MaxCompute

Quick Start

Quick Start

Create/Describe/Drop Table

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

You can create or delete tables by the following methods:

- By MaxCompute Studio. For details, see Visualization of Operating Tables.
- By DataWorks. For details, see Create a Table and Delete a Table.
- By the commonly used commands for the console.

This article introduce how to create, view and delete tables by the commonly used commands for console. For more information about the console installation, see Console.

Create Table

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
```

Descriptions are as follows:

The table name and column name are both case insensitive.

Exception will be thrown if a same name table has already existed. User must 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.

Only the following data types are supported: 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 now only string and bigint are supported. The partition value cannot have a double byte characters (such as Chinese), must begin with a letter a-z or A-Z, followed by letter or number. The name length cannot exceed 128 bytes. Allowed characters include: 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 reading.

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.

Currently, 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:

- For more information of creating table, see CREATE TABLE.
- For more information of partition operation, see Add/Remove Partition.
- For more information of lifecycle operation, see 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 must be partitioned by region and dt and the lifecycle 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;
```

Add Partition

After we create a partition table and need to import data into different partitions, we must 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');
```

View Table

After you create a table successfully, you can view the table information through 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;
```

The results are as follows:

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

If you want to view the information of test5, the two fields ‘pt’, ‘ds’ will only exist as two common columns, rather than the table partitions.

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');
```

Drop Table

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

For example, delete the table ‘test2’:

```
drop table test2;
```

For more introductions, see DROP TABLE.

Data Channel

MaxCompute provides several data import and export methods:

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

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

Tunnel Commands

Data Preparation

Suppose that we have 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 MaxCompute 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);
```

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

MaxCompute Studio

Make sure that you have installed MaxCompute Studio and configured Project Space Connection.

Data Preparation

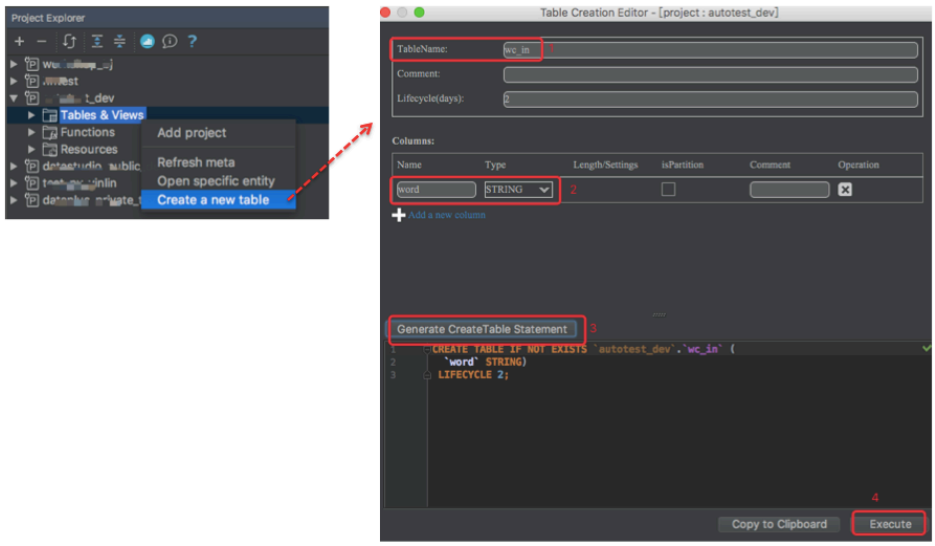
Suppose that we have prepared the local file wc_example.txt and its corresponding contents are shown as follows:

```
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 MaxCompute Table

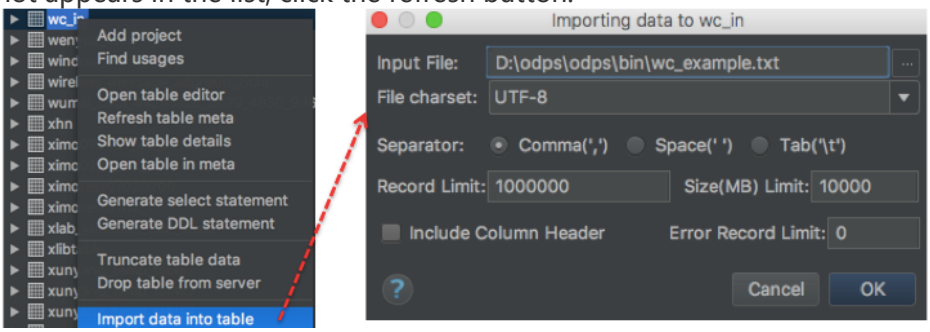
As we need to import the data mentioned above into a MaxCompute table, here we need to create a table at first. Right-click **tables&views** in the project and operate as follows:



If the statement is executed successfully , it means that the table has been created.

Upload Data Files

Right-click the table name you just created in the **tables&views** list in the project. If the table name not appears in the list, click the refresh button.



For more information, see Import and Export Data.

Tunnel SDK

On how to use SDK tunnel to upload data, the following simple scenario will be introduced.

Scenario Description

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

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
```

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

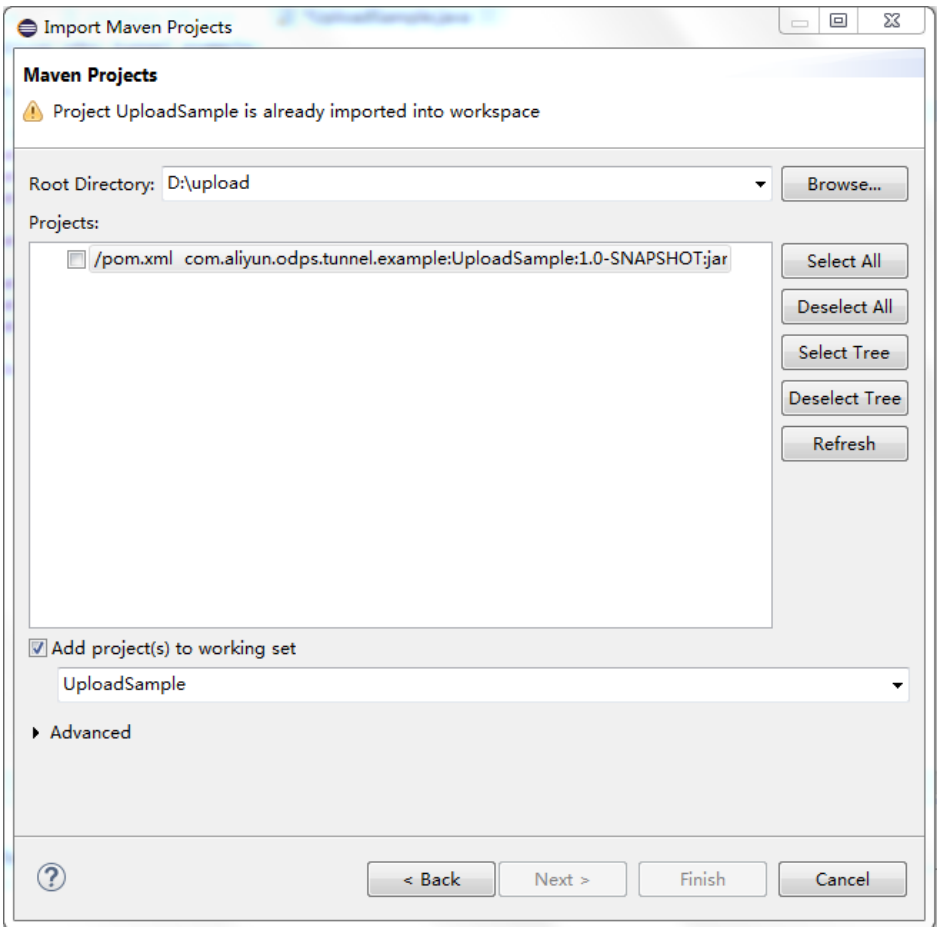
Compile and run:

Compile the program UploadSample:

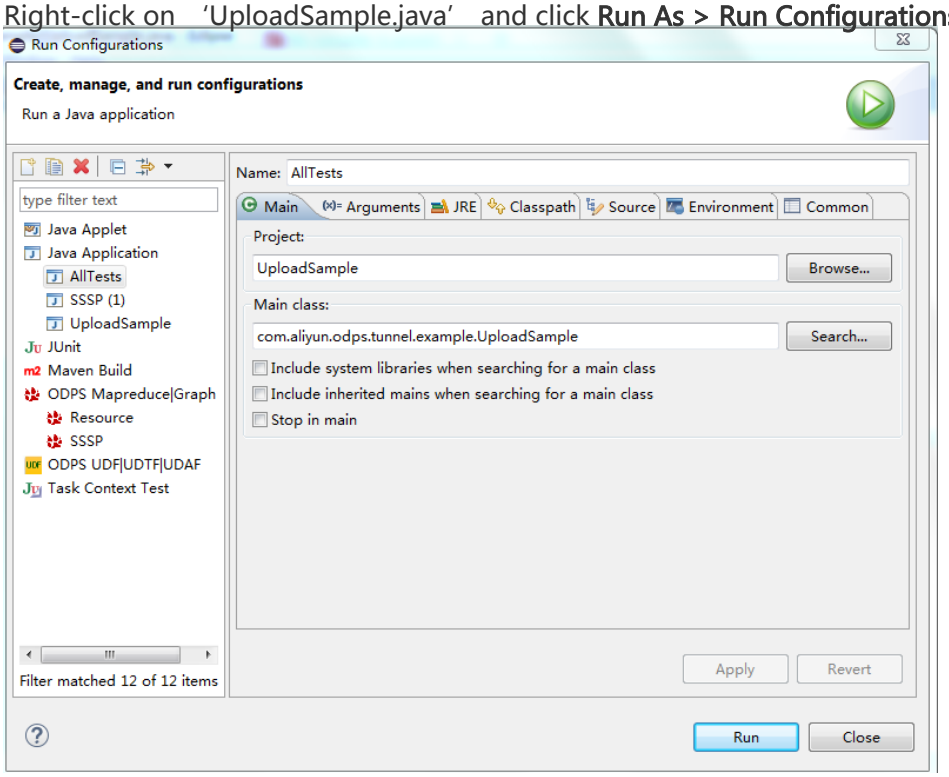
```
mvn package
```

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**.



Right-click on 'UploadSample.java' and click **Run As > Run Configurations**.



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

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

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 |
+----+------+----+----+
```

Notes:

- As an independent service in MaxCompute, Tunnel has exclusive access port provided for users.
- The intranet address in Alibaba Cloud accessing to MaxCompute:http://odps-ext.aliyun-inc.com/api.
- The Internet address accessing to MaxCompute: http://service.odps.aliyun.com/api.

Other Import Methods

In addition to MaxCompute Console and Tunnel Java SDK, data can also be imported through Alibaba Cloud DTplus products, open-source Sqoop, Fluentd, Flume, LogStash and so on. For more information, see Tools.

SQL

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

- Supports all kinds of operators.
- Manage tables, partitions and views through DDL statements.
- Query the records in the table through Select statements, and filter the records in the table through Where statements.
- Insert and update data through Insert statements.
- Supports Join statements to associate two tables. Supports the map join of small tables.
- Supports for computing through built-in functions and custom functions.
- Supports regular expressions.

This article briefly introduces problems need to be taken care of when you use MaxCompute SQL, and no longer gives an example of operation.

Notes

- 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.
- After you submit the MaxCompute jobs, jobs will be queued and scheduled for execution, and result for which can take time ranging from few seconds to several minutes. MaxCompute is suitable for handling batch operations. You can process massive data in one batch processing job.
- For the example of operation about SQL, see SQL.

DDL Statements

The basic DDL operations include creating tables, adding partitions, viewing tables or partitions information, modifying tables, deleting tables or partitions. For more information, see Create/Describe/Drop Table.

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 cannot 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' cannot 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 operation types: {LEFT OUTER|RIGHT OUTER|FULL OUTER|INNER} JOIN.
- Currently, MaxCompute SQL supports up to 16 concurrent Join operations.
- 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 will return all the results without deduplication. MaxCompute does not support to union two main query results. But you can do it on two subquery results.

> **Note:**
>
> The two Select queries connected by Union All, the number of columns, column names and column types must be strictly consistent. If the original names are inconsistent, you can set them the same name by alias.

Others

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

SQL Optimization Example

The Location of Where Condition in Join statement

When you join two tables, the Where condition of the main table can be written at the end of a statement, but the restriction condition about the partition in slave table can not be not written in the Where condition, and it is suggested to be written in the ON condition or subquery. The partition restrictions of main table can be written in Where condition(using subquery to filter in advance can be better).

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 will be run first, which leads to the data volume become larger and the performance decrease. Therefore, the second case should 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.

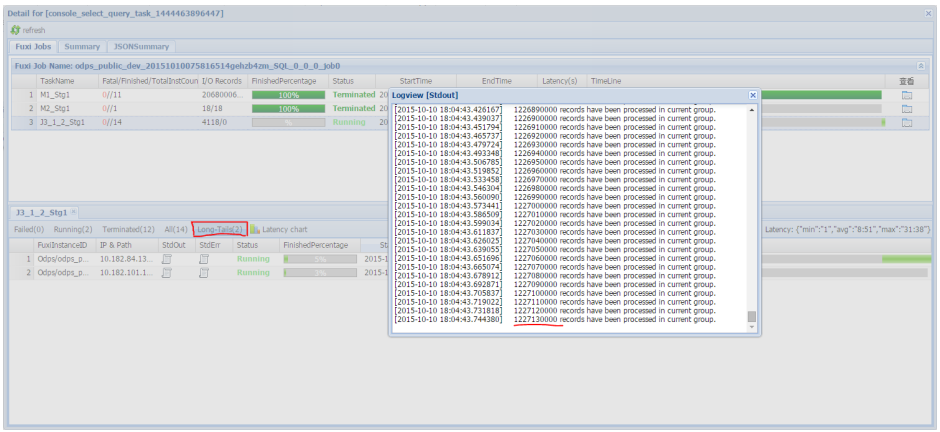
For more information about data skew optimization, see Optimize Long Tail Computing.

Data Skew Caused by Join

The reason for 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 webcosole page, and double click on 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 way :

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

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

- Handle the skewed key with an exclusive 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 fill in the random number before Join operation. An example is shown below:

```
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 can’ t get the key information that causes the data to skew, a general solution can be used to view the data skew.

```
select * from a join b on a.key=b.key; --This will Lead to data skew.
```

Now you can run the following statements:

```
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 join b.

Group by Skew

The reason for group by skew is the uneven distribution of keys operated b y group by.

Suppose that there are two fields(key and value) in table A, 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 webcosole page. To solve this problem, you need to set the anti-skew parameters before running SQL statement. set odps.sql.groupby.skewindata=true should be added into the SQL statement.

Data skew Caused by Incorrect Use of Dynamic Partitions

SQL to make tables dynamically partitioned will add a Reduce by default in MaxCompute, which is used to merge data from the same partition. The benefits are as follows:

- 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.

Due to the introduction of Reduce, long tail may appear if data in partitions is skewed. Because the same data can only be processed by a maximum of 10 Workers, so large volume of data can result in a long tail.

An example is as follows:

```
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, you need not to use dynamic partition. The statement can be modified into:

```
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, in general, each window function will form a Reduce job. If there are too many window functions, they will consume resources. In some specific scenarios, window functions can be optimized.

- The content after the over keyword must be exactly the same, such as grouping and sorting conditions.
- Multiple window functions are run on the same layer of SQL.

Window functions that meet the above two conditions are combined into a Reduce to implement. 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 below:

```
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 1000, the system will report an error such as ‘records returned from subquery exceeded limit of 1000’. At this time, 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)
```

Notes:

- If there is no Distinct keyword in the statement, and the result of the subquery c returns the same col1 value, it may cause larger number of results of table a.
- The Distinct subquery lead the whole query to fall into one Worker. If the subquery result 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, the Distinct keyword can be removed to improve performance.

Java UDF Development

MaxCompute UDF include: UDF, UDAF and UDTF. Usually these three kinds of functions are called '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>
```

In general, you can develop Java UDF through the following ways:

- Use MaxCompute Studio to complete the whole process of Java UDF development.
- Use the Eclipse plug-in to develop and debug the Java UDF code, export the jar package, and then add resources through command or DataWorks product. Register the function at last.

The code examples of UDF, UDAF, and UDTF will be given separately in this section. And the steps of UDF development will also be given(The steps of UDAF and UDTF development are the same as UDF).

Notes:

- For commands or statements about UDF registration and logout, see Function.
- For the data type mapping relations between Java and MaxCompute, see UDF.

UDF Example

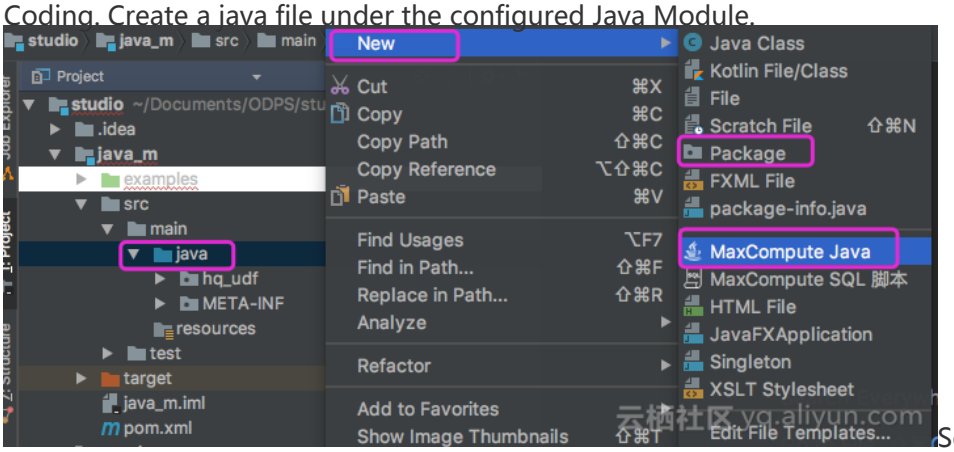
An entire development example of UDF to realize character lowercase conversion will be given below.

Using MaxCompute Studio

Tools and development environment preparation. Here we assume that the environment has been prepared, including:

- Studio installation
- Creating MaxCompute project link in Studio
- Creating MaxCompute Java Module

Coding. Create a java file under the configured Java Module.

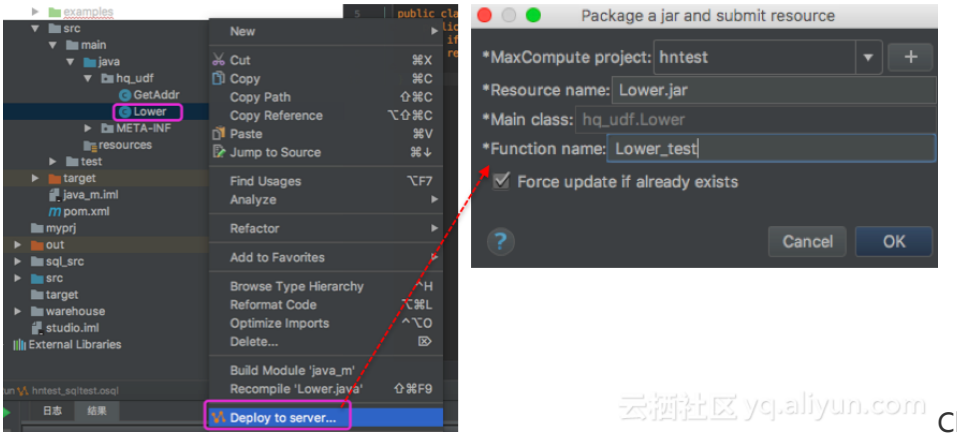


Select **MaxCompute java** and enter '*package name.filename*' in **Name** text box, select UDF for **Kind**, and then edit the code:

```
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.toLowerCase();
}
}
```

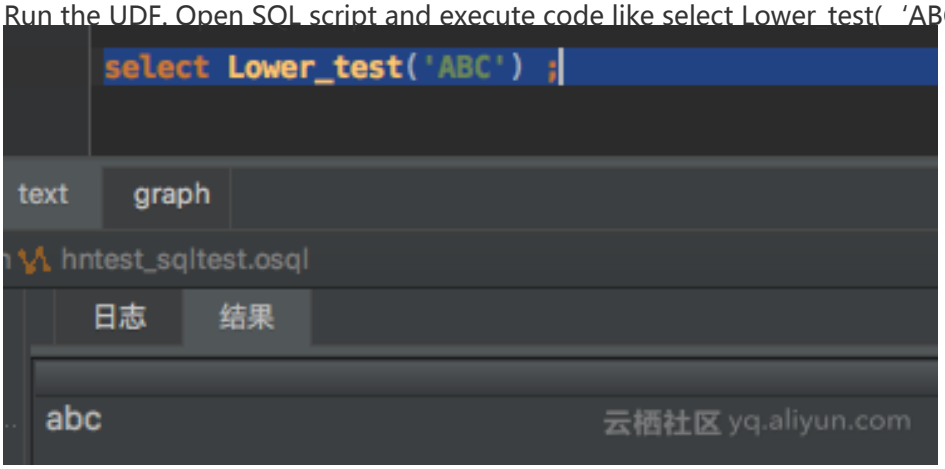
Note: If you need to debug Java UDF locally, see **Develop and Debug UDF**.

Register UDF. Right-click on the UDF's java file, select **Deploy to server**, select MaxCompute project that you want to register in the pop-up box, enter function name and modify the resource name.



Click **OK** and successful information appears .

Run the UDF. Open SQL script and execute code like select Lower_test('ABC').



Note: For Writing SQL scripts in Studio, see Write SQL Script.

Using Eclipse Plug-in

1. Create project. Suppose that a MaxCompute (formerly ODPS) project has been created in the Eclipse plug-in. For details, see Creating a MaxCompute Project.

   Coding. Realize the UDF function in accordance with ODPS UDF framework and do compiling. A simple coding instance is as follows:

```
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 this jar package ‘my_lower.jar’.

Note:

- For the information of SDK, see UDF SDK.
- For detailed code development and debugging introduction, see UDF.

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. UDF framework will load jar package automatically and run UDF. MaxCompute MapReduce also describes the use of resource.
Run 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 can obtain user’s code and run it after the jar package has been uploaded. But at this point, this UDF cannot be used because MaxCompute does not have any information about this UDF. It 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. For registering UDF, see 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);
}
}
}
```

MaxCompute provides a lot of built-in functions to meet your computing needs. However, you can also create custom functions to meet more computing needs. For more information, see Create UDFs.

MapReduce

This section is to introduce how to run the example program ‘MapReduce WordCount’ rapidly after the MaxCompute console has already been installed.

Note:

The users who use Maven can search “odps-sdk-mapred” from Maven Library to get different versions of Java SDK. The configuration is shown as follows:

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

Prerequisites

- To compile and run MapReduce， make sure there is JDK 1.6 or later version installed on your machine.

- For installing MaxCompute console quickly, see Quick Start. For the use method of MaxCompute console, see Console;

Procedures

Next we will introduce the operation step by step.

1. After the console has been installed and configured, open odpscmd.bat and enter the specified project space.
2. Create input and output tables.

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

   For the SQL statement to create table, see CREATE TABLE.
3. Upload data.
   You can upload data by the following two ways:
   - 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;
```

   Write MapReduce program and compile it.
   MaxCompute 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 a MaxCompute project in Eclipse and then write MapReduce program. After local debugging is passed, upload the compiled program to ODPS. For more information, please see MapReduce Eclipse Plug-in.

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;
```

6. 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;
```

7. Check the running result on MaxCompute console:

```
select * from wc_out;
```

Note:

If any resources are used in java program, make sure add ‘-resources’ parameters. For more information about jar commands, see Jar Commands.

Graph

The submitting method of Graph job is similar to MapReduce. Users who use Maven can search “odps-sdk-graph” from Maven Library to get different versions of Java SDK. The related configuration 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.

Procedures

Enter the console and run “odpscmd”.

Create input tables and output tables.

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

Note: For more statements to create table, see DDL.

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
```

Take the “tab” button as the separator of two columns to run the tunnel command to upload 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. This example assumes that the code is packaged as “odps-graph-example-sssp.jar”.

   Note: You only need to package SSSP code into “odps-graph-example-sssp.jar”.

   Add Jar.

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

   Note: For resource creation, see Resource Operation.

   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 MaxCompute Graph. Its using 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: For a user who needs to use graph, apply the service of submission graph calculation jobs.