MaxCompute

Quick Start

Quick Start

MapReduce

The following section assumes the MaxCompute console has been installed to use the example ‘MapReduce WordCount’. 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>0.20.7</version>
</dependency>
```

Note:

- To compile and run MapReduce requires JDK1.6.
- For installing the MaxCompute console, see Quick Start, and for using the MaxCompute client, see Console;

Procedure

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

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

Use Tunnel Commands to upload data:

```
tunnel u 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;
```

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.

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;
```

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;
```

Check the running result on ODPS console:

```
select * from wc_out;
```

Note:

If other resources are used the in java program, you must add ‘-resources’ parameters. For more information about JAR commands, see Jar Commands.

Graph

Submitting a Graph job is similar to submitting a job using MapReduce. The following section uses SSSP Algorithm as an example to show how to submit Graph jobs. Maven users can search “odps-sdk-graph” from Maven Library to get the required 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>
```

To run a Graph job using the SSSP algorithm, follow these steps.

Enter the console and run “odpscmd”.

Create Input Table and Output Table.

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

Note:

For the statement to create table, see SQL Create.

Upload Data.
The contents of local data is as follows:

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

“tab” button is taken as the separator of two columns.Upload the data:

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

Write SSSP example.

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

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

Add JAR package.

```
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 use method is consistent with MapReduce.

When GRAPH job is running, 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 superstep time (milliseconds)=30
Total time (milliseconds)=344
OK
```

NOTE:

New users are recommended to submit their Graph job, and provide the name of the MaxCompute project and its usage scenario, to Alibaba Cloud. If the review of the job is successful, the Graph function is opened.

Create, view, and delete a table

A user can use MaxCompute services once they are added to a project, and granted the corresponding 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 Tables.

DataWorks. For more information, see Create a Table and Delete a Table.

The DTplus console.

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

Create 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.

An exception is thrown if duplicate table name creation is attempted. User must specify the option [if not exists] to override the error. If the option [if not exists] is specified, regardless of if there are tables with the same name, and even if the source table structure and the target table structure are inconsistent, all returns are successful. The Meta information of 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 '_'.

Use 'Partitioned by' to specify the partition. Only STRING and BIGINT are supported. The conventions of the partition value are as follows: The value can be up to 128 bytes in length and can contain letters, numbers, and 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 need 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 1024 bytes in length.

Lifecycle indicates the lifecycle of the table. The unit is 'day'. 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.

Notes:

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.
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. Next, create a table named user that 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 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 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_col_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 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;
```

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

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

Drop 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_col_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 table

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

For example, to delete the table ‘test2’:

```
drop table test2;
```

For more information, see Drop Table.

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

Data preparation

In the following example, the contents of a local file ‘wc_example.txt’ are as follows:

```
Hello World!
```

The file is saved into the directory: D:\odps\odps\bin.

Create a MaxCompute table

To import the data created in the preceding step, a MaxCompute table must be created. An example is as follows:

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

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;
```

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
QWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnByb2plY3RzL29kcHNfcHVibGljX2Rldi9pbnN0YW5jZXMvMjAxNTA5MTgxMTA1MDE4NjRnNXo5YzYiXX1dLC
+------+
| word |
+------+
| Hello World! |
+------+
```

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.

Data Preparation

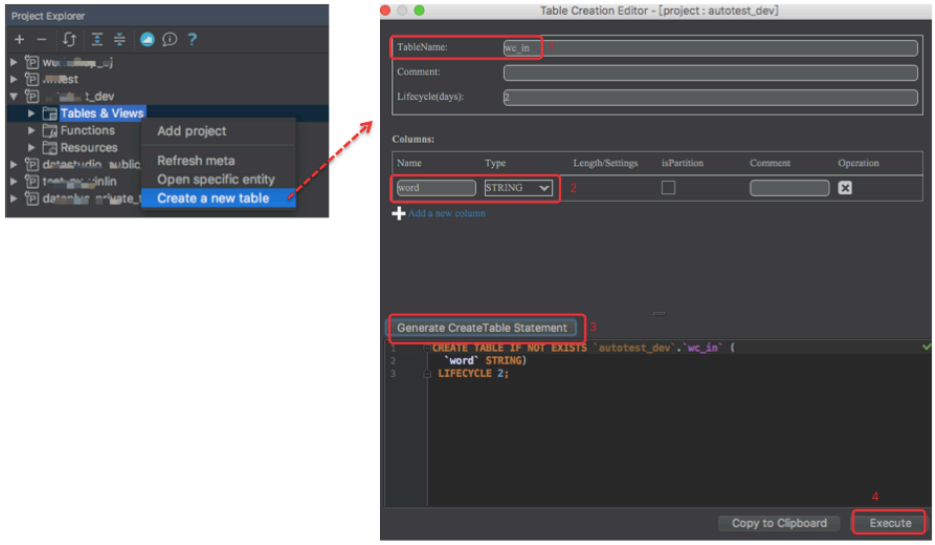
In the following example, the contents of a local file 'wc_example.txt' are as follows:

```
Hello World!
```

The file is saved into the directory: D:\odps\odps\bin.

Create a MaxCompute table

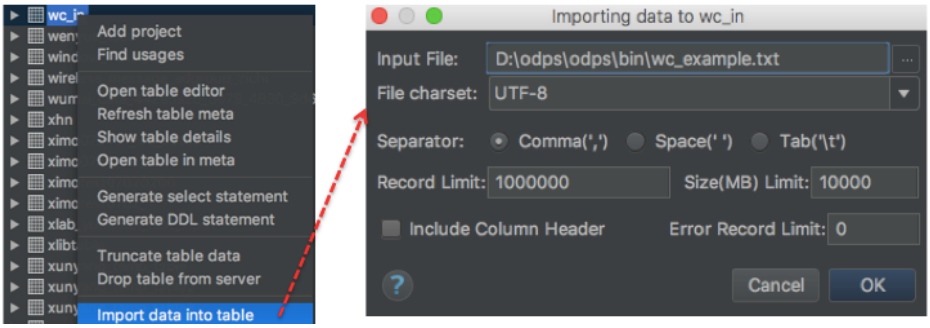
To import the data created in the preceding step, a MaxCompute table must be created first. Right-click **tables&views** in the project and operate as follows:



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

Upload data files

Right-click the table name created in the preceding **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

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

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:

The configuration of AccessKeyId and AccessKeySecret is ignored in the preceding example. In actual operation, apply the required AccessKeyId and AccessKeySecret.

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-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>
```

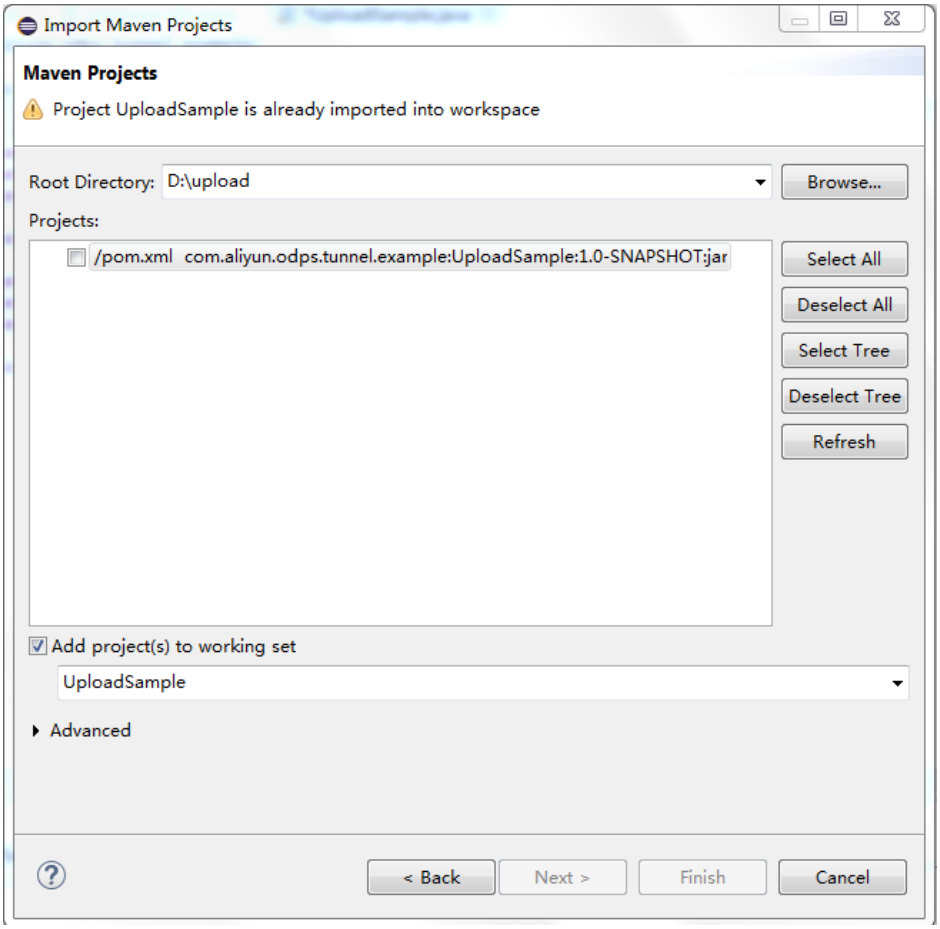
Compile and run:

Compile the program UploadSample:

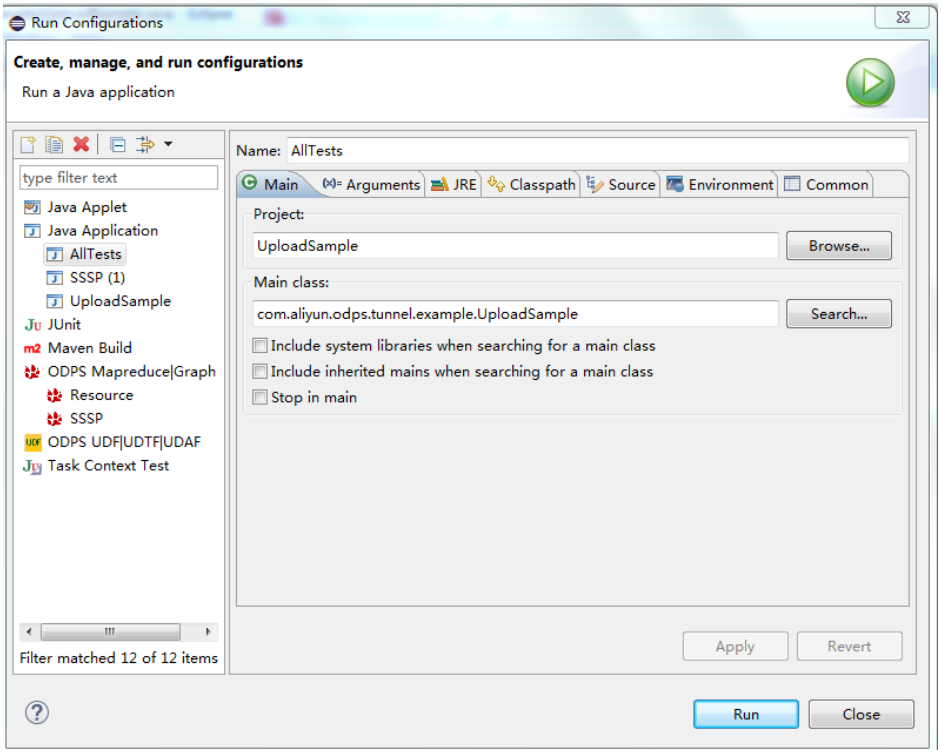
```
mvn package
```

Run the program UploadSample. Here, Eclipse is used to import the 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 |
+----+------+----+----+
```

Note:

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, Sqoop, Fluentd, Flume, LogStash, and more Tools.

Run SQL

MaxCompute SQL does not support transactions, index, or UPDATE/DELETE operations. The MaxCompute SQL syntax differs from Oracle and MySQL, notably, you cannot seamlessly migrate SQL statements of other databases into MaxCompute. After you submit MaxCompute jobs, the jobs can be queued and scheduled for execution. MaxCompute SQL can complete queries at the second- to millisecond-level. You can process massive data in one batch processing job.

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.

```
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 six 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.

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.

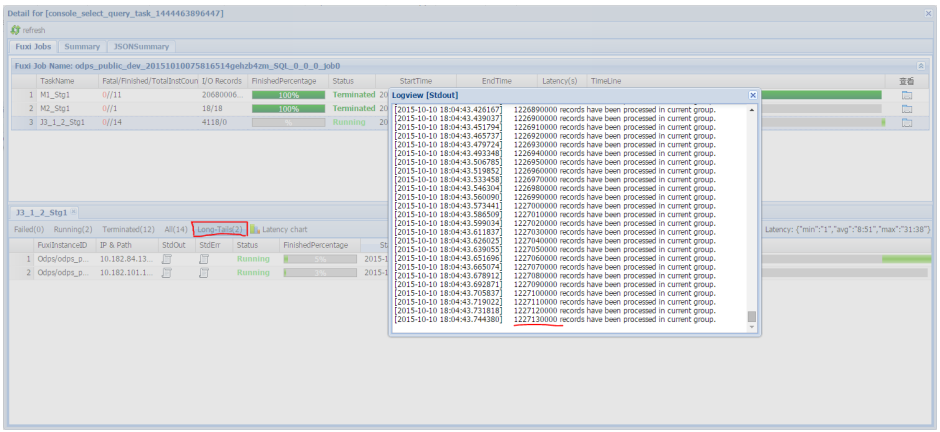
For more information about data skew optimization, see Optimize long tail computing.

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.

```
select * from a join b on a.key=b.key; --This Leads 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 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.

Java UDF Development

Summary

MaxCompute user-defined functions (UDFs) include User Defined Scalar Function (UDF), User Defined Aggregation Function (UDAF), and User Defined Table Valued Function (UDTF). 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-public</version>
</dependency>
```

Note:

Currently, UDF only supports Java language. If you want to write a UDF program, you can upload UDF code into the project through Add Resource and create UDF through Create Function.

To run a UDF with Eclipse, see UDF Eclipse Plug-in Introduction.

If you must use UDF, you must submit application in ticket system, providing the MaxCompute project name and application scenario. When your application is passed, you can use the UDF.

UDF Example

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

- Coding: realize the UDF function in accordance with MaxCompute UDF framework and compile the code. An example 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 the JAR package 'my_lower.jar'.

Note:

For more information about the SDK, see UDF SDK.

Add resource: Specify the referenced UDF code before running UDF. 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. 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.
```

- 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. For registering a UDF, see Create

Function. Next, 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 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 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 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);
}
}
}
```