Alibaba Cloud MaxCompute

Quick Start

Issue: 20180806

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.
5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade

secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).

6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 **Danger:** Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 **Warning:** Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 **Note:** Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 **Note:** You can use **Ctrl** + **A** to select all files.
>	Multi-level menu cascade.	**Settings** > **Network** > **Set network type**
Bold	It is used for buttons, menus, page names, and other UI elements.	Click **OK**.
`Courier font`	It is used for commands.	Run the `cd /d C:/windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich` *{stand | slave}*

Contents

1 Create/View/Delete a table

You can use MaxCompute services once they are added to a project, and granted the correspond ing privileges. Because the operation objects of MaxCompute (input and output) are performed on tables, you must create tables and partitions before processing data.

You can create or delete tables using the following methods:

- MaxCompute Studio. For more information, see *Visualization of operating the tables*.
- DataWorks. For more information, see and .
- Common client commands.

The following section explains how to create, view, and delete tables using commands through the DTplus console. For more information about console installation, see *Console*.

Create a table

The command format is shown as follows.

```
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
```

Command descriptions:

- The table name and column name are both case insensitive.
- If you do not specify IF NOT EXISTS when creating a table and a table with the same name exists, an error is returned. If the option is specified then all returns are successful, regardless of whether there are tables with the same name, and regardless of whether the source table structure and the target table structure are inconsistent. The Meta information of the existing table does not change.
- Only the data types BIGINT, DOUBLE, BOOLEAN, DATETIME, and STRING are supported.
- A table name and column name obey the same naming conventions as follows: The name can be up to 128 bytes in length and can contain letters, numbers, and underscores '_'.
- Partitioned by: Use PARTITIONED BY to specify the partition. Only String is supported. The value can be up to 128 bytes in length and can contain letters, numbers, and the special characters space ' ', colon ':', underscore '_', dollar sign '$', hash sign '#', dot '.', exclamation point '!' and at symbol '@'. Other characters are considered as undefined characters, such

as '\t', '\n', and '/' . If you are using partition fields in the partition table, a full table scan is not needed when adding partitions, or when updating data in the partition and then reading the partition.

- The comment content is the effective string, and it can be up to 1,024 bytes in length.
- Lifecycle indicates the lifecycle of the table. The unit is days. The statement`CREATE TABLE LIKE` does not copy the lifecycle attribute from source table.
- Currently, the partition hierarchy cannot exceed 6 levels. In a project, the maximum partition number of a table can be configured. The maximum number of tables is 60,000.

> **Note:**
>
> - For more information about creating a table, see *Create table*.
> - For more information about the partition operation, see *Add/Remove Partition*.
> - For more information about the lifecycle operation, see *Modify Lifecycle for a Table*.

The following example shows how to create a table:

```
create table test1 (key string); -- create a no-partition table. table
 name is test 1,  field name is key, data type is string.
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 for the lifecycle property,
other properties of test3 (field type, partition type) are 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.
```

In the preceding example, an instance is used to create a table.

Create a table named user that includes the following information:

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

It must be partitioned by region and dt and the lifecycle is 365 days.

An example of table creation is 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;
```

Add a partition

After creating a partition table, in order to import data into different partitions, a partition must be created. The statement format is as follows:

```
alter table table_name add [if not exists] partition partition_spec
partition_spec:
: (partition_col1 = partition_col_value1, partition_col2 = partiton_c
ol_value2, ...)
```

In the preceding example, partitions must be added for the table user (region is ‘hangzhou’ and dt is ‘20150923’). The statement is as follows:

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

View a table

View table information by using the following command: `desc <table_name>;`

For example, get information from test3: `desc test3;`

The results are as follows:

```
odps@ $odps_project>desc test3;
+------------------------------------------------------------------------------------
+
| Owner: ALIYUN$maojing.mj@alibaba-inc.com | Project: $odps_project
| TableComment: |
+------------------------------------------------------------------------------------
+
| CreateTime: 2015-09-18 12:26:57 |
| LastDDLTime: 2015-09-18 12:26:57 |
| LastModifiedTime: 2015-09-18 12:26:57 |
| Lifecycle: 100 |
+------------------------------------------------------------------------------------
+
| InternalTable: YES | Size: 0 |
+------------------------------------------------------------------------------------
+
| Native Columns: |
+------------------------------------------------------------------------------------
+
| Field | Type | Label | Comment |
+------------------------------------------------------------------------------------
+
| key | boolean | | |
+------------------------------------------------------------------------------------
+
| Partition Columns: |
+------------------------------------------------------------------------------------
+
| pt | string | |
| ds | string | |
```

```
+------------------------------------------------------------------------------------
+
```

Get information from test4: `desc  test4;`

```
odps@ $odps_project>desc test4;
+------------------------------------------------------------------------------------
+
| Owner: ALIYUN$maojing.mj@alibaba-inc.com | Project: $odps_project
| TableComment: |
+------------------------------------------------------------------------------------
+
| CreateTime: 2015-09-18 12:27:09 |
| LastDDLTime: 2015-09-18 12:27:09 |
| LastModifiedTime: 2015-09-18 12:27:09 |
+------------------------------------------------------------------------------------
+
| InternalTable: YES | Size: 0 |
+------------------------------------------------------------------------------------
+
| Native Columns: |
+------------------------------------------------------------------------------------
+
| Field | Type | Label | Comment |
+------------------------------------------------------------------------------------
+
| key | boolean | | |
+------------------------------------------------------------------------------------
+
| Partition Columns: |
+------------------------------------------------------------------------------------
+
| pt | string | |
| ds | string | |
+------------------------------------------------------------------------------------
+
```

Except for the lifecycle property, other properties of test3 (field type, partition type) are completely consistent with test4. For more information about describing a table, see *Describe Table*.

When you view the information of test5, the ‘pt’ and ‘ds’ fields only exist as two common columns, rather than as the table partitions.

Drop a partition

An example of how to drop a partition is as follows:

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

```
: (partition_col1 = partition_col_value1, partition_col2 = partiton_c
ol_value2, ...)
```

For example, to delete the partitions of region hangzhou and dt 20150923, the statement is as follows:

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

Drop a table

An example of how to drop a table is as follows:

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

For example, to delete the table test2:

```
drop table test2;
```

For more information, see *Drop Table*.

2 Data import

Data import and export through MaxCompute can be achieved by:

- Using *Tunnel Operation* on the console directly.
- Using MaxCompute Studio in the visualization method. For more information, see *Import and Export Data*.
- Writing Java tools with the SDK provided by *Tunnel*.
- Using Flume and Fluentd plug-ins.
- Using DataWorks. For more information, see *Data Sync Overview*.

For data export, see commands about downloading in *Tunnel Commands*.

Tunnel commands

1. Data preparation

   In the following example, the local file wc_example.txt is saved into the directory D:\odps\odps\bin，the contents as follows:

   ```
   I LOVE CHINA! My name is Maggie.
   I LIVE IN HANGZHOU! I LIKE PLAYING BASKETBALL!
   ```

2. Create a MaxCompute table

   To import the data created in the preceding step, a MaxCompute table must be created.

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

3. Run Tunnel command

   To import the data on MaxCompute console, run the tunnel command as follows:

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

4. After the running is successful, check the records in the table wc_in.

```
odps@ [redacted]>select * from wc_in;

ID = 20170725030626541gmbdz6jc2
Log view:
[redacted]
Job Queueing...
+-------------+
| word        |
+-------------+
| I LOVE CHINA! |
| MY NAME IS MAGGIE.I LIVE IN HANGZHOU!I LIKE PLAYING BASKETBALL! |
| NULL        |
| NULL        |
+-------------+
4 records (at most 10000 supported) fetched by instance tunnel.
```

Note:

- For more information of Tunnel commands, for example, how to import data into a partitioned table, see *Tunnel Operation*.
- If multiple columns are in the table, you can specify column separators by **-fd** parameter.

MaxCompute Studio

Before using MaxCompute Studie, make sure that you have *installed MaxCompute Studio* and *configured Project Space Connection*.

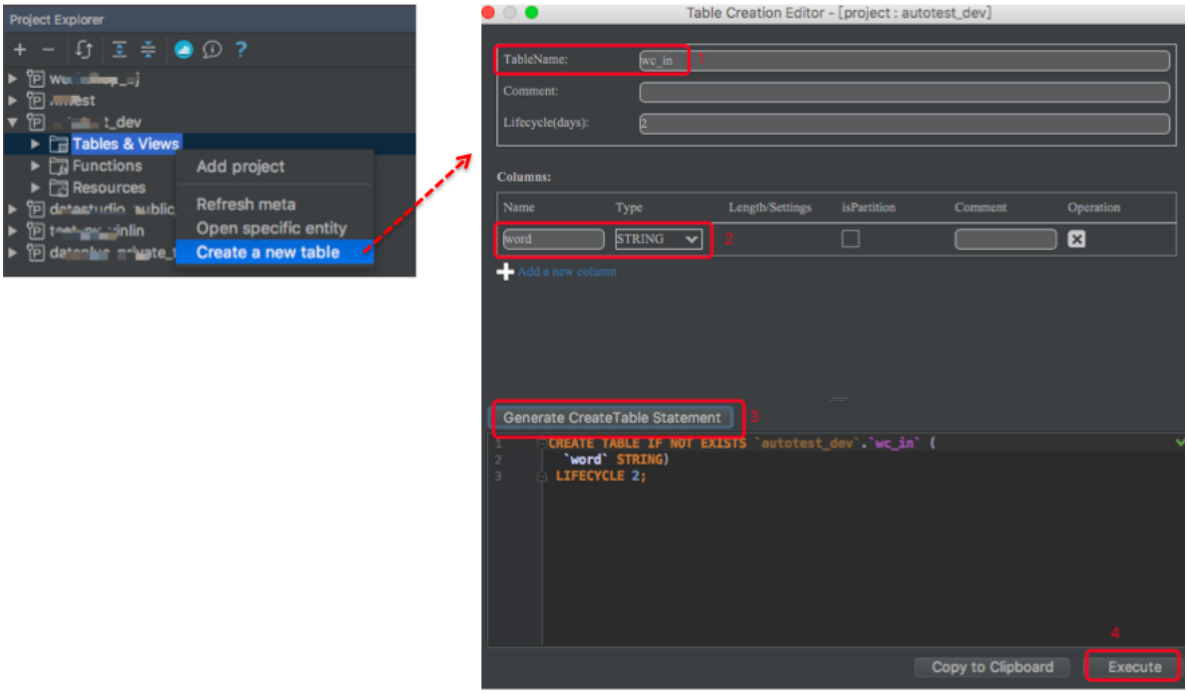
1. Data Preparation

   In the following example, the local file wc_example.txt is saved into the directory D:\odps\odps \bin, the contents as follows:

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

2. Create a MaxCompute table

   To import the data created in the preceding step, a MaxCompute table must be created first. Right-click **tables&views** in the project and operate as follows:

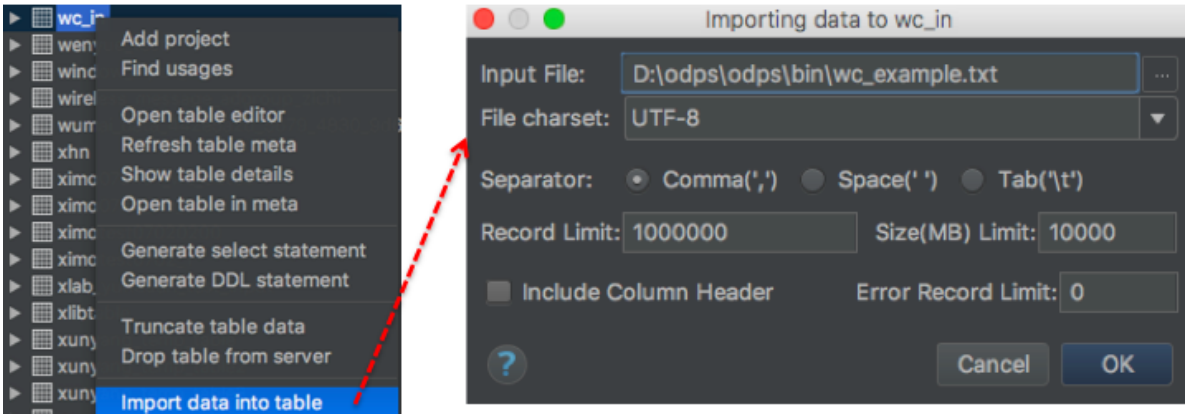


If the statement is executed successfully, then the table has been created.

3. Upload data files

   Right-click the table wc_in created in the preceding tables&view list in the project.

> **Note:**
> If the table name not appears in the list, click the **refresh** button.



Tunnel SDK

The following is a scenario example to show you how to upload data using the Tunnel SDK.

Scenario

Upload data into MaxCompute, where the project is odps_public_dev, the table name is tunnel_sample_test and the partitions are pt=20150801,dt=”hangzhou.

Procedure

1. 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.
```

2. Create the program directory structure of UploadSample as follows:

```
|---pom.xml
|---src
    |---main
        |---java
            |---com
                |---aliyun
                    |---odps
                        |---tunnel
                            |---example
                                |---UploadSample.java
```

Directory description:

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

3. 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.odps.aliyun.com";


private static String odpsUrl = "http://service.odps.aliyun.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();
United States of America, United States of America, United States of
 America (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();
UPR session. Commit (New long [] {0l });
System.out.println("upload success!") ;


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

```
}
}
}
```

Note:

The configuration of AccessKeyId and AccessKeySecret is ignored in the preceding example.

In actual operation, apply the required AccessKeyId and AccessKeySecret.

4. The configuration of pom.xml is 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</artifactId>
    <version>0.20.7-public</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>
```

5. Compile and run

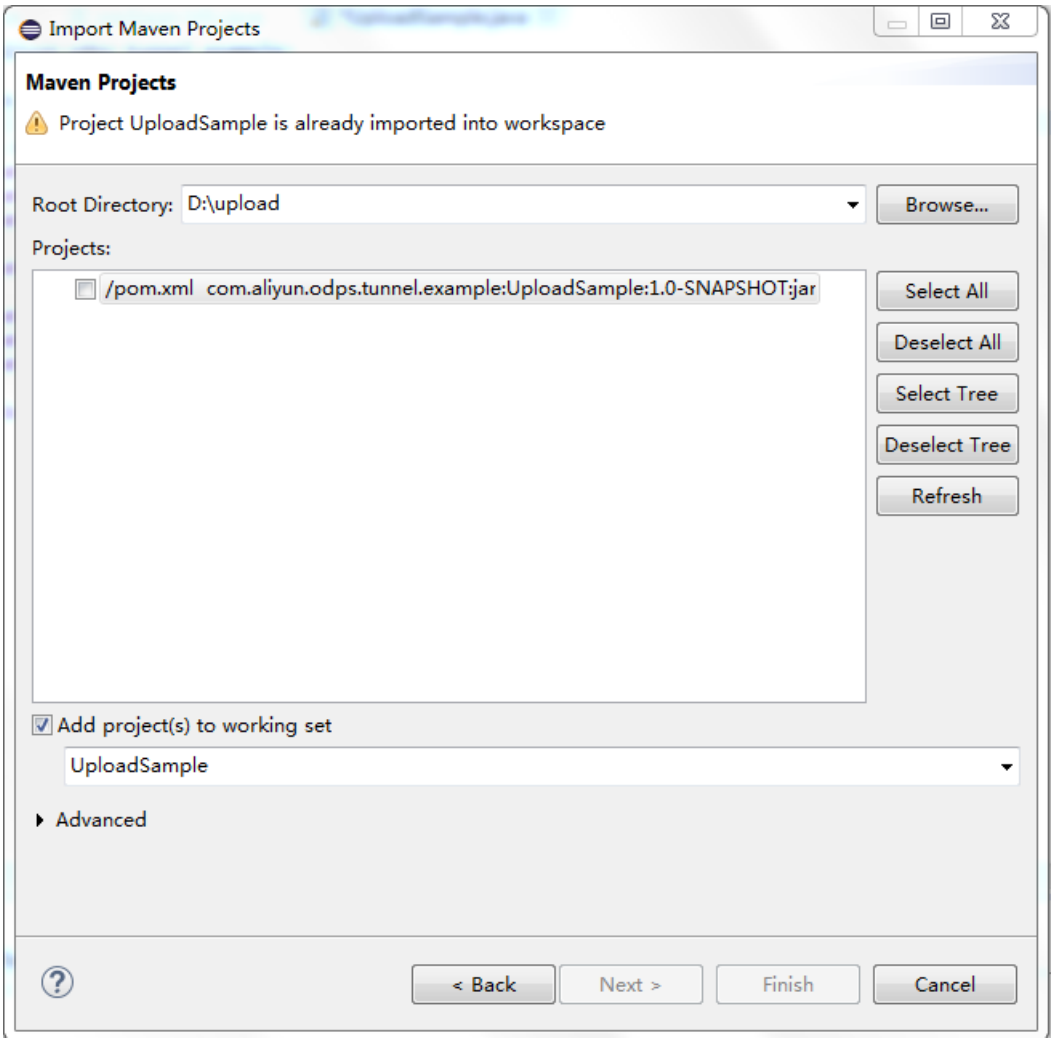
   Compile the program UploadSample:

```
mvn package
```

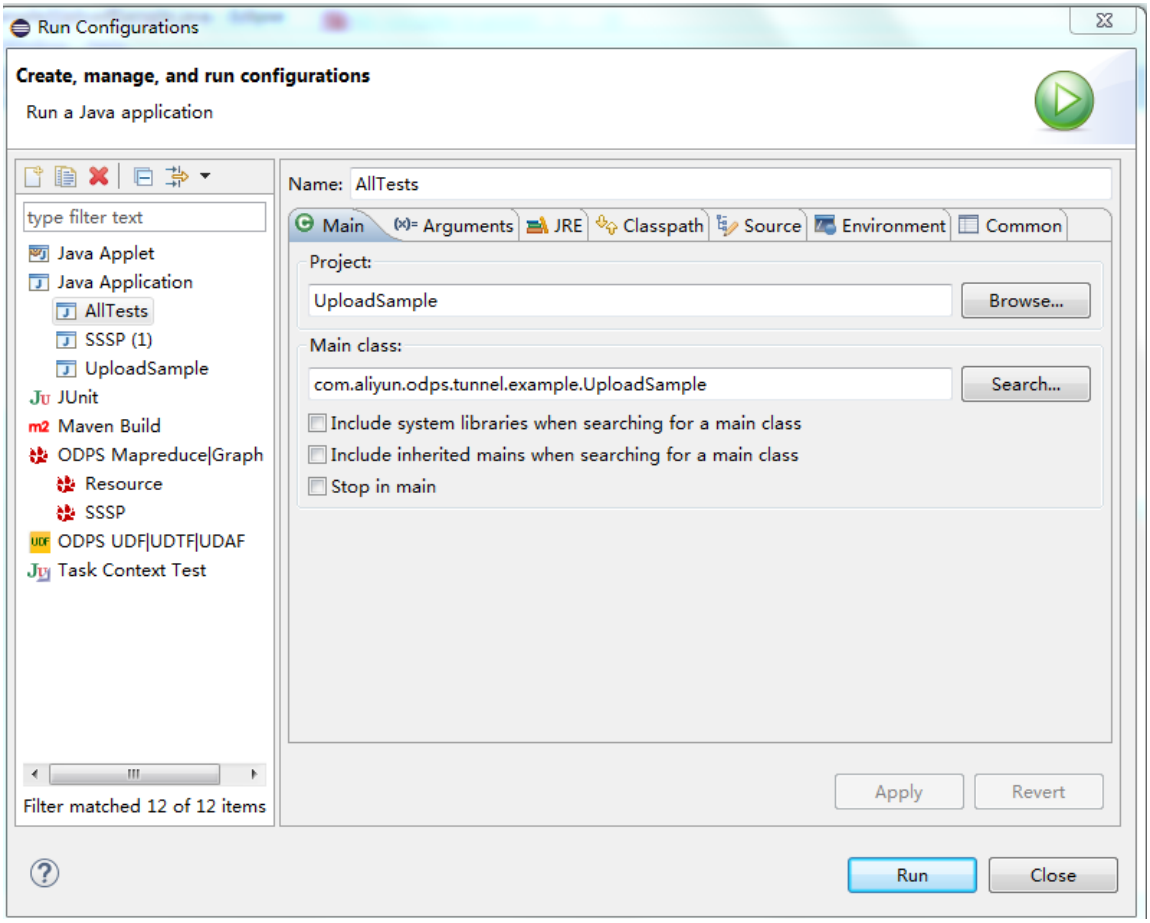
   Run the program UploadSample. Here, Eclipse is used to import the Maven project.

   1. Right-click on the **Java program** and click**Import** > **Maven** > **Existing Maven Projects**,

      The settings are as follows:



2. Right-click on **UploadSample.java** and click **Run As** > **Run Configurations**, as follows:



3. Click **Run** After running successfully, the console shows as follows:

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

6. Check running result.

   Input the following statement 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 |
```

```
+----+------+----+----+
```

Note:

- As an independent service in MaxCompute, Tunnel has exclusive access port provided for users. When you download data with MaxCompute Tunnel over the Alibaba Cloud intranet , MaxCompute does not bill you for the traffic produced by this operation. The Intranet address is only valid for cloud products in the Shanghai region.
- MaxCompute Alibaba Intranet and public network access addresses, see *Access domains and data centers*.

Other import methods

In addition to MaxCompute Console and Tunnel Java SDK, data can also be imported through Alibaba Cloud DTplus products, Sqoop, Fluentd, Flume, LogStash , and more *Tools*.

3 Run SQL

MaxCompute SQL is used to query and analyze massive data in MaxCompute. The main functions of SQL are as follows:

- Supports a variety of operators.
- Uses DDL statements to manage tables, partitions, and views.
- Uses Select statements to query records in tables and Where statements to filter records in tables.
- Uses Insert statements to insert and update data.
- Uses Join operations to join two tables. Supports Mapjoin operations for multiple small tables.
- Supports to use the built-in and user-defined functions for computing.
- Supports regular expressions.

This article gives you a brief introduction of the issues that need to be noticed using MaxCompute SQL.

Note:

- MaxCompute SQL does not support transactions, indexes, update/delete operations, and so on. At the same time, the SQL Syntax of MaxCompute is different from that of Oracle and MySQL, so that you cannot seamlessly migrate SQL statements from other databases to MaxCompute. For more differences, see *differences from other SQL syntax*.
- After you submit a MaxCompute job, it will take several dozen seconds to several minutes to schedule the job. Therefore, MaxCompute is suitable for batch jobs, which processes a massive volume of data. It is not suitable for frontend business systems that must process several thousand or tens of thousands of transactions per second.
- For a detailed example of SQL operations, see *SQL*.

DDL Statement

Simple DDL operations include creating tables, adding partitions, viewing tables and partition information, modifying tables, delete tables and partitions. For more information, see *create/View/Delete tables*.

Select Statements

- The key of GROUP BY statement can be the column name of input table, and the expression consisted of input table columns, but it cannot be the 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’ cannot be the column alias, i.e., the output column of
Select statement;
```

  For SQL parsing, GROUP BY operations are conducted before SELECT operations, which means GROUP BY can only use the column or expression of the input table as the key.

- ORDER BY must be used in combination with LIMIT.
- DISTRIBUTE BY 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, that is, the column alias. An example is shown as follows:

```
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.
```

  For SQL parsing, ORDER BY/SORT BY/DISTRIBUTE BY 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 the 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 operation types: {LEFT OUTER|RIGHT OUTER |FULL OUTER|INNER} JOIN.
- MaxCompute SQL supports up to 16 concurrent JOIN operations.
- MaxCompute supports the map JOIN up to 8 small tables.

Union all

Union All can combine the results returned from multiple Select operations into a data set. It returns all the results without deduplication. MaxCompute does not support union two main query results, but you can do it on two subquery results.

Note:

- The two Select queries connected by Union All, must have the same number of columns, column names, and column types.
- If the original names are inconsistent, you can set the same name by the alias.

Additional information

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

For more restrictions on MaxCompute SQL, see *SQL Restrictions Summary*.

SQL optimization example

- **Where condition in Join statement**

  When you join two tables, the Where condition of the master table can be written at the end of the statement, but the restriction condition of the partition in the slave table cannot be written in the Where condition. We recommend to write it in the ON condition or subquery. The partition restrictions of the master table can be written in Where condition (it is better to filter by subquery first). Several SQL examples are as follows:

```
select * from A join (select * from B where dt=20150301)B on B.id=A.
id where A.dt=20150301;
select * from A join B on B.id=A.id where B.dt=20150301; --Not
allowed.
```

```
select * from (select * from A where dt=20150301)A join (select *
from B where dt=20150301)B on B.id=A.id ;
```

The Join operation in the second statement runs first, data volume becomes larger and the performance can be decreased. Therefore, the second statement must be avoided.

- **Data skew**

The root cause of data skew is that the amount of data processed by some Workers is much larger than that of other Workers, resulting in the running hours of some Workers are more than the average, which leads to the job delay.

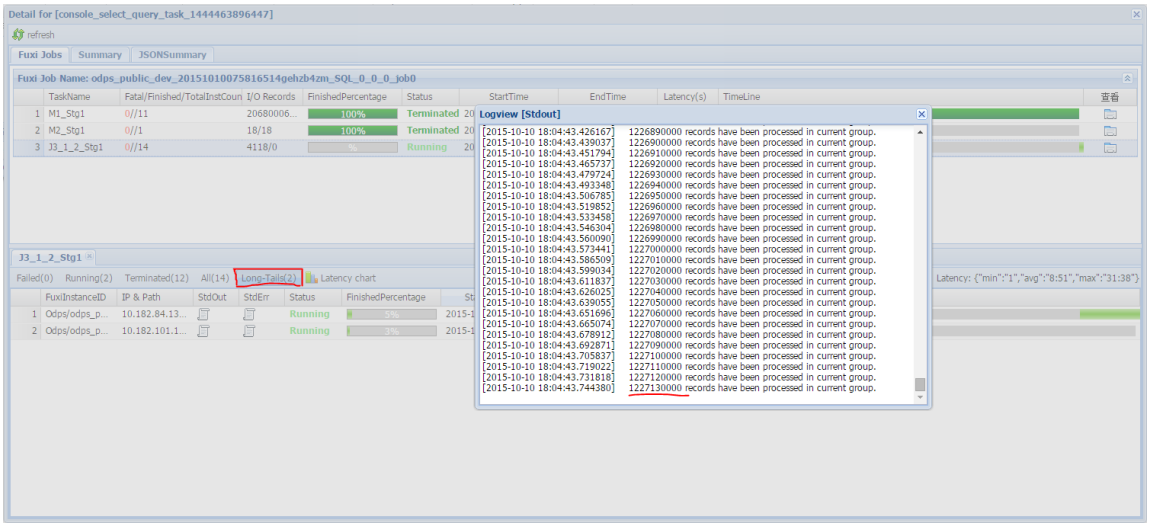
For more information about data skew optimization, see *Optimize long tail computing*.

— Data skew caused by Join

The reason of the data skew caused by Join operation is that keys distribution of Join on is uneven. For the preceding example, to join a large table A and a small table B, run the following statement:

```
select * from A join B on A.value= B.value;
```

Copy the logview link to enter the web console page, and double click the Fuxi job that runs the Join operation. You can see a long tail in the Long-Tails tab, which indicates that the data has skewed, as shown in the following figure:



You can optimize the statement by the following methods:

- Since table B is a small table and does not exceed 512 MB, you can optimize the preceding statement into mapjoin statement.

```
select /*+ MAPJOIN(B) */ * from A join B on A.value= B.value;
```

- Handle the skewed key with a separate logic. For example, a large number of null value of the key in both tables may usually cause data skew. It is necessary to filter out the null data or add a random number before Join operation, for example:

```
select * from A join B
on case when A.value is null then concat('value',rand() ) else
A.value end = B.value;
```

If you have realized that the data is skewed, but you cannot get the key information that causes the data skew, a general solution can be used to view the data skew. See the following example:

```
select * from a join b on a.key=b.key; --This Leads to data skew.
Now you can run the following statements:
```sql
select left.key, left.cnt * right.cnt from
(select key, count(*) as cnt from a group by key) left
join
(select key, count(*) as cnt from b group by key) right
on left.key=right.key;
```

Check the distribution of keys to view whether data skew happens when A joins B.

- **Group by skew**

  The reason of group by skew is that the key distribution of group by is uneven.

  Suppose the table A has two fields: key and value. The data volume in the table is large enough, and the value distribution of key is uneven. Run the following statement:

```
select key,count(value) from A group by key;
```

  You can see the long tail on the web console page. To solve this problem, you must set the anti-skew parameters before running SQL statement. `set odps.sql.groupby.skewindata=true` must be added into the SQL statement.

- **Data skew caused by incorrect use of dynamic partitions**

  Dynamic partitions of SQL in MaxCompute by default add a Reduce function, which is used to merge the same partition data. The benefits are as following:

  - Reduce small files generated by the MaxCompute and improve the efficiency of processing.
```

- Avoid occupying a large amount of memory when a Worker output many files.

When partition data is skewed, using the Reduce function lead to the appearance of long tails. The same data can only be processed by a maximum of 10 Workers, so large volume of data results in a long tails, for example:

```
insert overwrite table A2 partition(dt)
select
split_part(value,'\t',1) as field1,
split_part(value,'\t',2) as field2,
dt
from A
where dt='20151010';
```

In this case, we recommend that you do not use dynamic partition, and modify the statement in the following way:

```
insert overwrite table A2 partition(dt='20151010')
select
split_part(value,'\t',1) as field1,
split_part(value,'\t',2) as field2
from A
where dt='20151010';
```

- **Window function optimization**

If you use window functions in your SQL statement, each window function typically forms a Reduce job. If window functions are too many, they consume resources. In some specific scenarios, you can optimize window functions.

- The content after the over keyword must be the same, with the similar grouping and sorting conditions.
- Multiple window functions must run on the same SQL layer.

Window functions that meet these two conditions merge into Reduce implementation. An SQL example is as follows:

```
select
rank()over(partition by A order by B desc) as rank,
row_number()over(partition by A order by B desc) as row_num
from MyTable;
```

- **Convert the subquery to Join**

A subquery is shown as follows:

```
SELECT * FROM table_a a WHERE a.col1 IN (SELECT col1 FROM table_b b
WHERE xxx);
```

If the number of col1 returned by the table_b subquery in this statement exceeds 1,000, the system reports an error: `records returned from subquery exceeded limit of 1,000`. In this case, you can use the Join statement instead:

```
SELECT a. * FROM table_a a JOIN (SELECT DISTINCT col1 FROM table_b b
 WHERE xxx) c ON (a.col1 = c.col1)
```

Note:

- If no Distinct is keyword in the statement, and the result of the subquery c returns the same col1 value, it may cause the larger number of results of table_a.
- The Distinct subquery lead the whole query to fall into one Worker. If the subquery data is large, it may cause the whole query to be slower.
- If you have already made sure the col1 values are distinct in the subquery from the business, for example, querying by the primary key field, to improve performance the Distinct keyword can only be removed.

4 MapReduce

The following section explains how to use the MapReduce coding model with MaxCompute. For the example code in the procedure, it is assumed that the MaxCompute console is installed.

Note:

Maven users can search odps-sdk-mapred from the *Maven Library* to get the required SDK (available in different versions). The configuration is as follows:

```
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-sdk-mapred</artifactId>
    <version>0.26.2-public</version>

</dependency>
```

Preparation

- JDK1.7 is required to compile and run MapReduce.
- For inforation about *installing the MaxCompute console*, see Quick Start. For infromation about using the MaxCompute client, see*Console*.

Procedure

1. After installing and configuring the client, open odpscmd.bat and enter the appropriate project space.
2. Input build table statements to create Input and Output tables. An example is shown as follows:

   ```
   CREATE TABLE wc_in (key STRING, value STRING);
   CREATE TABLE wc_out (key STRING, cnt BIGINT);
   -- Create input table and output table
   ```

   For information about using SQL statements to create tables, see *Table Operations*.
3. Upload data.

   You can upload data in two ways.

   - Use Tunnel Commands to upload data:

     ```
     tunnel upload kv.txt wc_in
     --  Upload example data
     ```

     The data is shown in kv.txt as follows:

     ```
     238,val_238
     186,val_86
     ```

```
186,val_86
```

- You can also insert data directly using the INSERT statement as follows:

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

4. Write MapReduce program and compile it.

   MaxCompute supports an Eclipse development plug-in to help quickly develop MapReduce programs and provide a local debugging MapReduce function.

   Users must create a MaxCompute project in Eclipse first, and then write the MapReduce program. After the local debugging is run successfully, users can upload the compiled program to MaxCompute. For more information, see *MapReduce Eclipse Plug-in*.

5. Add .jar package into the project. (in this example, the name of the JAR package is “word-count-1.0.jar”):

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

6. Run the “jar” command on the 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;
```

7. Check the running result on the MaxCompute console:

```
select * from wc_out;
```

Note:

If other resources are used in a Java program, you must add **`-resources`** parameters. For more information about JAR commands, see *Jar Commands*.

5 Java UDF Development

MaxCompute user-defined functions (UDFs) include User Defined Scalar Function (UDF), User Defined Aggregation Function (UDAF), and User Defined Table Valued Function (UDTF). In general, all these functions are called UDF.

Users who use Maven can search **odps-sdk-udf** in the *Maven Library* to get the required Java SDK (available in different versions). The related configuration is shown as follows:

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

Currently, JAVA UDF can be developed in the following ways:

- Using *MaxCompute Studio completes Java UDF development throughout the process*.
- Use the *develop and debug JAVA UDF with Eclipse plug-ins*, export jar packages, then use the command or DataWorks to *add the resource* and then *register the function*.

The code examples for UDF, UDAF, udtf are given separately in this article, and the complete process example for developing UDF in two methods will be displayed(The steps of UDAF, UDTF are the same as that of UDF).

Note:

- For related statements about custom function registration and logout, and viewing function list, see *function*.
- For the data types mapping for Java and MaxCompute, see *parameters and return types*.

UDF Example

The following example shows how to develop a UDF to realize character lowercase conversion.

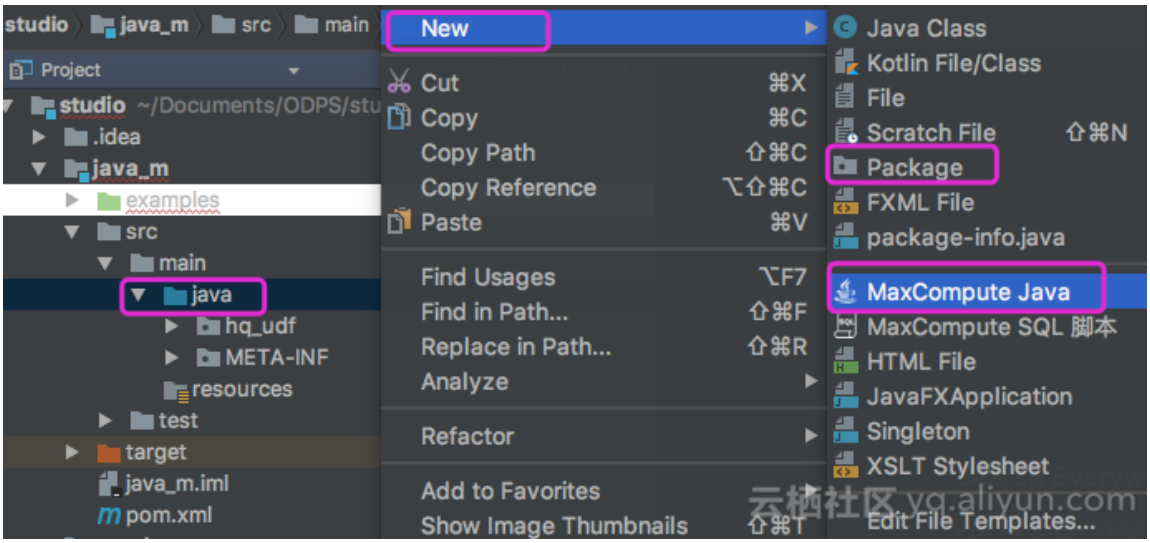
- **Developing using MaxCompute Studio**

  1. **Prepare tools and environment**.

     Suppose that the environment has been prepared, which includes *installing Studio*, creating a *MaxCompute project link* on Studio, and creating a *MaxCompute Java module*.

  2. **Write program**.

     Create a Java file under the configured Java Module.



Select MaxCompute Java directly and enter the `package name in name text box, file name`. Select UDF for Kind. Edit the code as following:

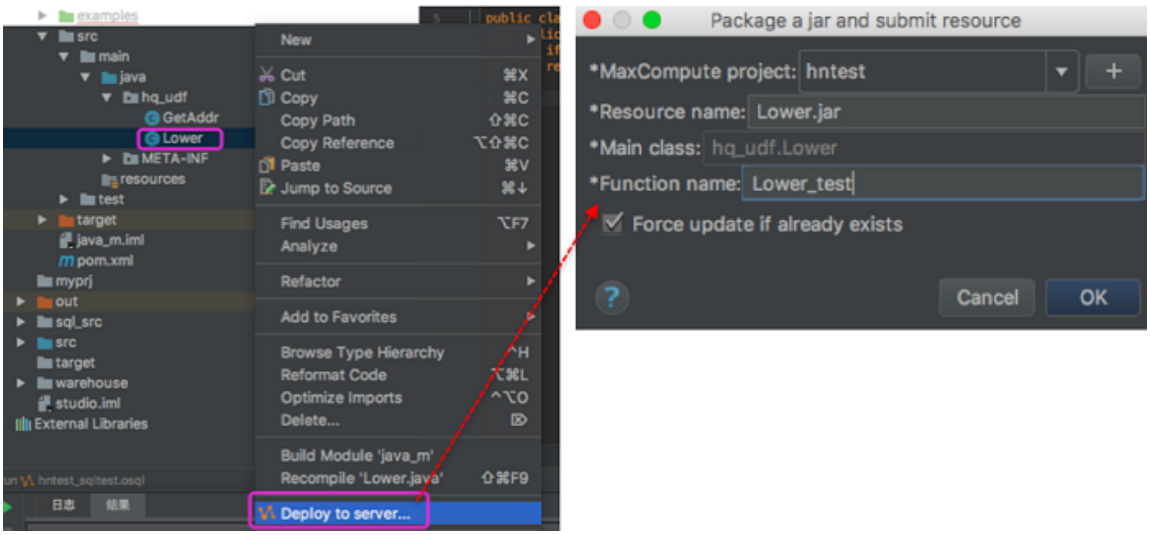
```
package <package name>;
import com.aliyun.odps.udf.UDF;
public final class Lower extends UDF {
Public String evaluate (string s ){
 if (s == null) { return null; }
 return s.toLowerCase();
}
}
```

Note:

If you want to debug Java locally UDF, see *develop and debug UDF*

3. **Register MaxCompute UDF**.

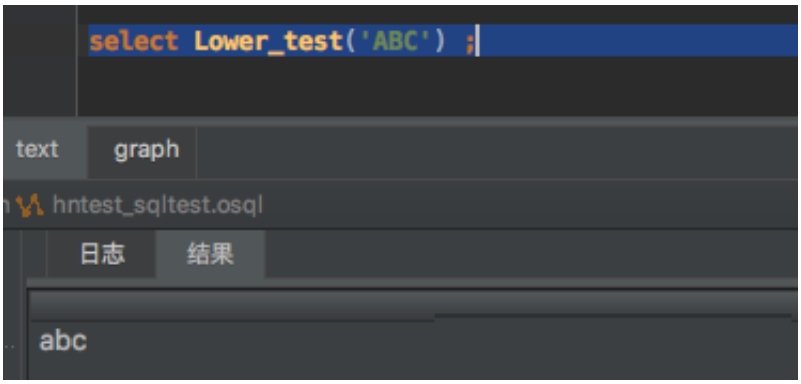
   As shown in the following figure, right-click the UDF's Java file and select **Deploy to server** , select the MaxCompute project to be registered in the dialog box. enter a `function name`. The resource name also can be modified.



When all configurations are finished, click **OK**. There are prompts after the registration is successful.

4. **Try the UDF**.

   Open the SQL script and execute the code such as `select Lower_test('ABC');`. The result is shown in the following figure:



> **Note:**
> To write SQL scripts in Studio, see *writing SQL scripts*.

- **Developing using the Eclipse plug-in**

  1. **Creating a Project**

     Suppose that a MaxCompute (formerly ODPS) project has been created in the Eclipse plug-in, for more information, see *creating a MaxCompute project*.

  2. **Writing program**

To archive function, write a program and compile in terms of MaxCompute UDF frame. For examples:

```
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();
}
}
```

Name the JAR package **my_lower.jar**.

Note:

- For more detailed introduction of developing debugging code, see *UDF*.
- For more information about the SDK, see *UDF SDK* .

3. **Add resource**

Before running the UDF, use the cited UDF code. The user's code is added to MaxCompute in the form of resource. Java UDF must be compiled into the Jar package and added in MaxCompute as a Jar resource. The UDF framework loads the JAR package automatically and runs UDF.

Note:

MaxCompute *MapReduce* also describes the use of resource.

Run the command:

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

4. **Register UDF**

After the JAR package has been uploaded, MaxCompute can obtain a user's code and run it. Note that, for the UDF to be usable, MaxCompute requires the user to register a unique function name in MaxCompute and specify which function is corresponding to this function name in the Jar resource.

run the command:

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

5. Use this function in SQL:

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

UDAF Example

The registration method of a UDAF is similar to a UDF. Its usage is also the same as *Aggregation Function* in built-in function. The following example shows a UDAF code 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 UDFExcepti
on {
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 registration method and usage of a UDTF is similar to a UDF. The code example is 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 );
}
}
}
```

MaxCompute provides many built-in functions to meet your computing needs, while you can also create custom functions. For more information, see *creating an UDF*.

6 Graph

This article uses the *SSSP Algorithm* in an example to show how to submit Graph jobs.

Submitting a *Graph* job is similar to submitting a job using *MapReduce*. Maven users can search `odps-sdk-graph` from *Maven Library* to get the Java SDK (available in different versions). The related configuration information is as follows:

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

Procedure

1. Enter the console and run odpscmd.
2. Create Input Table and Output Table and select the target cluster.

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

   For the statement used to create a table, see *SQL Create*.
3. Upload Data.

   The content of local data is as follows:

   ```
   2, 2, 3, 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
   ```

   The tab button is used to separate two columns.

   ```
   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. In this example, we assume the code is packaged as *`odps-graph-example-sssp.jar`*.

   Note:

   You only need to package the SSSP code. You do not need to package the SDK in *`odps-graph-example-sssp.jar`*.

5. Add JAR package.

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

Note:

For resource creation, see *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.example.SSSP 1
sssp_in sssp_out;
```

Jar command is used to run MaxCompute Graph. Its use is consistent with *MapReduce*.

When Graph job is running, the corresponding instance ID, execution schedule and result summary are printed on the 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 superb time (milliseconds) = 30
Total time (milliseconds)=344
OK
```

Note:

If you need to use the Graph function, simply open the submit Graph calculation job.