阿里云
MaxCompute

快速入门

文档版本：20181109

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止：** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告：** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明：** 您也可以通过按**Ctrl** + **A**选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 创建/查看/删除表

当您被添加到项目空间并被赋予建表等权限后，即可操作MaxCompute。由于在MaxCompute中的操作对象（输入、输出）都是表，所以在处理数据之前，首先要创建表、分区。

创建/删除表的方式有以下几种：

- 通过MaxCompute Studio实现，详情请参见可视化创建/修改/删除表。
- 通过DataWorks实现，详情请参见创建表和删除表。
- 通过客户端常用命令实现。

本文将为您介绍如何通过客户端常用命令进行创建表、查看表和删除表的操作，客户端的安装请参见安装并配置客户端。

创建表

建表语句如下所示：

```
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
```

建表语句说明：

- 表名与列名均对大小写不敏感。
- 在创建表时，如果不指定if not exists选项而存在同名表，则返回出错；若指定此选项，则无论是否存在同名表，即使原表结构与要创建的目标表结构不一致，均返回成功。已存在的同名表的元信息不会被改动。
- 数据类型：包括Bigint、Double、Boolean、Datetime、Decimal和String等多种数据类型。
- 表名，列名中不能有特殊字符，只能用英文的a-z，A-Z及数字和下划线 _，且以字母开头，名称的长度不超过128字节。
- Partitioned by：指定表的分区字段，目前仅支持String类型，其他类型行为未定义。分区值不可以有双字节字符（如中文），必须是以英文字母a-z，A-Z开始后可跟字母数字，名称的长度不超过128字节。允许的字符包括空格、冒号（:）、下划线（_）、美元符（$）、井号（#）、点（.）、感叹号（!）和（@），出现其他字符行为未定义。例如（\t），（\n），（/）等 。当

利用分区字段对表进行分区时，新增分区、更新分区内数据和读取分区数据均不需要做全表扫描，可以提高处理效率。

- 注释内容是长度不超过1024字节的有效字符串。
- lifecycle指明此表的生命周期，单位：天。`create table like`语句不会复制源表的生命周期属性。
- 目前，在表中建的分区层次不能超过6级。一个表允许的分区个数支持按照具体的project配置，默认60,000个。

说明：

- 创建表的详细介绍请参见表操作。
- 添加分区请参见分区/列操作。
- 生命周期的修改请参见修改表的生命周期属性。

创建表示例如下：

```
create table test1 (key string); -- 创建非分区表，表名 test1，字段名 key ，数据类型 string。
create table test2 (key bigint) partitioned by (pt string, ds string);  --创建分区表
create table test3 (key boolean) partitioned by (pt string, ds string ) lifecycle 100; -- 创建带有生命周期的表
create table test4 like test3; -- 除生命周期属性外，test3 的其他属性（字段类型，分区类型等）均与 test4 完全一致
create table test5 as select * from test2; -- 这个操作会创建 test5，但分区，生命周期信息不会被拷贝到目标表中。
-- 此操作仅会将 test2 的数据复制到 test5 中（如果 test2 有数据的话，此示例中 test2 为空表，后续章节会介绍数据导入）。
```

创建表的场景如下：

假设需要创建一张用户表user，包括如下信息：

- user_id bigint类型：用户标识，唯一标识一个用户。
- gender bigint类型：性别（0，未知；1，男；2，女）。
- age bigint：用户年龄。

按照Region（区域）和dt（日期）进行分区，生命周期为365天。

建表语句如下所示：

```
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;
```

创建分区

当创建一张分区表之后，为了往该表里面导入不同分区数据，您需要创建分区。命令如下：

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

如上述示例，给用户表user添加区域为hangzhou，日期为20150923的分区，句子显示如下：

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

查看表信息

当创建表成功之后，您可以通过`desc <table_name>;`命令查看表的信息。

您可执行命令`desc test3;`查看上述示例中表test3的信息。

结果显示如下：

```
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 |      |
```

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

您可执行命令desc test4;查看上述示例中表test4的信息。

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

您会发现，除生命周期属性外，test3的其他属性（字段类型、分区类型等）均与test4完全一致。查看表信息的更多详情请参见表操作。

您如果查看test5的表信息，pt、ds两个字段仅会作为普通列存在，而不是表的分区 。

删除分区

删除分区的命令如下所示：

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

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

比如删除区域为hangzhou，日期为20150923的分区，语句如下所示：

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

删除表

删除表的命令如下所示：

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

删除test2表的示例如下：

```
drop table test2;
```

更多详情请参见表操作删除表。

2 导入数据

本文向您详细介绍Tunnel命令导入数据、MaxCompute Studio导入数据以及其他一些导入数据到MaxCompute的方式。

MaxCompute提供多种数据导入导出方式，如下所示：

- 直接在客户端使用*Tunnel*命令。
- 通过MaxCompute Studio工具可视化方式实现本地数据文件导入导出，详情请参见导入导出数据。
- 通过*Tunnel*提供的SDK自行编写Java工具。
- 通过Flume及Fluentd插件方式导入。
- 通过DataWorks对数据导入和导出，详情请参见数据集成概述。

导出数据请参见*Tunnel*命令操作中Download的相关命令。

Tunnel命令导入数据

1. 准备数据

   假设您已准备本地文件wc_example.txt，本地存放路径为D:\odps\odps\bin，内容如下：

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

2. 创建MaxCompute表

   您需要把上面的数据导入到MaxCompute的一张表中，所以需要创建MaxCompute表。

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

3. 执行tunnel命令

   输入表创建成功后，可以在MaxCompute客户端输入Tunnel命令进行数据的导入，如下所示：

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

4. 执行成功后，查看表wc_in的记录。

说明：

- 有关Tunnel命令的更多详细介绍，例如如何将数据导入分区表等，请参见*Tunnel*操作。
- 当表中含有多个列时，可以通过**-fd**参数指定列分隔符。

MaxCompute Studio导入数据

在使用MaxCompute Studio导入数据前，您需要先安装*MaxCompute Studio*，并配置项目空间链接。

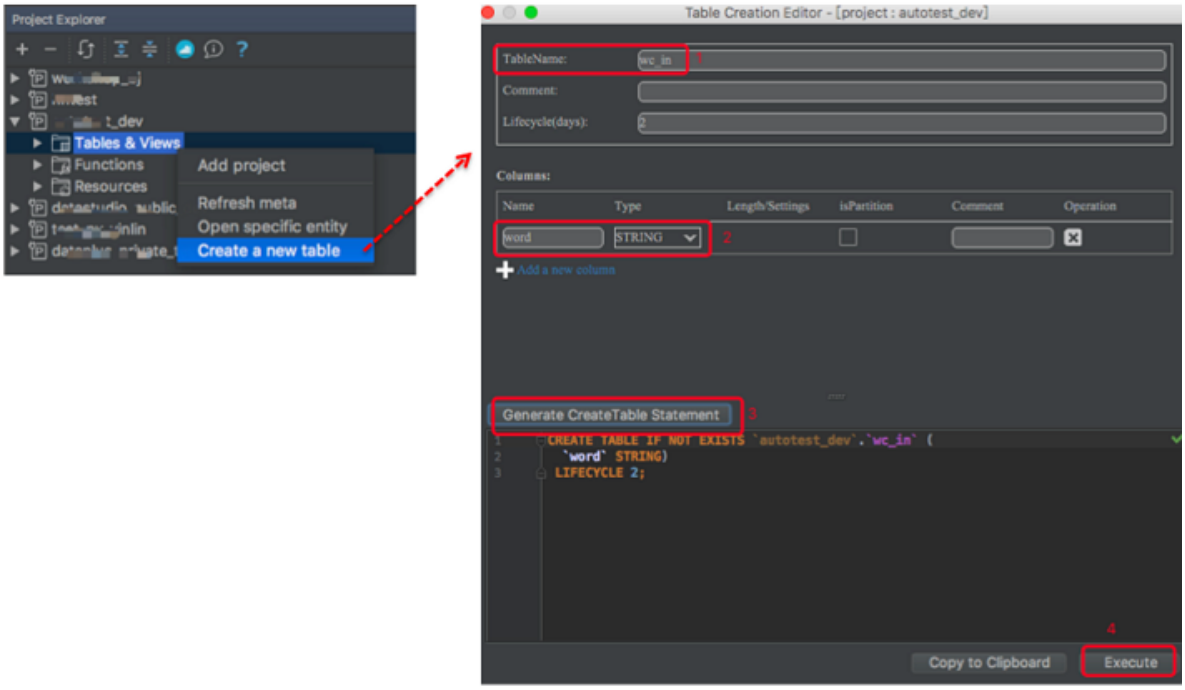
1. 准备数据。

   假设您已准备本地文件wc_example.txt，本地存放路径为D:\odps\odps\bin，内容如下：

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

2. 创建MaxCompute表。

   您需要把上面的数据导入到MaxCompute的一张表中，所以需要创建MaxCompute表。右键单击项目的**Tables&Views**列表：

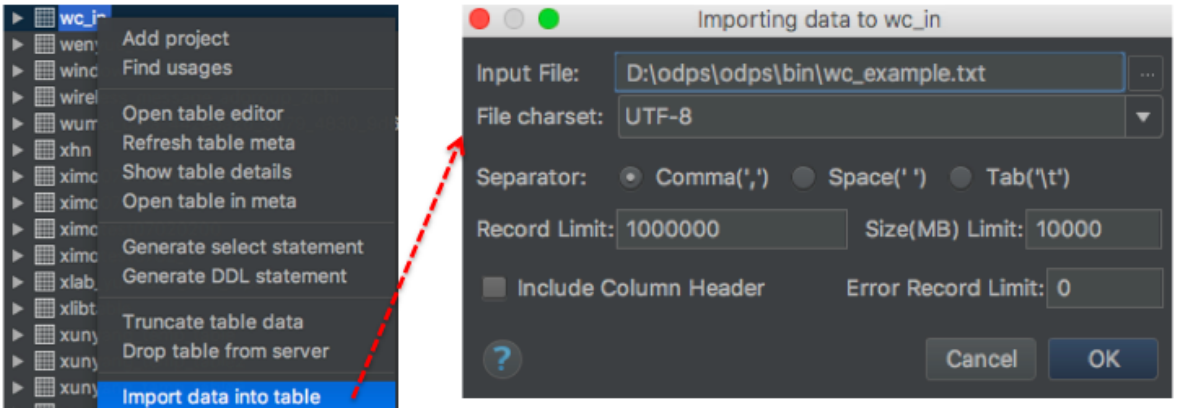


执行成功，则建表成功。

3. 上传数据文件。

   右键单击tables&views列表中新建的表wc_in

> 说明：
> 如果新建表未显示，请单击刷新。



Tunnel SDK

下文将通过场景示例，为您介绍如何利用*Tunnel SDK*上传数据。

场景描述

上传数据到MaxCompute，其中，项目空间为odps_public_dev，表名为tunnel_sample_test，分区为pt=20150801，dt=hangzhou。

操作步骤

1. 创建表，添加分区，SQL语句如下所示：

```
CREATE TABLE IF NOT EXISTS tunnel_sample_test(
id STRING,
name STRING)
PARTITIONED BY (pt STRING, dt STRING); --创建表
ALTER TABLE tunnel_sample_test
ADD IF NOT EXISTS PARTITION (pt='20150801',dt='hangzhou'); --添加分区
```

2. 创建UploadSample的工程目录结构，如下所示：

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

目录说明：

- pom.xml：maven工程文件。
- UploadSample：Tunnel源文件。

3. 编写UploadSample程序。代码如下所示：

```
package com.aliyun.odps.tunnel.example;
import java.io.IOException;
import java.util.Date;

import com.aliyun.odps.Column;
import com.aliyun.odps.Odps;
import com.aliyun.odps.PartitionSpec;
import com.aliyun.odps.TableSchema;
import com.aliyun.odps.account.Account;
import com.aliyun.odps.account.AliyunAccount;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.RecordWriter;
import com.aliyun.odps.tunnel.TableTunnel;
import com.aliyun.odps.tunnel.TunnelException;
import com.aliyun.odps.tunnel.TableTunnel.UploadSession;

public class UploadSample {
  private static String accessId = "####";
  private static String accessKey = "####";
  private static String tunnelUrl = "http://dt.odps.aliyun.com";
```

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

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

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


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


        TableSchema schema = uploadSession.getSchema();
        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();
        }
    }
}
```

说明：

此处省略了AccessKeyID和AccessKeySecret的配置，实际运行时请换上您自己的相关信息。

4. 配置pom.xml文件。如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.
apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.aliyun.odps.tunnel.example</groupId>
<artifactId>UploadSample</artifactId>
<version>1.0-SNAPSHOT</version>
<dependencies>
  <dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>odps-sdk-core</artifactId>
    <version>0.20.7-public</version>
  </dependency>
</dependencies>
<repositories>
  <repository>
  <id>alibaba</id>
  <name>alibaba Repository</name>
  <url>http://mvnrepo.alibaba-inc.com/nexus/content/groups/public/</
url>
  </repository>
</repositories>
</project>
```

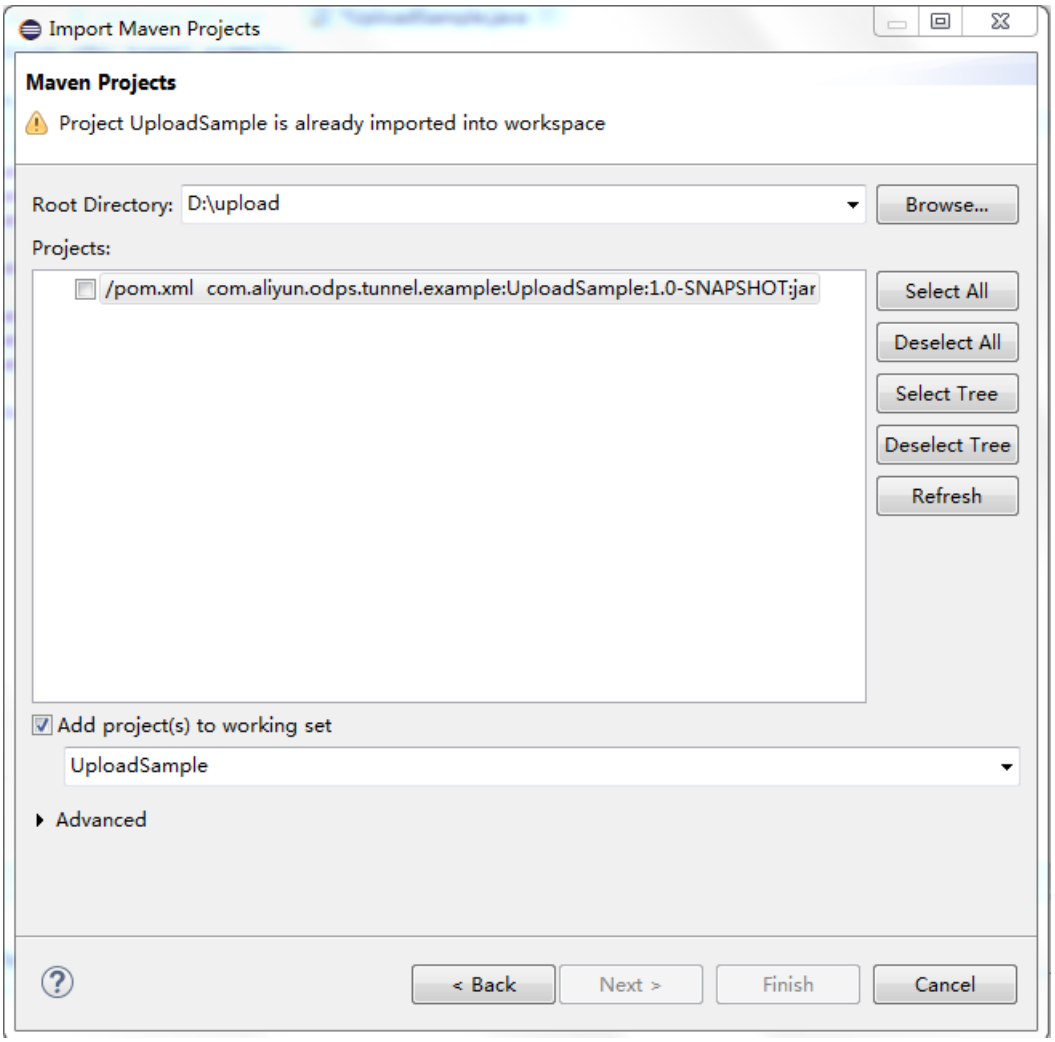
5. 编译与运行。

   编译UploadSample工程，如下所示：

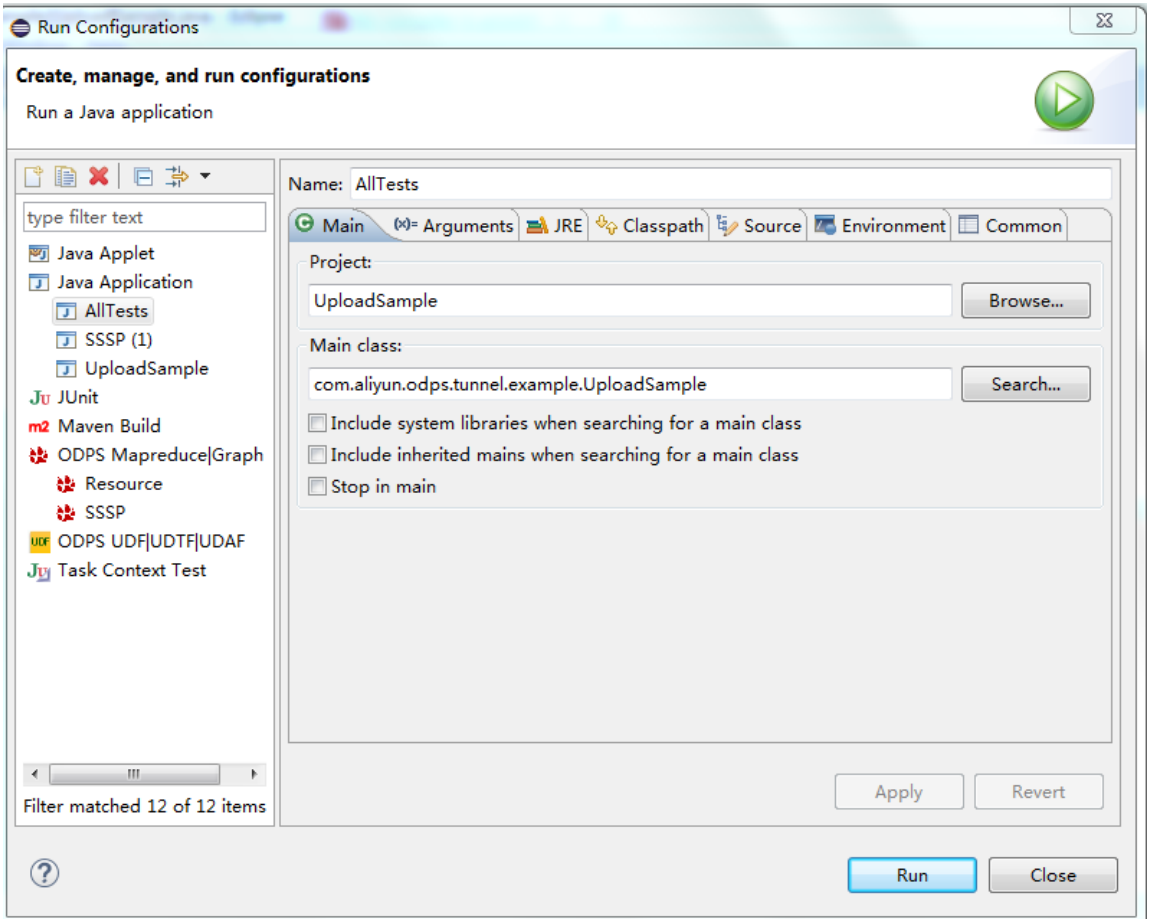
   ```
   mvn package
   ```

   运行UploadSample程序，此处使用eclipse导入maven project。

   1. 右键单击**java**工程并导航至**Import** > **Maven** > **Existing Maven Projects**， 设置如下：



2. 右键单击**UploadSample.java** 并单击**Run As > Run Configurations**，如下所示：



3. 单击**Run** 运行成功，控制台显示如下：

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

6. 查看运行结果。

   您在客户端输入如下语句，即可查看运行结果。

```
select * from tunnel_sample_test;
```

   显示结果如下：

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

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

说明：

- Tunnel作为MaxCompute中一个独立的服务，有专属的访问端口提供给大家。当您在阿里云内网环境中，使用Tunnel内网连接下载数据时，MaxCompute不会将该操作产生的流量计入计费。此外内网地址仅对上海域的云产品有效。
- MaxCompute阿里云内网和公网访问地址请参见访问域名和数据中心。

其他导入方式

除了通过客户端及Tunnel Java SDK导入数据，阿里云数加数据集成、开源的Sqoop、Fluentd、Flume、LogStash 等工具都可以进行数据导入到MaxCompute，详情请参见数据上传下载-工具介绍。

3 运行SQL

您对SQL语法可能并不陌生，MaxCompute SQL用于查询和分析MaxCompute中的大规模数据，主要功能如下所示。

- 支持各类运算符。
- 通过DDL语句对表、分区以及视图进行管理。
- 通过Select语句查询表中的记录，通过Where语句过滤表中的记录。
- 通过Insert语句插入数据、更新数据。
- 通过等值连接Join操作，支持两张表的关联，并支持多张小表的Mapjoin。
- 支持通过内置函数和自定义函数来进行计算。
- 支持正则表达式。

本文将为您简单介绍MaxCompute SQL使用中需要注意的问题，不再做操作示例。

说明：

- MaxCompute SQL不支持事务、索引及Update/Delete等操作，同时MaxCompute的SQL语法与Oracle，MySQL有一定差别，您无法将其他数据库中的SQL语句无缝迁移到MaxCompute上来，更多差异请参见与其他SQL语法的差异。
- 在使用方式上，MaxCompute作业提交后会有几十秒到数分钟不等的排队调度，所以适合处理跑批作业，一次作业批量处理海量数据，不适合直接对接需要每秒处理几千至数万笔事务的前台业务系统。
- 关于SQL操作的详细示例，请参见SQL模块。

DDL语句

简单的DDL操作包括创建表、添加分区、查看表和分区信息、修改表、删除表和分区，更多详情请参见创建/查看/删除表。

Select语句

- group by语句的key可以是输入表的列名，也可以是由输入表的列构成的表达式，不可以是Select语句的输出列。

```
select substr(col2, 2) from tbl group by substr(col2, 2);-- 可以，group by的key可以是输入表的列构成的表达式；
 select col2 from tbl group by substr(col2, 2); -- 不可以，group by的key不在select语句的列中；
```

```
 select substr(col2, 2) as c from tbl group by c; -- 不可以，group by
的key 不可以是列的别名，即select语句的输出列；
```

之所以有这样的限制，是因为在通常的SQL解析中，group by的操作先于Select操作，因此group by只能接受输入表的列或表达式为key。

- order by必须与limit连用。
- sort by前必须加distribute by。
- order by/sort by/distribute by的key必须是Select语句的输出列，即列的别名。

```
select col2 as c from tbl order by col1 limit 100 --不可以，order by的
key不是select语句的输出列，即列的别名
select col2 from tbl order by col2 limit 100; -- 可以，当select语句的输
出列没有别名时，使用列名作为别名。
```

之所以有这样的限制，是因为在通常的SQL解析中，order by/sort by/distribute by后于Select操作，因此它们只能接受Select语句的输出列为key。

Insert语句

- 向某个分区插入数据时，分区列不可以出现在Select列表中。

```
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;
  -- 报错返回，sale_date，region为分区列，不可以出现在静态分区的insert语句
中。
```

- 动态分区插入时，动态分区列必须在Select列表中。

```
insert overwrite table sale_detail_dypart partition (sale_date='2013
', region)
select shop_name,customer_id,total_price from sale_detail;
--失败返回，动态分区插入时，动态分区列必须在select列表中。
```

Join操作

- MaxCompute SQL支持的Join操作类型包括{LEFT OUTER|RIGHT OUTER|FULL OUTER|INNER} JOIN。
- 目前最多支持16个并发Join操作。
- 在Mapjoin中，最多支持8张小表的Mapjoin。

Union All

Union All可以把多个Select操作返回的结果，联合成一个数据集。它会返回所有的结果，但是不会执行去重。MaxCompute不支持直接对顶级的两个查询结果进行Union操作，需要写成子查询的形式。

> 说明：
>
> - Union All连接的两个Select查询语句，两个Select的列个数、列名称、列类型必须严格一致。
> - 如果原名称不一致，可以通过别名设置成相同的名称。

其他

- MaxCompute SQL目前最多支持128个并发Union操作。
- 最多支持128个并发insert overwrite/into操作。

MaxCompute SQL的更多限制请参见*SQL*限制项汇总。

SQL优化示例

- **Join语句中Where条件的位置**

  当两个表进行Join操作时，主表的Where限制可以写在最后，但从表分区限制条件不要写在Where条件中，建议写在ON条件或者子查询中。主表的分区限制条件可以写在Where条件中（最好先用子查询过滤）。示例如下：

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

  第二个语句会先Join，后进行分区裁剪，数据量变大，性能下降。在实际使用过程中，应该尽量避免第二种用法。

- **数据倾斜**

  产生数据倾斜的根本原因是有少数Worker处理的数据量远远超过其他Worker处理的数据量，从而导致少数Worker的运行时长远远超过其他的平均运行时长，从而导致整个任务运行时间超长，造成任务延迟。

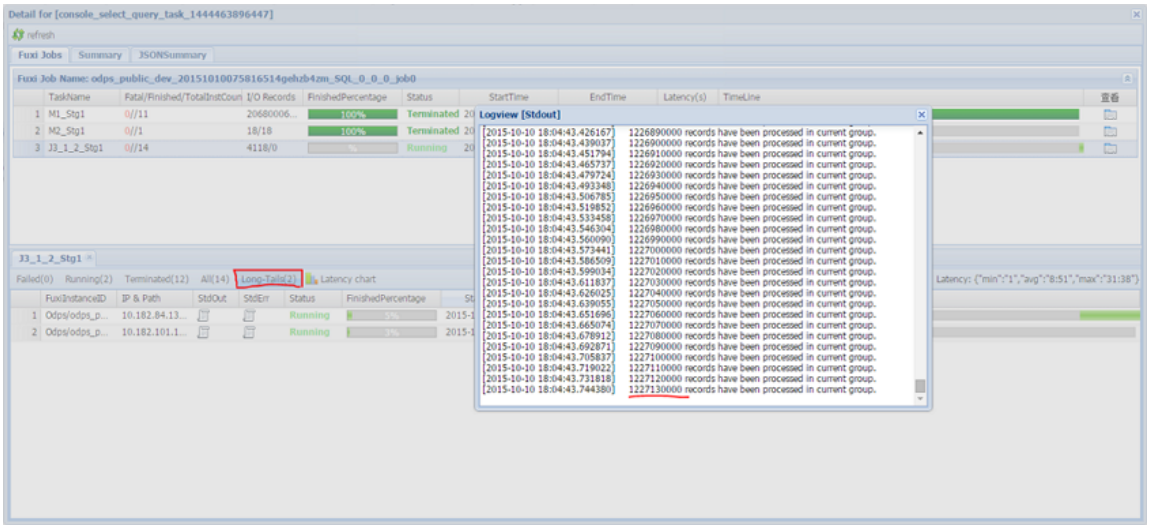
  更多数据倾斜优化的详情请参见计算长尾调优。

  — Join造成的数据倾斜

造成Join数据倾斜的原因是Join on的key分布不均匀。假设还是上述示例语句，现在将大表A和小表B进行Join操作，运行如下语句。

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

此时，复制logview的链接并打开webcosole页面，双击执行Join操作的fuxi job，可以看到此时在[Long-tails]区域有长尾，表示数据已经倾斜了。



此时您可通过如下方法进行优化。

- 由于表B是个小表并且没有超过512MB，您可将上述语句优化为mapjoin语句再执行，语句如下。

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

- 您也可将倾斜的key用单独的逻辑来处理，例如经常发生两边的key中有大量null数据导致了倾斜。则需要在Join前先过滤掉null的数据或者补上随机数，然后再进行Join，示例如下。

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

在实际场景中，如果您知道数据倾斜了，但无法获取导致数据倾斜的key信息，那么可以使用一个通用的方案，查看数据倾斜，如下所示。

```
例如：select * from a join b on a.key=b.key; 产生数据倾斜。
您可以执行：
```sql
select left.key, left.cnt * right.cnt from
(select key, count(*) as cnt from a group by key) left
```
```

```
join
(select key, count(*) as cnt from b group by key) right
on left.key=right.key;
```

查看key的分布，可以判断a join b时是否会有数据倾斜。

- **group by倾斜**

  造成group by倾斜的原因是group by的key分布不均匀。

  假设表A内有两个字段（key，value），表内的数据量足够大，并且key的值分布不均，运行语句如下所示：

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

  当表中的数据足够大时，您会在webcosole页面看见长尾。若想解决这个问题，您需要在执行SQL前设置防倾斜的参数，设置语句为`set odps.sql.groupby.skewindata=true`。

- **错误使用动态分区造成的数据倾斜**

  动态分区的SQL，在MaxCompute中会默认增加一个Reduce，用来将相同分区的数据合并在一起。这样做的好处，如下所示。

  - 可减少MaxCompute系统产生的小文件，使后续处理更快速。
  - 可避免一个Worker输出文件很多时占用内存过大。

  但也正是因为这个Reduce的引入，导致分区数据如果有倾斜的话，会发生长尾。因为相同的数据最多只会有10个Worker处理，所以数据量大，则会发生长尾，示例如下。

  ```
  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';
  ```

  这种情况下，没有必要使用动态分区，所以可以改为如下语句：

  ```
  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';
  ```

- **窗口函数的优化**

如果您的SQL语句中用到了窗口函数，一般情况下每个窗口函数会形成一个Reduce作业。如果窗口函数略多，那么就会消耗资源。在某些特定场景下，窗口函数是可以进行优化的。

- 窗口函数over后面要完全相同，相同的分组和排序条件。
- 多个窗口函数在同一层SQL执行。

符合上述两个条件的窗口函数会合并为一个Reduce执行。SQL示例如下所示。

```
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;
```

- **子查询改Join**

例如有一个子查询，如下所示。

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

当此语句中的table_b子查询返回的col1的个数超过1000个时，系统会报错为`records returned from subquery exceeded limit of 1000`。此时您可以使用Join语句来代替，如下所示。

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

 说明：

- 如果没用Distinct，而子查询c返回的结果中有相同的col1的值，可能会导致a表的结果数变多。
- 因为Distinct子句会导致查询全落到一个Worker里，如果子查询数据量比较大的话，可能会导致查询比较慢。
- 如果已经从业务上控制了子查询里的col1不可能会重复，比如查的是主键字段，为了提高性能，可以把Distinct去掉。

4 编写MapReduce

本文将为您介绍安装好MaxCompute客户端后，如何快速运行MapReduce WordCount示例程序。

说明：

如果您使用Maven，可以从*Maven 库*中搜索odps-sdk-mapred获取不同版本的Java SDK。相关配置信息如下所示：

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

前提条件

- 编译、运行MapReduce时，需要首先安装JDK1.6或以上版本。
- 请参见安装并配置客户端对MaxCompute客户端进行部署。更多关于MaxCompute客户端的使用，请参见*MaxCompute*客户端。

操作步骤

1. 安装并配置好客户端后，打开odpscmd.bat，进入相应项目空间中。
2. 输入建表语句，创建输入和输出表。如下所示：

   ```
   CREATE TABLE wc_in (key STRING, value STRING);
   CREATE TABLE wc_out (key STRING, cnt BIGINT);
   -- 创建输入、输出表
   ```

   更多创建表的语句请参见创建表。
3. 上传数据。

   您可以通过以下两种方式上传数据。

   - 使用Tunnel命令上传数据。

     ```
     tunnel upload kv.txt wc_in
     -- 上传示例数据
     ```

     kv.txt文件中的数据如下所示：

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

```
186,val_86
```

- 您也可以通过SQL语句直接插入数据，示例如下：

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

4. 编写MapReduce程序并编译。

   MaxCompute为您提供了便捷的Eclipse开发插件，方便您快速开发MapReduce程序，并提供了本地调试MapReduce的功能。

   您需要先在Eclipse中创建一个项目工程，而后在此工程中编写MapReduce程序。本地调试通过后，将编译好的程序（Jar 包，如*Word-count-1.0.jar*）导出并上传至MaxCompute。详情请参见*MapReduce*开发插件介绍。

5. 添加Jar包到project资源（比如这里的Jar包名为word-count-1.0.jar）。

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

6. 在MaxCompute客户端运行Jar命令。

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

7. 在MaxCompute客户端查看结果。

```
select * from wc_out;
```

说明：

如果您在 Java 程序中使用了任何资源，请务必将此资源加入**-resources**参数。Jar命令的详细介绍请参见作业提交。

5 JAVA UDF开发

MaxCompute的UDF包括UDF、UDAF和UDTF三种函数。通常情况下，这三种函数被统称为UDF。

 说明：

当前MaxCompute已支持*UDJ*、*UDT*，详细信息可参见*JAVA UDF*。

实现JAVA UDF使用Maven的用户可以从*Maven*库中搜索**odps-sdk-udf**获取不同版本的Java SDK，相关配置信息如下所示：

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

通常情况下，JAVA UDF的开发可以通过以下几种方式：

- 使用*MaxCompute Studio*完成*JAVA UDF*开发整个流程。
- 使用*Eclipse*插件开发和调试*JAVA UDF*，导出Jar包，然后通过命令或者DataWorks添加资源后再注册函数。

本文中会分别给出UDF、UDAF、UDTF的代码示例，并通过两种方式给出开发UDF完整流程步骤示例（UDAF、UDTF操作步骤与UDF操作步骤一样）。

 说明：

- 关于自定义函数注册和注销、查看函数列表的相关命令语句请参见函数操作。
- Java和MaxCompute的数据类型对应关系，请参见参数与返回值类型。

UDF示例

下面将为您介绍一个字符小写转换功能的UDF实现示例。

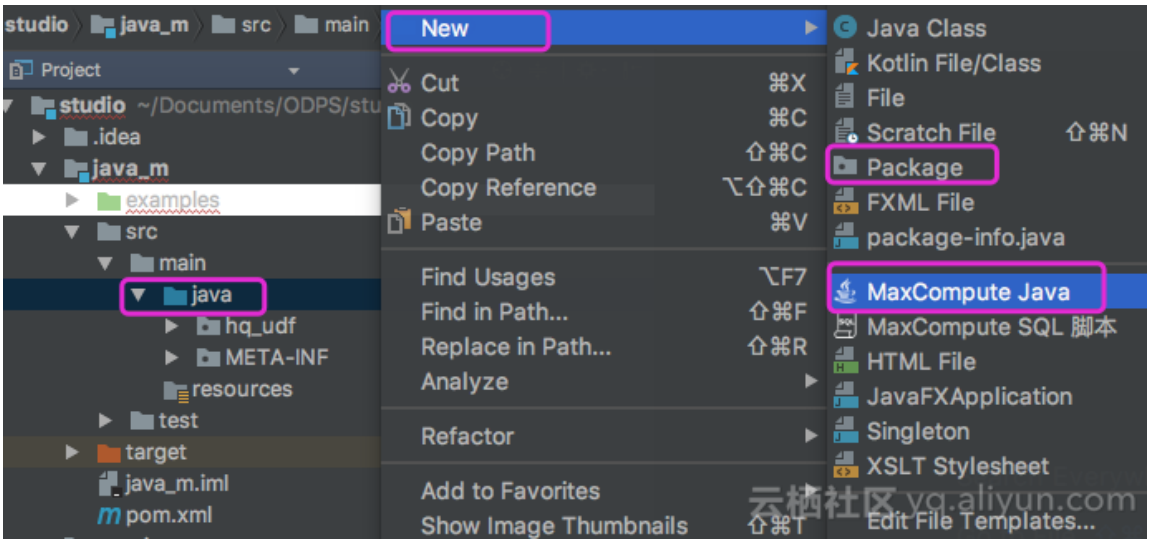
- 使用**MaxCompute Studio**开发

  1. **准备工具环境并创建Java Module。**

     这里假设已经完成环境准备，包括安装*Studio*并在Studio上创建*MaxCompute*项目链接以及创建*MaxCompute Java Module*。

  2. **编写代码。**

     在配置好的Java Module下创建Java文件。



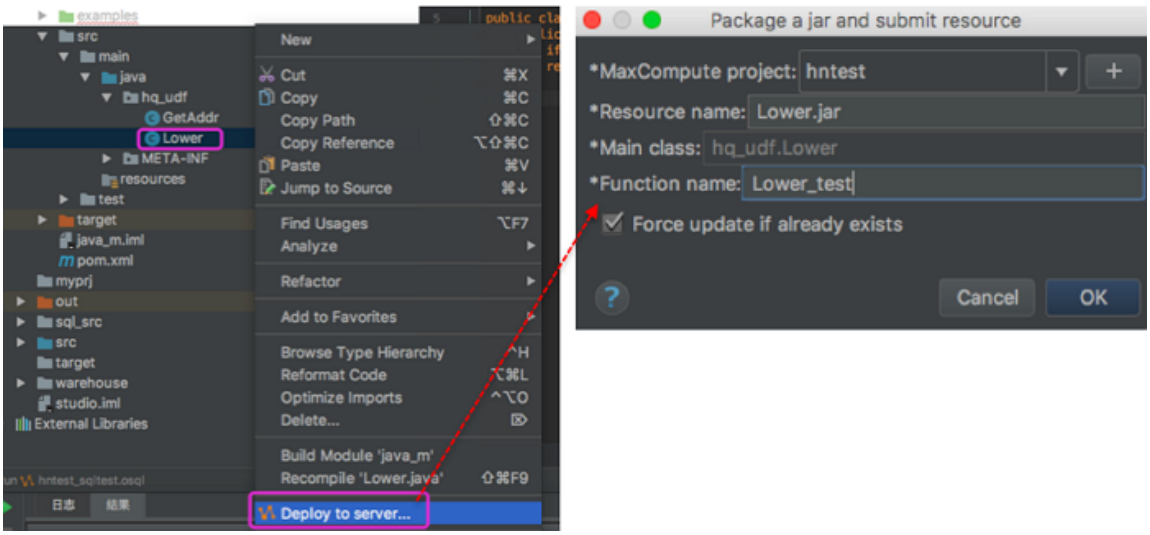
直接选择MaxCompute Java，然后在name一栏里输入`package名称.文件名`，Kind选择UDF。 之后编辑如下代码：

```
package <package名称>;
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();
}
}
```

> 说明：
>
> 若需本地调试java udf，请参见开发和调试*UDF*

3. 注册**MaxCompute UDF**。

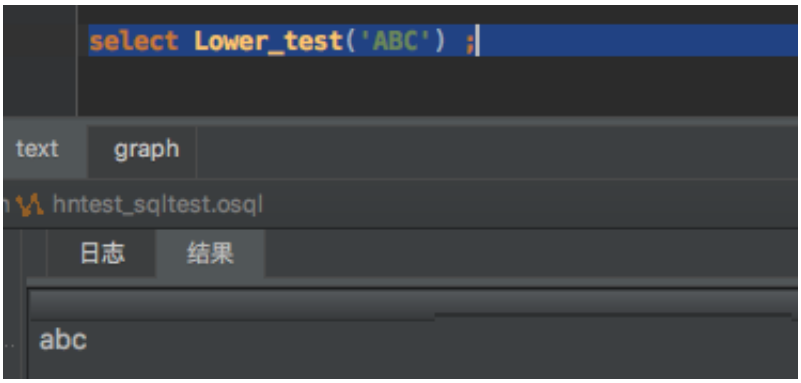
   如下图所示，右键单击UDF的Java文件，选择**Deploy to server**，弹框里选择注册到那个MaxCompute project，输入`function name`，Resource name也可以修改。



填写好后，单击**OK**即可。注册成功后会有提示。

4. 试用**UDF**。

   打开SQL脚本，执行代码如`select Lower_test('ABC');`结果如下图所示：



> 说明：
> Studio中编写SQL脚本请参见编写*SQL*脚本。

- 使用**Eclipse**插件开发

  1. 创建工程

     此处假设已经在Eclipse插件创建好一个MaxCompute（原名ODPS）工程，详情请参见创建*MaxCompute*工程。

  2. 编写代码

     按照MaxCompute UDF框架的规定，实现函数功能，并进行编译。示例如下：

```
package <package名称>;
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();
}
}
```

将这个Jar包命名为**my_lower.jar**。

说明：

- 更详细的开发调试代码的介绍请参见*UDF*开发插件介绍。
- SDK的使用信息请参见*UDF SDK* 。

3. **添加资源**

在运行UDF之前，必须指定引用的UDF代码。代码通过资源的形式添加到MaxCompute中。Java UDF必须被打成Jar包，以Jar资源添加到MaxCompute中，UDF框架会自动加载Jar包，运行用户自定义的UDF。

说明：

MaxCompute MapReduce也用到了资源这一特有概念，*MapReduce*文档中对资源的使用也有阐述。

执行如下命令：

```
add jar my_lower.jar;
-- 如果存在同名的资源请将这个jar包重命名
-- 并注意修改下面示例命令中相关jar包的名字
-- 又或者直接使用-f选项覆盖原有的jar资源
```

4. **注册UDF函数**

命令格式如下：

```
CREATE FUNCTION AS <package_to_class> USING <resource_list>;
```

参数说明：

- **function_name**：UDF函数名，这个名字就是SQL中引用该函数所使用的名字。
- **package_to_class**：如果是Java UDF，这个名字就是从顶层包名一直到实现UDF类名的fully qualified class name。如果是python UDF，这个名字就是python脚本名.类名。并且这个名字必须使用引号。
- **resource_list**：UDF所用到的资源列表。

- 此资源列表必须包括UDF代码所在的资源。
- 如果您的代码中通过distributed cache接口读取资源文件，此列表中还要包括UDF所读取的资源文件列表。
- 资源列表由多个资源名组成，资源名之间由逗号分隔，且资源列表必须用引号引起来。
- 如果需要指定资源所在的project，写法为`<project_name>/resources/<resource_name>`。

Jar包被上传后，使得MaxCompute有条件自动获取代码并运行。但此时仍然无法使用这个UDF，因为MaxCompute中并没有关于这个UDF的任何信息。因此需要在MaxCompute中注册一个唯一的函数名，并指定这个函数名与哪个jar资源的哪个类对应。

执行如下命令：

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

说明：

- 与资源文件一样，同名函数只能注册一次。
- 一般情况下，您的自建函数无法覆盖系统内建函数。只有项目空间的Owner才有权利覆盖内建函数。如果您使用了覆盖内建函数的自定义函数，在SQL执行结束后，会在Summary中打印出warning信息。

5. 在SQL中使用此函数进行验证：

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

UDAF示例

UDAF的注册方式与UDF基本相同，使用方式与内建函数中的聚合函数相同。计算平均值的UDAF的代码示例，如下所示：

```
package org.alidata.odps.udf.examples;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.Text;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.udf.Aggregator;
import com.aliyun.odps.udf.UDFException;
/**
* project: example_project
* table: wc_in2
* partitions: p2=1,p1=2
* columns: colc,colb,cola
*/
public class UDAFExample extends Aggregator {
```

```
@Override
public void iterate(Writable arg0, Writable[] arg1) throws UDFExcepti
on {
LongWritable result = (LongWritable) arg0;
for (Writable item : arg1) {
Text txt = (Text) item;
result.set(result.get() + txt.getLength());
}
}
@Override
public void merge(Writable arg0, Writable arg1) throws UDFException {
LongWritable result = (LongWritable) arg0;
LongWritable partial = (LongWritable) arg1;
result.set(result.get() + partial.get());
}
@Override
public Writable newBuffer() {
return new LongWritable(0L);
}
@Override
public Writable terminate(Writable arg0) throws UDFException {
return arg0;
}
}
```

UDTF示例

UDTF的注册和使用方式与UDF相同。代码示例如下：

```
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提供了很多内建函数来满足您的计算需求，同时您还可以通过创建自定义函数来满足不同的计算需求。详情请参见创建自定义函数。

6 编写Graph

本文将以*SSSP*算法为例，为您介绍如何提交Graph作业。

*Graph*作业的提交方式与*MapReduce*基本相同。如果您使用Maven，可以从*Maven*库中搜索`odps-sdk-graph`获取不同版本的Java SDK，相关配置信息如下所示：

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

操作步骤

1. 进入console并运行odpscmd。
2. 创建输入表和输出表 。

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

   创建表的更多语句请参见表操作。
3. 上传数据。

   本地数据的内容如下：

   ```
   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
   ```

   以空格键做两列的分隔符，执行Tunnel命令上传数据。

   ```
   tunnel u -fd " " sssp.txt sssp_in;
   ```
4. 编写SSSP示例。

   根据*Graph*开发插件的介绍，本地编译、调试*SSSP*算法示例。本示例中假设代码被打包为*odps-graph-example-sssp.jar*。

   > 说明：
   > 仅需要将SSSP代码打包即可，不需要同时将SDK打入*odps-graph-example-sssp.jar*中。

5. 添加Jar资源。

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

说明：

创建资源的介绍请参见资源操作。

6. 运行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命令用于运行MaxCompute Graph作业，用法与*MapReduce*作业的运行命令完全一致。

Graph作业执行时命令行会打印作业实例ID，执行进度，结果Summary等。

输出示例如下所示：

```
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
```

说明：

如果您需要使用Graph功能，直接开通提交图计算作业即可。