Alibaba Cloud MaxCompute

Quick Start

Issue: 20190318

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.	 Notice: 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]*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich {stand | slave}`

Contents

1 Create and view a table

This topic describes how to use MaxCompute to analyze data concerning house buyers with bank loans. After you purchase a MaxCompute project with your Alibaba Cloud account, you are added to the project and authorized to create a table. Afterwards, you can complete client configurations on your local PC and operate MaxCompute.

> Note:
> If you are using MaxCompute for the first time, be sure to complete all ***preparations*** before you get started.
>
> The MaxCompute quick start guide describes how to use the *MaxCompute client* and *MaxCompute Studio* to create tables and how to upload, process, and export data. Alternatively, you can use DataWorks to perform these operations. For more information, see *DataWorks quick start*.

In MaxCompute, the basic operations (input and output) objects are tables, so you need to create tables and partitions before processing data. You can use any of the following methods to create, view, and delete tables:

- Run *common commands* on the MaxCompute client.
- Follow the instructions provided in *Visualization of operating the tables* to use MaxCompute Studio.
- Follow the instructions provided in *Create a table* and *Delete a table* to use DataWorks.

The following describes how to use the ***MaxCompute client*** to create and view a table.

Create a table

Log on to the MaxCompute client, and then create a table by running the following command:

```
CREATE TABLE [ IF NOT EXISTS ] table_name
                [( col_name data_type [ COMMENT col_commen t
], ...)]
                [ COMMENT table_comm ent ]
                [ PARTITIONE D BY ( col_name data_type [
COMMENT col_commen t ], ...)]
                [ LIFECYCLE days ]
                [ AS select_sta tement ]
                 CREATE TABLE [ IF NOT EXISTS ] table_name
```

```
                    LIKE   existing_t  able_name
```

Note:

For more information, see *Table operations*.

After logging on to the MaxCompute client, you need to confirm that you are in the correct project. In this example, the project name is MaxCompute_DOC, so you can run `use  MaxCompute  _DOC ;` to switch to the project.

In this example, the bank_data table is used to store service data, and the result_table table is used to store data processing results.

Create the bank_data table by running the following command:

```
CREATE   TABLE   IF   NOT   EXISTS   bank_data
               (
                age               BIGINT   COMMENT  ' age ',
                job               STRING   COMMENT  ' job   type ',
                marital           STRING   COMMENT  ' marital
status ',
                education         STRING   COMMENT  ' educationa  l
level ',
                default           STRING   COMMENT  ' credit   card
ownship ',
                housing           STRING   COMMENT  ' mortgate ',
                loan              STRING   COMMENT  ' loan ',
                contact           STRING   COMMENT  ' contact
informatio  n ',
                month             STRING   COMMENT  ' month ',
                day_of_wee  k       STRING   COMMENT  ' day   of
the   week ',
                duration          STRING   COMMENT  ' Duration ',
                campaign          BIGINT   COMMENT  ' contact   times
  during   the   campaign ',
                pdays             DOUBLE   COMMENT  ' time   interval
  from   the   last   contact ',
                previous          DOUBLE   COMMENT  ' previous
contact   times   before   this   activity ',
                poutcome          STRING   COMMENT  ' market
activity   results   before   this   activity ',
                emp_var_ra  te      DOUBLE   COMMENT  ' employment
change   rate ',
                cons_price  _idx    DOUBLE   COMMENT  ' consumer
price   index ',
                cons_conf_  idx     DOUBLE   COMMENT  ' consumer
confidence   index ',
                euribor3m         DOUBLE   COMMENT  ' euro   deposit
  rate ',
                nr_employe  d       DOUBLE   COMMENT  ' number   of
employees ',
                y                 BIGINT   COMMENT  ' whether   a
fixed   deposit   holder '
               );
```

The command output OK is displayed, as shown in the following figure.

Create the result_table table by running the following command:

```
CREATE   TABLE   IF   NOT   EXISTS   result_tab  le
                (
                 education     STRING   COMMENT  ' education   level
',
                 num           BIGINT   COMMENT  ' number   of
people '
                );
```

View a table

Run `desc < table_name >;` to view information related to a table.

For example, you can run `desc bank_data ;` to view information related to the bank_data table that was created in the previous step.

The following figure shows the command output.

```
odps@ MaxCompute_DOC>desc bank_data;

+------------------------------------------------------------------------------------
| Owner: RAM$dtplus_docs:[redacted] | Project: maxcompute_doc
| TableComment:
+------------------------------------------------------------------------------------
| CreateTime:                   2018-12-18 17:01:50
| LastDDLTime:                  2018-12-18 17:01:50
| LastModifiedTime:             2018-12-18 17:01:50
+------------------------------------------------------------------------------------
| InternalTable: YES      | Size: 0
+------------------------------------------------------------------------------------
| Native Columns:
+------------------------------------------------------------------------------------
| Field           | Type       | Label | Comment
+------------------------------------------------------------------------------------
| age             | bigint     |       | age
| job             | string     |       | job type
| marital         | string     |       | marital status
| education       | string     |       | educational level
| default         | string     |       | credit card ownership
| housing         | string     |       | mortgage
| loan            | string     |       | loan
| contact         | string     |       | contact information
```

For more information, see *Table operations*.

(Optional) Create a partition

The table used in this example is a non-partition table.

If you want to *run the Tunnel command to import data of different partitions* to a *partition table*, you must create a partition by running the following commands:

```
alter table table_name add [ if not exists ] partition
  partition ( partition_ col1 = partition_ col_value1 ,
partition_ col2 = partiton_c ol_value2 , ...);
```

You do not need to create an independent partition for different operations, such as *data integration* and insert operations.

(Optional) Delete a partition

Delete a partition by running the following command:

```
alter table table_name drop [ if exists ] partition (
partition_ col1 = partition_ col_value1 , partition_ col2 =
partiton_c ol_value2 , ...);
```

For example, if you want to delete a partition with the date of `20180923` and region of `hangzhou`, run the following command:

```
alter table user drop if exists partition ( region ='
hangzhou ', dt =' 20180923 ');
```

(Optional) Delete a table

Create a table by running the following command:

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

For more information, see *Table operations*.

What to do next

You can refer to *Import data* for data processing.

2 Import data

This topic describes how to import data to MaxCompute by running a tunnel command.

MaxCompute provides *different methods for importing and exporting data*. For more information about how to import data by running a tunnel command on the MaxCompute client, see *Tunnel commands*.

Run a tunnel command to import data

1. Prepare the data.

   Download the *bank.txt* file for this example and save it to `D :\`. This file contains the information such as the age, profession, and highest level of educational attainment of different people. The following is an example:

   ```
   44 , blue - collar , married , basic . 4y , unknown , yes , no ,
   cellular , aug , thu , 210 , 1 , 999 , 0 , nonexisten  t , 1 . 4 ,
   93 . 444 ,- 36 . 1 , 4 . 963 , 5228 . 1 , 0
   53 , technician , married , unknown , no , no , no , cellular , nov
   , fri , 138 , 1 , 999 , 0 , nonexisten  t ,- 0 . 1 , 93 . 2 ,- 42 ,
   4 . 021 , 5195 . 8 , 0
   28 , management , single , university . degree , no , yes , no ,
   cellular , jun , thu , 339 , 3 , 6 , 2 , success ,- 1 . 7 , 94 .
   055 ,- 39 . 8 , 0 . 729 , 4991 . 6 , 1
   ```

2. Create a MaxCompute table.

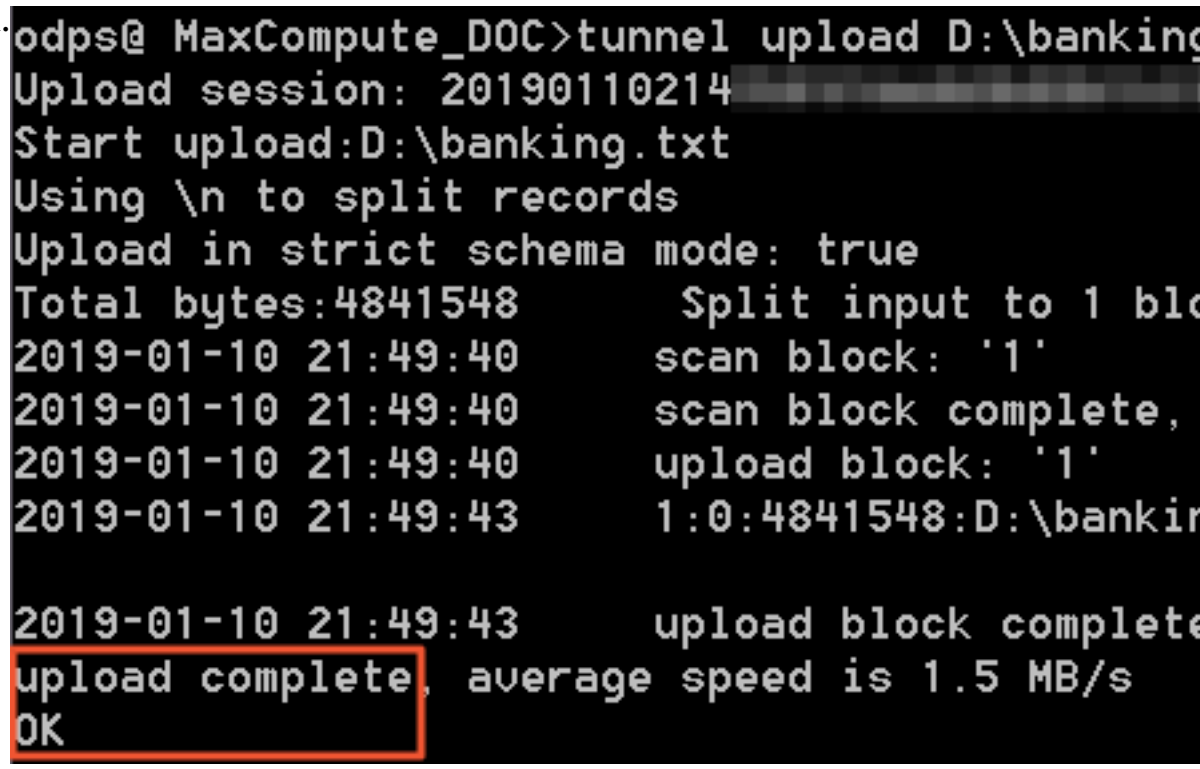
   Create a MaxCompute table to store the preceding data. If you have created a table according to *Create and view a table*, skip this step. In this example, the table is named bank_data.

3. Run a tunnel command.

   Run the following tunnel command to import data to the MaxCompute client:

   ```
   tunnel upload D :\ banking . txt bank_data ;
   ```

   If OK (outlined in red in the following figure) is displayed, the data is uploaded.



4. Verify the result.

   Use `select count (*) from bank_data ;` to check whether all the records in the bank_data table are successfully uploaded. In this example, the table has 41,188 records in total.

   Note:

- For more information about tunnel commands, for example, how to import data into a partitioned table, see *Tunnel commands*.

Other import methods

You can also import data to MaxCompute by using *MaxCompute Studio*, *Tunnel SDK*, *data integration*, or open-source applications such as Sqoop, Fluentd, Flume, and Logstash. For more information, see *Data upload and download tools*.

What to do next

After importing data to MaxCompute, you can run SQL statements to process the data in MaxCompute. For more information, see *Run SQL statements and export data*.

3 Run SQL statements and export data

This topic describes how to run SQL statements to export data.

After importing data to MaxCompute, you can process the data by running SQL statements on the *MaxCompute client* or in *DataWorks*.

Currently, MaxCompute SQL supports the use of:

- A variety of operators.
- DDL statements to manage tables, partitions, and views.
- SELECT statements to query records in tables and WHERE statements to filter records in tables.
- INSERT statements to insert and update data.
- JOIN operations to associate two tables and MAPJOIN operations for multiple small tables.
- Built-in and user-defined functions for computing.
- Regular expressions.

Note:

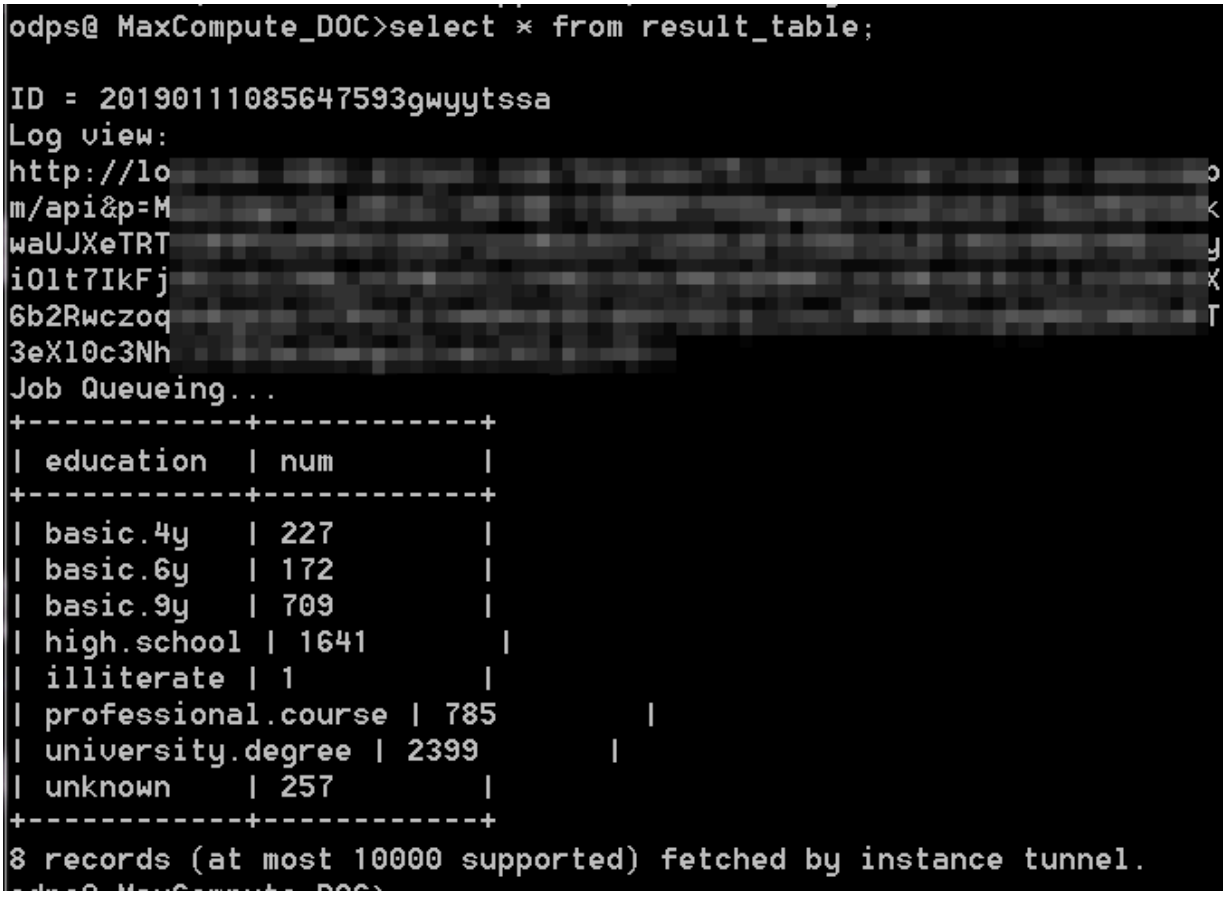
- MaxCompute SQL does not support transactions, indexes, update operations, or delete operations. The SQL syntax of MaxCompute is different from that of Oracle and MySQL. Therefore, you cannot migrate SQL statements from other databases to MaxCompute.
- For more information about SQL operations, see *SQL summary*.
- After you submit a MaxCompute job, some time is required to schedule the job. Therefore, MaxCompute is suitable for processing jobs in batches, that is, processing a massive volume of data. MaxCompute is not suitable for frontend business systems that must process several thousand or tens of thousands of transactions per second. For more information about how to optimize a job, see *SQL optimization*.
- For the limits of MaxCompute SQL, see *SQL limits*.

Obtain and analyze data

The following is an example of using SQL to query the number of single people (with different education backgrounds) that have purchased a house. The result is saved to the result_table table for analysis and display.

```
INSERT OVERWRITE TABLE result_tab le -- Insert data to
the result_tab le table .
SELECT education , COUNT ( marital ) AS num
FROM bank_data
WHERE housing = ' yes '
    AND marital = ' single '
GROUP BY education ;
```

You can use `select * from result_tab le ;` to view data in the result_table table, as shown in the following figure.



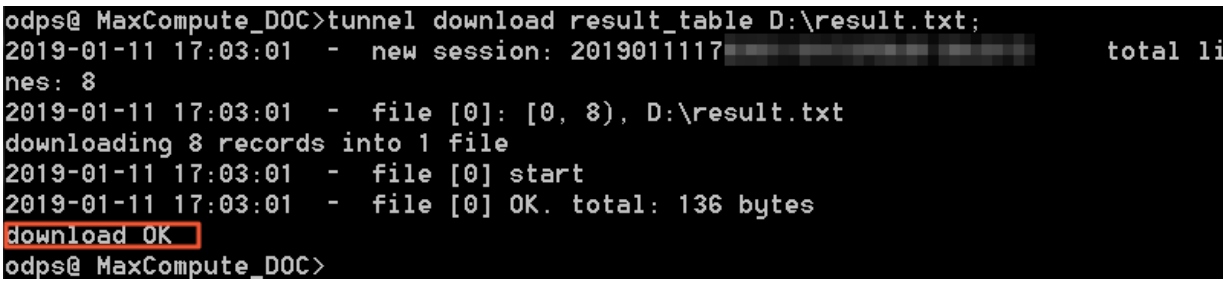
You can process multiple tables by using multiple SQL statements in DataWorks. For more information, see *Business flow*.

Export data

After processing data by using SQL statements, you can export data to your D drive by running a tunnel command. For more information, see *Tunnel commands*.

```
tunnel download result_table D:\result.txt;
```

If download OK (outlined in red in the following figure) is displayed, the data is successfully exported.



Note:

You can use the *data integration* feature to export data to other data sources (such as MySQL).

4 (Optional) Use MapReduce

This topic describes how to compile and run the MapReduce WordCount example program after you install the MaxCompute client.

Note:

Maven users can search `odps - sdk - mapred` at *Maven library* to download the preferred Java SDK (available in different versions). The configuration information is as follows:

```
< dependency >
    < groupId > com . aliyun . odps </ groupId >
    < artifactId > odps - sdk - mapred </ artifactId >
    < version > 0 . 26 . 2 - public </ version >
</ dependency >
```

Prerequisites

- JDK1.6 or a later version is installed.
- The MaxCompute client is deployed by following the instructions provided in *Install and configure a client*. For more information, see *Client*.

Procedure

1. After installing and configuring the client, open `odpscmd . bat` and enter the corresponding project space.
2. Run the following statements to create input and output tables:

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

   For more information about table creation statements, see *Table operations*.

3. Upload data by using either of the following methods:
   - Use the `Tunnel` command, for example:

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

     The data in the *kv . txt* file is as follows:

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

```
186 , val_86
```

- Insert data by running an SQL statement, for example:

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

4. Compile the MapReduce program and upload it to the MaxCompute client.

   In this topic, Eclipse-based MapReduce program development is used as an example. MaxCompute provides the *Eclipse development plug-in* and local MapReduce debugging (which you can use alongside *MaxCompute Studio*). These methods can help you to quickly develop MapReduce programs.

   You must create a MaxCompute project in Eclipse first, and then compile the MapReduce program (a .jar package, for example, *Word-count-1.0.jar*). After the local debugging is performed successfully, upload the compiled program to MaxCompute. For more information, see *Install*.

5. Add the .jar package to the project by running the following command:

```
add jar word - count - 1 . 0 . jar ;
```

6. In the MaxCompute client, run the following command:

```
jar - resources word - count - 1 . 0 . jar - classpath / home
/ resources / word - count - 1 . 0 . jar com . taobao . jingfan .
WordCount wc_in wc_out ;
```

7. In the MaxCompute client, check the results by running the following command:

```
select * from wc_out ;
```

Note:

If any resource is used in a Java program, you must add the resource to the `- resources` parameter. For more information about JAR commands, see *Commands*.

5 (Optional) Develop Java UDFs

MaxCompute user-defined functions (UDFs) include User Defined Scalar Functions (UDFs), User Defined Aggregation Functions (UDAFs), and User Defined Table Valued Functions (UDTFs).

> Note:
> MaxCompute currently supports *JAVA UDFs*, *Python UDFs*, *UDJs*, and *UDTs*. For more information, see *JAVA UDFs*.

Users who use Maven can search `odps - sdk - udf` at *Maven library* to download the preferred Java SDK (available in different versions). The configuration information is as follows:

```
< dependency >
    < groupId > com . aliyun . odps </ groupId >
    < artifactId > odps - sdk - udf </ artifactId >
    < version > 0 . 20 . 7 </ version >
</ dependency >
```

You can use the following methods to develop Java UDFs:

- Use *MaxCompute Studio*.
- Use the *Eclipse plug-in to develop and debug Java UDFs*, export .jar packages, use commands or DataWorks to *add resources*, and then *register the functions*.

This topic provide code examples for UDFs, UDAFs, and UDTFs and the whole development procedures by using the two methods.

> Note:
> - For more information about statements for custom function registration and logout and function list display, see *Functions operations*.
> - For more information about the data type mapping between Java and MaxCompute, see *Parameters and return value types*.

UDF example

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

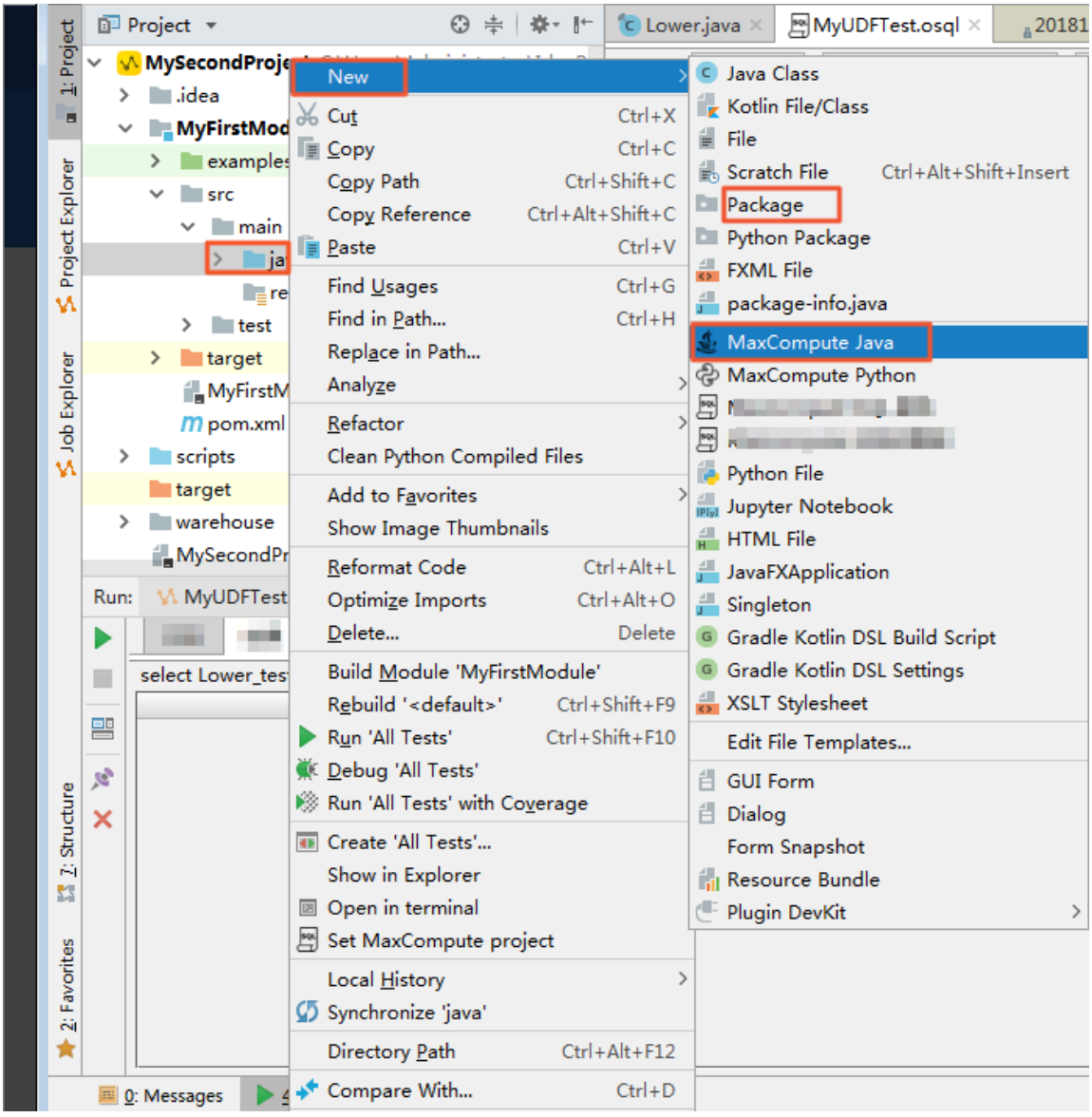
- Use MaxCompute Studio to develop a UDF.

  1. Prepare the development environment and create a Java module.

     Environment preparations include *installing MaxCompute Studio*, *creating a MaxCompute project link* on MaxCompute Studio, and *creating a MaxCompute Java module*.

  2. Compile code.

     Create a Java file under the configured Java module.



     As shown in the preceding figure, java > MaxCompute Java. In the displayed dialog box, enter a `package name . file name` and set Kind to UDF. Then, compile the following code:

```
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 . toLowerCas  e ();
}
}
```

Note:

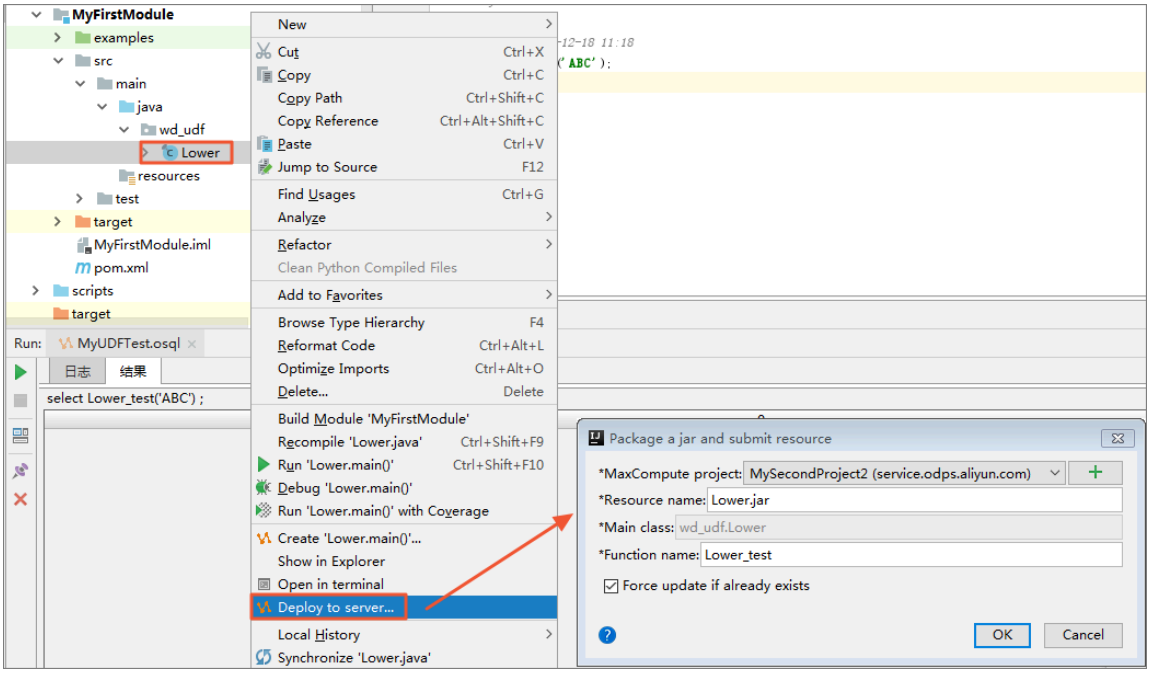
If you want to debug the Java UDF on your local PC, perform the operations mentioned in *Develop and debug UDF*.

3. **Register a MaxCompute UDF.**

   Right-click the UDF Java file and click Deploy to server. In the displayed dialog box, select the MaxCompute project to be registered for the MaxCompute project field, enter a function name. Then, click OK.

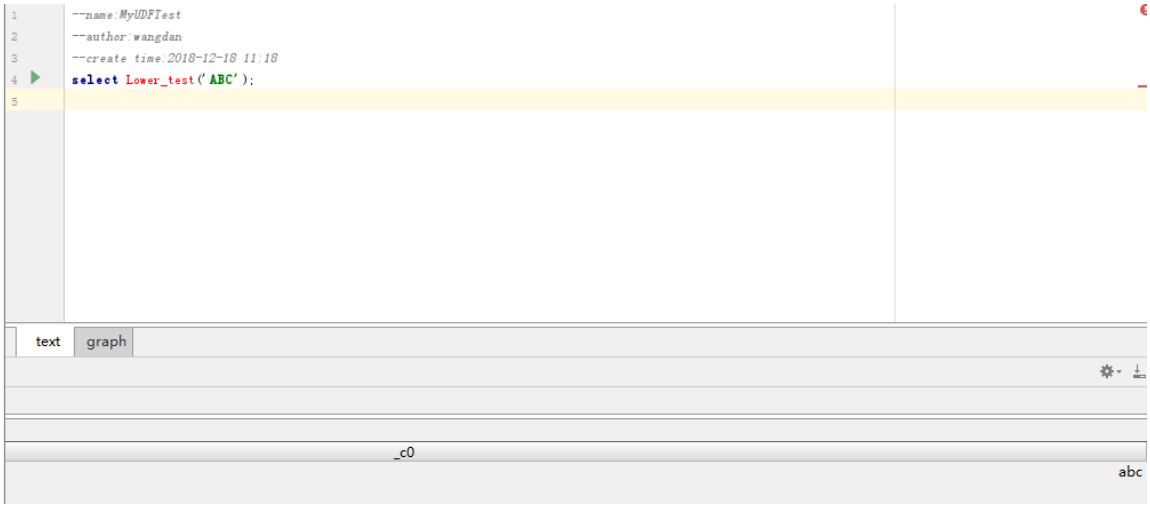
Note:

You can modify Resource name if you want to.



4. Test the UDF.

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



Note:

For more information about compile an SQL script in MaxCompute Studio, see *Write SQL*.

- Use the Eclipse plug-in to develop a UDF.

  1. Creating a project.

     Fro more information, see *Create a project*.

  2. Compile code.

     Compile the following code to achieve UDF functions according to the MaxCompute UDF frame specifications:

```
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 . toLowerCas e ();
}
}
```

     Name the .jar package my_lower.jar.

     > **Note:**
     >
     > - For more information about development and debugging code, see *UDF*.
     > - For more information about the SDK usage, see *UDF Summary*.

  3. Add resources.

     Specify the referenced UDF code before running the UDF. The code is added to MaxCompute in the form of resources. The Java UDF must be compiled into the a .jar package and added in MaxCompute as a JAR resource. The UDF framework then automatically loads the .jar package and runs the custom UDF.

     > **Note:**
     > MaxCompute MapReuce also defines "resource" in this way. For more information, see *MapReduce*.

     Run the following 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 .
```

```
-- Alternativ ely , use - f option directly to
overwrite original JAR resource .
```

4. Register a UDF.

   Run a command similar to the following:

   ```
   CREATE FUNCTION < function_n ame > AS < package_to _class
   > USING < resource_l ist >;
   ```

   Parameter description:

   - function_name: The UDF name, which is also referenced and used in SQL.
   - package_to_class: In the case of a Java UDF, the name contains the top-level package name and the fully qualified class name that implements the UDF class name. In case of a Python UDF, the name is the "Python script name.class name". The name must be quoted.
   - resource_list: The list of resources used by the UDF.
     - The resource list must include the resources in which the UDF code is located.
     - If your code reads resource files through the distributed cache interface, the list must include a list of resource files read by the UDF.
     - The name of the resource list consists of multiple resource names, which are separated by commas (,) and must be quoted.
     - If you need to specify the project to which the resource belongs, the command is `< project_na me >/ resources /< resource_n ame >`.

   After the .jar package is uploaded, MaxCompute can automatically obtain the code and run it. However, the UDF still cannot be used because MaxCompute lacks the UDF information. In this case, you need to register a unique UDF name in MaxCompute, and specify the mapping between the UDF name and the type of the target JAR resource.

   Run the following command:

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

   Note:

   - Each registered UDF name is unique, being similar to resource file names.

- Usually, only the owner of the project space can override a built-in UDF. However, if you use a custom UDF to override the built-in UDF, warning information is printed in the command summary after the SQL statement.

5. In SQL, verify the UDF by running the following command:

```
select test_lower (' A ') from my_test_ta ble ;
```

UDAF example

The registration method for a UDAF is similar to that for a UDF. You can use a UDAF by following the instructions provided in *Aggregate functions*. The following shows an example of the UDAF code for calculating the average value:

```
package org . alidata . odps . udf . examples ;
import com . aliyun . odps . io . LongWritab le ;
import com . aliyun . odps . io . Text ;
import com . aliyun . odps . io . Writable ;
import com . aliyun . odps . udf . Aggregator ;
import com . aliyun . odps . udf . UDFExcepti on ;
/**
* project : example_pr oject
* table : wc_in2
* partitions : p2 = 1 , p1 = 2
* columns : colc , colb , cola
*/
public class UDAFExampl e extends Aggregator {
 @ Override
  public void iterate ( Writable arg0 , Writable [] arg1 )
throws UDFExcepti on {
    LongWritab le result = ( LongWritab le ) arg0 ;
    for ( Writable item : arg1 ) {
      Text txt = ( Text ) item ;
      result . set ( result . get () + txt . getLength ());
    }
 }
 @ Override
  public void merge ( Writable arg0 , Writable arg1 )
throws UDFExcepti on {
    LongWritab le result = ( LongWritab le ) arg0 ;
    LongWritab le partial = ( LongWritab le ) arg1 ;
    result . set ( result . get () + partial . get ());
 }
 @ Override
  public Writable newBuffer () {
    return new LongWritab le ( 0L );
 }
 @ Override
  public Writable terminate ( Writable arg0 ) throws
UDFExcepti on {
    return arg0 ;
 }
```

```
}
```

UDTF example

The registration and use methods of a UDTF is similar to those of a UDF. The following is an code example:

```
 package   org . alidata . odps . udtf . examples ;
 import   com . aliyun . odps . udf . UDTF ;
 import   com . aliyun . odps . udf . UDTFCollec  tor ;
 import   com . aliyun . odps . udf . annotation . Resolve ;
 import   com . aliyun . odps . udf . UDFExcepti  on ;
//  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   UDFExcepti
 on  {
     String   a  = ( String )  args [ 0 ];
     Long   b  = ( Long )  args [ 1 ];
     for  ( String   t :  a . split ("\\ s +")) {
       forward ( t ,  b );
    }
  }
}
```

6 (Optional) Submit Graph jobs

This topic uses the *SSSP algorithm* as an example to describe how to submit Graph jobs.

Submitting a *Graph* job is similar to submitting a *MapReduce* job. Maven users can search `odps - sdk - graph` at *Maven library* to download the preferred Java SDK (available in different versions). The configuration information is as follows:

```
< dependency >
    < groupId > com . aliyun . odps </ groupId >
    < artifactId > odps - sdk - graph </ artifactId >
    < version > 0 . 20 . 7 </ version >
</ dependency >
```

Procedure

1. Log on to the console and run `odpscmd`.
2. Create input and output tables.

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

   For more information about table creation statements, see *Table operations*.
3. Upload data.

   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
   ```

   A space is used to separate two columns.

   ```
   tunnel   u  - fd  " "  sssp . txt   sssp_in ;
   ```
4. Compile an SSSP example.

   Compile and debug the *SSSP example* on your local PC by following the instructions provided in *Detailed introduction*. In this example, the code is packaged as the *odps - graph - example - sssp . jar* file.

   Note:

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

5. Add JAR resources.

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

Note:

For more information, see *Resources operations*.

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 commands are used to run MaxCompute Graph jobs in the same way as the commands for running *MapReduce* jobs.

When a Graph job is running, the corresponding instance ID, progress, and result summary are displayed in the command output, for example:

```
ID = 2013073016 0742915gl2 05u3
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_inpu t _ [BSP. sssp_in] _ bytes = 211
Graph_inpu t _ [BSP. sssp_in] _ records = 5
graph_outp ut_[bsp.sssp_out]_bytes = 161
Graph_outp ut _ [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 (millisecon ds) = 7
Load time (millisecon ds)= 8
Max superstep time (millisecon ds) = 14
Max time superstep = 0
Min superstep time (millisecon ds) = 5
Min time superstep = 2
Setup Time (millisecon ds) = 277
Shutdown Time (millisecon ds) = 20
Total superb time (millisecon ds) = 30
Total time (millisecon ds)= 344
OK
```

Note:

To use the Graph function, you only need to open and submit a *Graph job*.