlog
<table in.log>
table mysql_data
fields id,field1,field2,fields3
</table>
</match>
```

The data will be imported into the table ‘mysql_data’ in the project ‘projectforlog’. The imported fields include id, field1, field2 and field3.

Plug-in Parameter Description

To import data into MaxCompute, we need configure MaxCompute plug-in in the item ‘match’ in ‘conf’ file. The supported parameter descriptions are shown as follows:

- type(Fixed): fixed value, aliyun_odps.
- aliyun_access_id(Required): access_id.
- aliyun_access_key(Required): access key.
- aliyun_odps_hub_endpoint(Required): If your sever is deployed on ECS, set this value to be ‘http://dh-ext.odps.aliyun-inc.com’; Otherwise, set it to be ‘http://dh.odps.aliyun.com’.
- aliyunodps_endpoint(Required):If your server is deployed on ESC, set this value to be ‘http://odps-ext.aiyun-inc.com/api’; Otherwise, set it to be ‘http://service.odps.aliyun.com/api’.
- buffer_chunk_limit(Optional): bolock size, the unit supports “k”(KB), “m”(MB) and “g”(GB). The default value is 8MB. Its suggested value is 2MB.
- buffer_queue_limit(Optional): the size of block queue. This value together with ‘buffer_chunk_limit’ determine the size of entire buffer.
- flush_interval(Optional): Mandatory sending interval. After the time reached the threshold and block data has not be full, then send the message mandatory. The defalt value is 60s.
- project(Required): project name.
- table(Required): table name.
- fields(Required): corresponding to ‘source’. The field names must exist in ‘source’.
- partition(Optional)：if the table is partition table, set this item.
- Setting mode of partition name:
    - fixed value: partition ctime=20150804
    - keyword: partition ctime=${remote} (remote is a field in ‘source’.)
    - time format keyword: partition ctime=${datetime.strftime(‘%Y%m%d’)} (‘datetime’ is a time format field in ‘source’. The output format is %Y%m%d, as the partition name).
    - time_format(Optional): if using ‘time format keyword’ as < partition >, please set this parameter. For example, source[datetime]=”29/Aug/2015:11:10:16 +0800”,

then set < time_format >to be “%d/%b/%Y:%H:%M:%S %z” .

Flume

Besides using Fluentd to import dat, MaxCompute also supports importing data through Flume. Flume is an open source software of Apache. MaxCompute opened the source code of import plug-in based on the Flume. If you are interested in more details, refer to Flume MaxCompute Plug-in.

Since most users are already familiar with SQL syntax, some special notices will be mentioned here.

It is worth to mention that MaxCompute SQL does not support transactions, index and Update/Delete operations. MaxCompute SQL syntax differs from Oracle and MySQL, so the user cannot migrate SQL statements of other databases into MaxCompute seamlessly. In addition, MaxCompute SQL can not complete the query in minutes even seconds and unable to return the result in millisecond.

Select Statements

- The key of “group by” statement can be the column name of input table and also can be the expression consisted of input table columns, but it can not be output column of Select statements.

```
select substr(col2, 2) from tbl group by substr(col2, 2); -- Yes, the key of ‘group by’ can be the expression consisted of input table column;

select col2 from tbl group by substr(col2, 2); -- No, the key of ‘group by’ is not in the column of Select statement;

select substr(col2, 2) as c from tbl group by c; -- No, the key of ‘group by’ can not be the column alias, i.e., the output column of Select statement;
```

The reason for such a restriction: for usual SQL parsing, “group by” operations are conducted before “select” operations, therefore, “group by” can only use the column or expression of input table as the key.

- “Order by” statement must be used in combination with “limit” .
- “Distribute by” statement must be added in front of “sort by” .
- The key of “order by/sort by/distribute by” must be the output column of “select” statement, i.e., the column alias.

```
select col2 as c from tbl order by col2 limit 100 -- No, the key of ‘order by’ is not the output column (column alias) of Select statement.
select col2 from tbl order by col2 limit 100; -- Yes, use column name as the alases if the output column of Select statement has no alias.
```

The reason for such a restriction: for usual SQL parsing, order by / sort by / distribute by operations are conducted after “select” operations; therefore, they can only use the output column of select statements as the key.

Insert Statement

- To insert data into a specified partition, the partition column is not allowed in Select list:

```
    insert overwrite table sale_detail_insert partition (sale_date='2013', region='china')

select shop_name, customer_id, total_price, sale_date, region from sale_detail;

-- Return error; sale_date and region are partition columns, which are not allowed in Select statement in static partition.
```

- To insert a dynamic partition, the dynamic partition column must be in Select list:

```
    insert overwrite table sale_detail_dypart partition (sale_date='2013', region)

select shop_name,customer_id,total_price from sale_detail;

-- Failed, to insert the dynamic partition, the dynamic partition column must be in Select list.
```

Join

- MaxCompute SQL supports the following Join operaton types: {LEFT OUTER|RIGHT OUTER|FULL OUTER|INNER} JOIN.
- At present, MaxCompute SQL supports up to 16 concurrent Join operations.
- Support the mapjoin up to six small tables.

Others

- MaxCompute SQL currently supports up to 128 concurrent union operations;
- Support up to 128 concurrent insert overwrite/into operations.

Summary

MaxCompute UDF include: UDF, UDAF and UDTF. Usually these three kinds of functions are called by a joint name 'UDF'. Users who use Maven can search "odps-sdk-udf" from Maven Library to get Java SDK with different versions. The related configuration is shown as follows:

```
<dependency>
<groupId>com.aliyun.odps</groupId>
<artifactId>odps-sdk-udf</artifactId>
<version>0.20.7-public</version>
</dependency>
```

Notes:

- UDF currently only supports Java language. If you want to write UDF program, you can upload UDF code into the project through Add Resource and create UDF through Create Function.
- The code examples of UDF, UDAF and UDTF will be given seperately in this section. To run UDF, please refer to UDF Eclipse Plug-in Introduction.
- If you need to use UDF, you should submit application in workorder system, providing odps project name and application scenario. After the application is passed, you can use UDF.

UDF Example

An entire development example of UDF will be given below. For example, to develop the UDF to realize character lowercase conversion, go through the following steps:

- Coding: realize the UDF function in accordance with ODPS UDF framework and do compiling. Next is a simple coding instance:

```
package org.alidata.odps.udf.examples;

import com.aliyun.odps.udf.UDF;

public final class Lower extends UDF {

public String evaluate(String s) {

if (s == null) { return null; }
return s.toLowerCase();
}
}
```

- Nominate this jar package ‘my_lower.jar’.

Note:

- For the information of SDK, please refer to UDF SDK.

Add resource: Specifying the referenced UDF code is needed before running UDF. The user’ s code is added to ODPS in form of resource. Java UDF must be made into jar package and added in ODPS in form of jar resource. UDFframework will load jar package automatically and run UDF. MaxCompute MapReduce aslo describes the use of resource.

Execute the command:

```
add jar my_lower.jar;
-- If the resource name has existed, rename the jar package.
-- Pay attention to modifying related name of jar package in following command.
-- Or use –f option directly to overwrite original jar resource.
```

- Register UDF: MaxCompute is able to obtain user’ s code and run it after the jar package has been uploaded. But now this UDF can not be used because MaxCompute does not have any information about this UDF. It requires the user to register a unique function name in ODPS and specify which function corresponding to this function name with which jar resource. For registering UDF, please refer to Create Function. Run the command:

```
 CREATE FUNCTION test_lower AS org.alidata.odps.udf.examples.Lower USING my_lower.jar;
```

Use this function in sql:

```
select test_lower('A') from my_test_table;
```

UDAF Example

The register method of UDAF is similar to UDF. Its usage is also the same as Aggregation Function in Built-in function. Next is a UDAF code example to calculate the average:

```
package org.alidata.odps.udf.examples;

import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.Text;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.udf.Aggregator;
import com.aliyun.odps.udf.UDFException;
```

```
/**
* project: example_project
* table: wc_in2
* partitions: p2=1,p1=2
* columns: colc,colb,cola
*/

public class UDAFExample extends Aggregator {

@Override

public void iterate(Writable arg0, Writable[] arg1) throws UDFException {
LongWritable result = (LongWritable) arg0;
for (Writable item : arg1) {
Text txt = (Text) item;
result.set(result.get() + txt.getLength());
}
}

@Override

public void merge(Writable arg0, Writable arg1) throws UDFException {
LongWritable result = (LongWritable) arg0;
LongWritable partial = (LongWritable) arg1;
result.set(result.get() + partial.get());
}

@Override

public Writable newBuffer() {
return new LongWritable(0L);
}

@Override

public Writable terminate(Writable arg0) throws UDFException {
return arg0;
}
}
```

UDTF Example

The register method of UDTF is similar to UDF. Its usage is the same as UDF. The code example is shown as follows:

```
package org.alidata.odps.udtf.examples;

import com.aliyun.odps.udf.UDTF;
import com.aliyun.odps.udf.UDTFCollector;
import com.aliyun.odps.udf.annotation.Resolve;
import com.aliyun.odps.udf.UDFException;
```

```
// TODO define input and output types, e.g., "string,string->string,bigint".

@Resolve({"string,bigint->string,bigint"})

public class MyUDTF extends UDTF {

@Override

public void process(Object[] args) throws UDFException {
String a = (String) args[0];
Long b = (Long) args[1];
for (String t: a.split("\\s+")) {
forward(t, b);
}
}
}
```

This section is to introduce how to run the example program ‘MapReduce WordCount’ rapidly after the MaxCompute console has already been installed. The users who use Maven can search “odps-sdk-mapred” from Maven Library to get different versions of Java SDK. The configuration is shows as follows:

```
<dependency>
<groupId>com.aliyun.odps</groupId>
<artifactId>odps-sdk-mapred</artifactId>
<version>0.20.7</version>
</dependency>
```

Notes:

- To compile and run MapReduce requires JDK1.6.
- For installing ODPS console quickly, refer to Quick Start. For the use method of ODPS client, please refer to Console;

Next we will introduce the operation step by step.

1.Create input and output tables and upload the data. For the SQL statement to create table, refer to CREATE TABLE :

```
CREATE TABLE wc_in (key STRING, value STRING);

CREATE TABLE wc_out (key STRING, cnt BIGINT);

-- Create input table and output table
```

2.Use Tunnel Commands to upload data:

```
tunnel u kv.txt wc_in

-- Upload example data
```

The data in kv.txt is shown as follows:

```
238,val_238

186,val_86

186,val_86
```

You can also insert data directly by SQL statement as follows:

```
insert into table wc_in select '238',' val_238' from (select count(*) from wc_in) a;
```

3.Write MapReduce program and compile it.

ODPS provides a convenient Eclipse development plug-in for the user, to facilitate the users to develop MapReduce program quickly and provides local debugging MapReduce function.

User needs to create an MaxCompute project in Eclipse and then write MapReduce program. After loacl debugging is passed, upload the compiled program to ODPS. For more details, please refer to MapReduce Eclipse Plug-in.

4.Add .jar package into the project. (for example, the name of jar package is "word-count-1.0.jar"):

```
add jar word-count-1.0.jar;
```

5.Run "-jar" command on MaxCompute console:

```
jar -resources word-count-1.0.jar -classpath /home/resources/word-count-1.0.jar
com.taobao.jingfan.WordCount wc_in wc_out;
```

6.Check the running result on ODPS console:

```
select * from wc_out;
```

Note:

- If other resources are used in java program, make sure to add '-resources' parameters. For more details about jar commands, please refer to Jar Commands.

The sumbitting method of Graph job is similar to MapReduce. Next, take SSSP Algorithm as an example to illustrate how to submit Graph job. Users who use Maven can search "odps-sdk-graph" from Maven Library to get different versions of Java SDK. The related configuraion information:

```
<dependency>
<groupId>com.aliyun.odps</groupId>
<artifactId>odps-sdk-graph</artifactId>
<version>0.20.7</version>
</dependency>
```

Next we will take SSSP (Single Source Shortest Path) as an example to help you quickly grasp how to run Graph job.

1.Enter the console and run "odpscmd".

2.Create Input Table and Output Table.

```
create table sssp_in (v bigint, es string);
create table sssp_out (v bigint, l bigint);
```

Note:

- For the statement to create table, refer to SQL Create.

3.Upload Data

The contents of local data:

```
1	2:2,3:1,4:4
2 1:2,3:2,4:1
3 1:1,2:2,5:1
4 1:4,2:1,5:1
5 3:1,4:1
```

"tab" button is taken as the separator of two columns.

Upload the data:

```
tunnel u -fd " " sssp.txt sssp_in;
```

4.Write SSSP Example

According to the introduction in Graph Eclipse Plug-in, compile and debug SSSP Example on local.

Note:Please note that you just need to package SSSP code, not to package SDK in "odps-graph-example-sssp.jar".

5.Add Jar

```
add jar $LOCAL_JAR_PATH/odps-graph-example-sssp.jar odps-graph-example-sssp.jar
```

Note:

- For resource creation, refer to Resource Operation.

6.Run SSSP

```
jar -libjars odps-graph-example-sssp.jar -classpath $LOCAL_JAR_PATH/odps-graph-example-sssp.jar
com.aliyun.odps.graph.examples.SSSP 1 sssp_in sssp_out;
```

Jar command is used to run ODPS GRAPH. Its use method is consistent with MapReduce.When GRAPH job is running, corresponding instance ID, execution schedule and result summary will be printed on command line, as follows:

```
ID = 20130730160742915gl205u3
2013-07-31 00:18:36 SUCCESS
Summary:
Graph Input/Output
Total input bytes=211
Total input records=5
Total output bytes=161
Total output records=5
graph_input_[bsp.sssp_in]_bytes=211
graph_input_[bsp.sssp_in]_records=5
graph_output_[bsp.sssp_out]_bytes=161
graph_output_[bsp.sssp_out]_records=5
Graph Statistics
Total edges=14
Total halted vertices=5
Total sent messages=28
Total supersteps=4
Total vertices=5
Total workers=1
Graph Timers
Average superstep time (milliseconds)=7
Load time (milliseconds)=8
Max superstep time (milliseconds) =14
Max time superstep=0
Min superstep time (milliseconds)=5
Min time superstep=2
Setup time (milliseconds)=277
Shutdown time (milliseconds)=20
Total superstep time (milliseconds)=30
Total time (milliseconds)=344

OK
```

Note:If a new user needs to use graph, he must submit the application on workorder system, provide the name of his ODPS project and describe the use scenario simply. Only after your application is passed and the corresponding privileges are opened, you can use odps graph function.