Alibaba Cloud MaxCompute

快速入門

檔案版本：20180929

目錄

1 建立/查看/刪除表.....1
2 匯入資料..... 5
3 運行SQL..... 14
4 編寫MapReduce..... 20
5 JAVA UDF開發..... 22
6 編寫Graph.....28

1 建立/查看/刪除表

當您被添加到項目空間並被賦予建表等許可權後，即可操作MaxCompute。由於在MaxCompute中的操作對象（輸入、輸出）都是表，所以在處理資料之前，首先要建立表、分區。

建立/刪除表的方式有以下幾種：

- 通過MaxCompute Studio實現，詳情請參見#####/##/###。
- 通過DataWorks實現，詳情請參見###和###。
- 通過用戶端常用命令實現。

本文將為您介紹如何通過用戶端常用命令進行建立表、查看錶和刪除表的操作，用戶端的安裝請參見準備工作中的########。

建立表

建表語句如下所示：

```
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
```

建表語句說明：

- 表名與列名均對大小寫不敏感。
- 在建立表時，如果不指定if not exists選項而存在同名表，則返回出錯；若指定此選項，則無論是否存在同名表，即使原表結構與要建立的目標表結構不一致，均返回成功。已存在的同名表的元資訊不會被改動。
- ####：包括Bigint、Double、Boolean、Datetime、Decimal和String等多種資料類型。
- 表名，列名中不能有特殊字元，只能用英文的a-z，A-Z及數字和底線 _，且以字母開頭，名稱的長度不超過128位元組。
- Partitioned by：指定表的分區欄位，目前僅支援String類型，其他類型行為未定義。分區值不可以有雙位元組字元（如中文），必須是以英文字母a-z，A-Z開始後可跟字母數字，名稱的長度不超過128位元組。允許的字元包括空格、冒號（:）、底線（_）、美元符（$）、井號（#）、點（.）、驚嘆號（!）和（@），出現其他字元行為未定義。例如（\t），（\n），（/）等。當利用分區欄位對錶進行分區時，新增分區、更新分區內資料和讀取分區資料均不需要做全表掃描，可以提高處理效率。
- 注釋內容是長度不超過1024位元組的有效字串。

- lifecycle指明此表的生命週期，單位：天。`create table like`語句不會複製源表的生命週期屬性。
- 目前，在表中建的分區層次不能超過6級。一個表允許的分區個數支援按照具體的project配置，預設60,000個。

> 说明：
>
> - 建立表的詳細介紹請參見###。
> - 添加分區請參見##/###。
> - 生命週期的修改請參見##########。

建立表示例如下：

```
create table test1 (key string); -- 建立非分區表，表名 test1，欄位名 key，資料類型 string。
create table test2 (key bigint) partitioned by (pt string, ds string);
 --建立分區表
create table test3 (key boolean) partitioned by (pt string, ds string
) lifecycle 100; -- 建立帶有生命週期的表
create table test4 like test3; -- 除生命週期屬性外，test3 的其他屬性（欄位類型，分區類型等）均與 test4 完全一致
create table test5 as select * from test2; -- 這個操作會建立 test5，但分區，生命週期資訊不會被拷貝到目標表中。
-- 此操作僅會將 test2 的資料複製到 test5 中（如果 test2 有資料的話，此樣本中 test2 為空白表，後續章節會介紹資料匯入）。
```

建立表的情境如下：

假設需要建立一張使用者表user，包括如下資訊：

- user_id bigint類型：使用者標識，唯一標識一個使用者。
- gender bigint類型：性別（0，未知；1，男；2，女）。
- age bigint：使用者年齡。

按照Region（地區）和dt（日期）進行分區，生命週期為365天。

建表語句如下所示：

```
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;
```

建立分區

當建立一張分區表之後，為了往該表裡面匯入不同分區資料，您需要建立分區。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, ...)
```

如上述樣本，給使用者表user添加地區為hangzhou，日期為20150923的分區，句子顯示如下：

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

查看錶資訊

當建立表成功之後，您可以通過`desc <table_name>;`命令查看錶的資訊。

您可執行命令`desc test3;`查看上述樣本中表test3的資訊。

結果顯示如下：

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

您可執行命令`desc test4;`查看上述樣本中表test4的資訊。

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

您會發現，除生命週期屬性外，test3的其他屬性（欄位類型、分區類型等）均與test4完全一致。查看錶資訊的更多詳情請參見###。

您如果查看test5的表資訊，pt、ds兩個欄位僅會作為普通列存在，而不是表的分區 。

刪除分區

刪除分區的命令如下所示：

```
alter table table_name drop [if exists] partition_spec; partition_spec
:
: (partition_col1 = partition_col_value1, partition_col2 = partiton_c
ol_value2, ...)
```

比如刪除地區為hangzhou，日期為20150923的分區，語句如下所示：

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

刪除表

刪除表的命令如下所示：

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

刪除test2表的樣本如下：

```
drop table test2;
```

更多詳情請參見###刪除表。

2 匯入資料

MaxCompute提供多種資料匯入匯出方式，如下所示：

- 直接在用戶端使用*Tunnel##*。
- 通過MaxCompute Studio工具可視化方式實現本機資料檔案匯入匯出，詳情請參見######。
- 通過*Tunnel*提供的SDK自行編寫Java工具。
- 通過Flume及Fluentd外掛程式方式匯入。
- 通過DataWorks對資料匯入和匯出，詳情請參見*Data Integration##*。

匯出資料請參見*Tunnel####*中Download的相關命令。

Tunnel命令匯入資料

1. 準備資料

   假設您已準備本地檔案wc_example.txt，本地存放路徑為D:\odps\odps\bin，內容如下：

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

2. 建立MaxCompute表

   您需要把上面的資料匯入到MaxCompute的一張表中，所以需要建立MaxCompute表。

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

3. 執行tunnel命令

   輸入表建立成功後，可以在MaxCompute用戶端輸入Tunnel命令進行資料的匯入，如下所示：

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

4. 執行成功後，查看錶wc_in的記錄。

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

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

說明：

- 有關Tunnel命令的更多詳細介紹，例如如何將資料匯入分區表等，請參見*Tunnel##*。
- 當表中含有多個列時，可以通過**-fd**參數指定資料行分隔符號。

MaxCompute Studio匯入資料

在使用MaxCompute Studio匯入資料前，您需要先*##MaxCompute Studio*，並########。

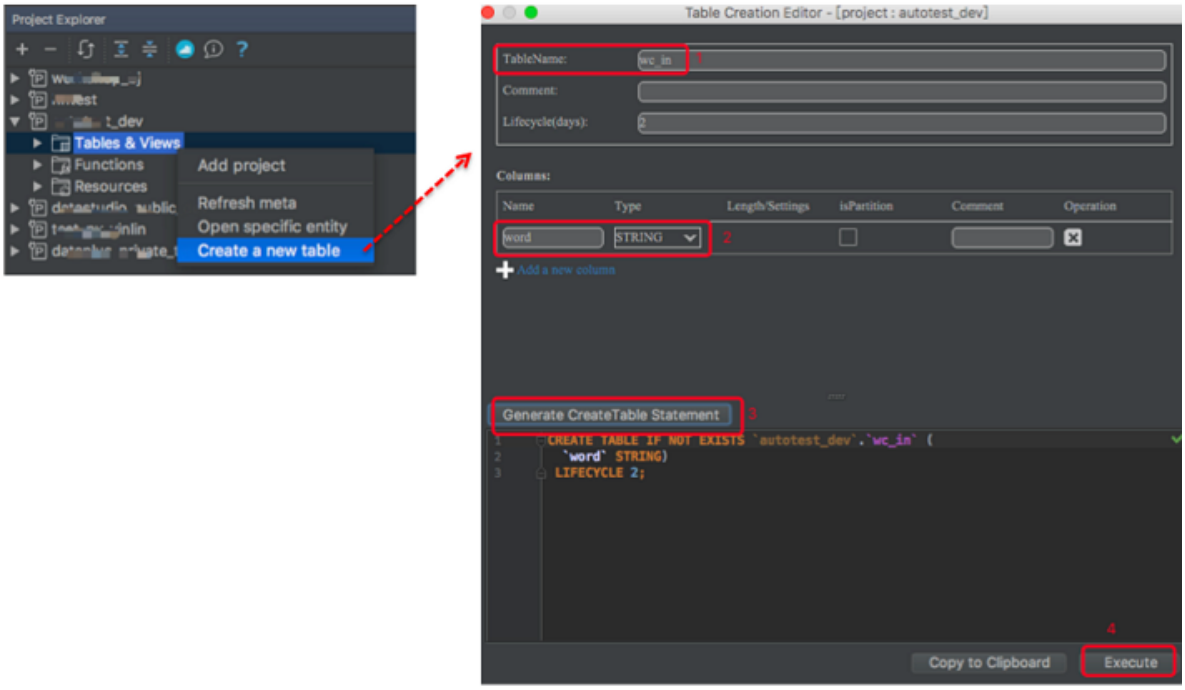
1. 準備資料。

   假設您已準備本地檔案wc_example.txt，本地存放路徑為D:\odps\odps\bin，內容如下：

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

2. 建立MaxCompute表。

   您需要把上面的資料匯入到MaxCompute的一張表中，所以需要建立MaxCompute表。按右鍵項目的**tables&views**列表：

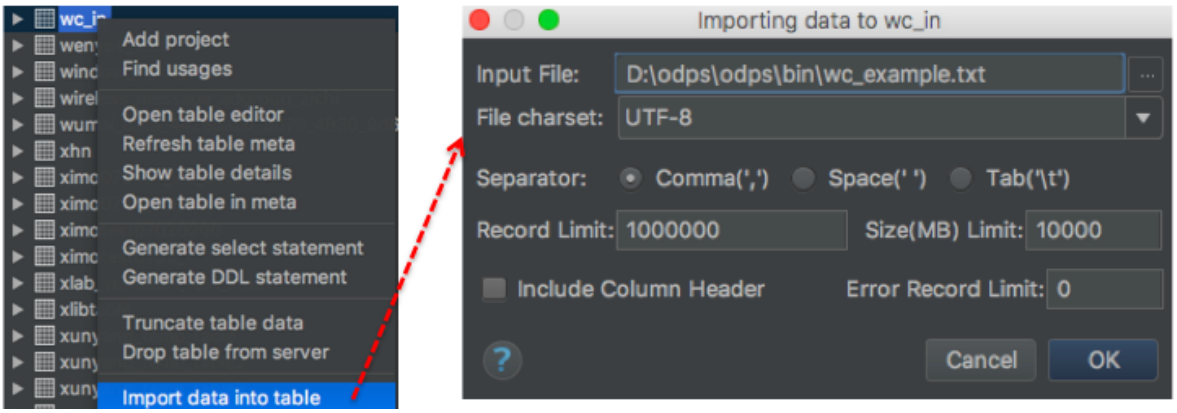


執行成功，則建表成功。

3. 上傳資料檔案。

   按右鍵tables&views列表中建立的表wc_in

   > 说明：
   > 如果建立表未顯示，請單擊重新整理。



Tunnel SDK

下文將通過情境樣本，為您介紹如何利用Tunnel SDK上傳資料。

情境描述

上傳資料到MaxCompute，其中，項目空間為odps_public_dev，表名為tunnel_sample_test，分區為pt=20150801，dt=hangzhou。

操作步驟

1. 建立表，添加分區，SQL語句如下所示：

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

2. 建立UploadSample的工程目錄結構，如下所示：

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

目錄說明：

- pom.xml：maven工程檔案。
- UploadSample：Tunnel源檔案。

3. 編寫UploadSample程式。代碼如下所示：

```
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：

此處省略了AccessKeyID和AccessKeySecret的配置，實際運行時請換上您自己的相關資訊。

4. 配置pom.xml檔案。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>
```

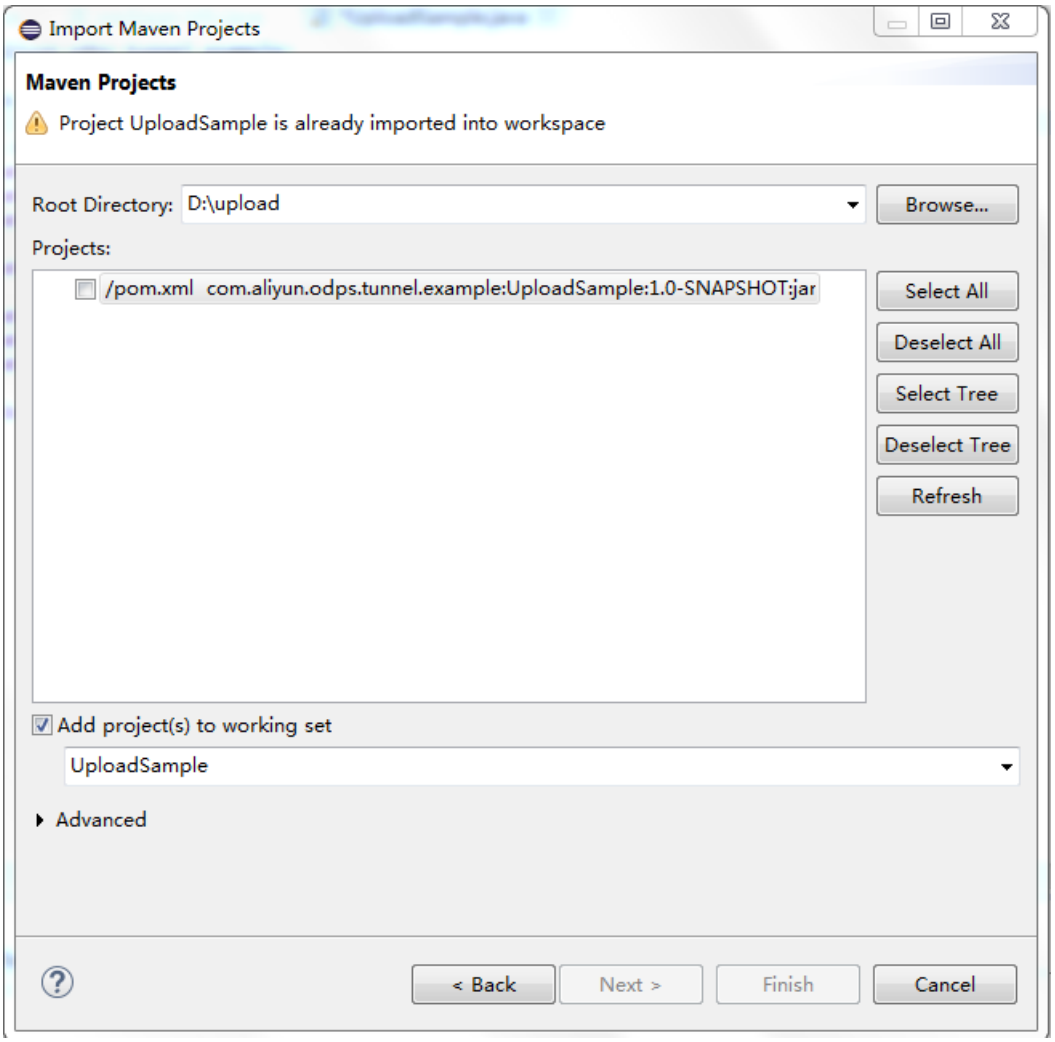
5. 編譯與運行。

   編譯UploadSample工程，如下所示：

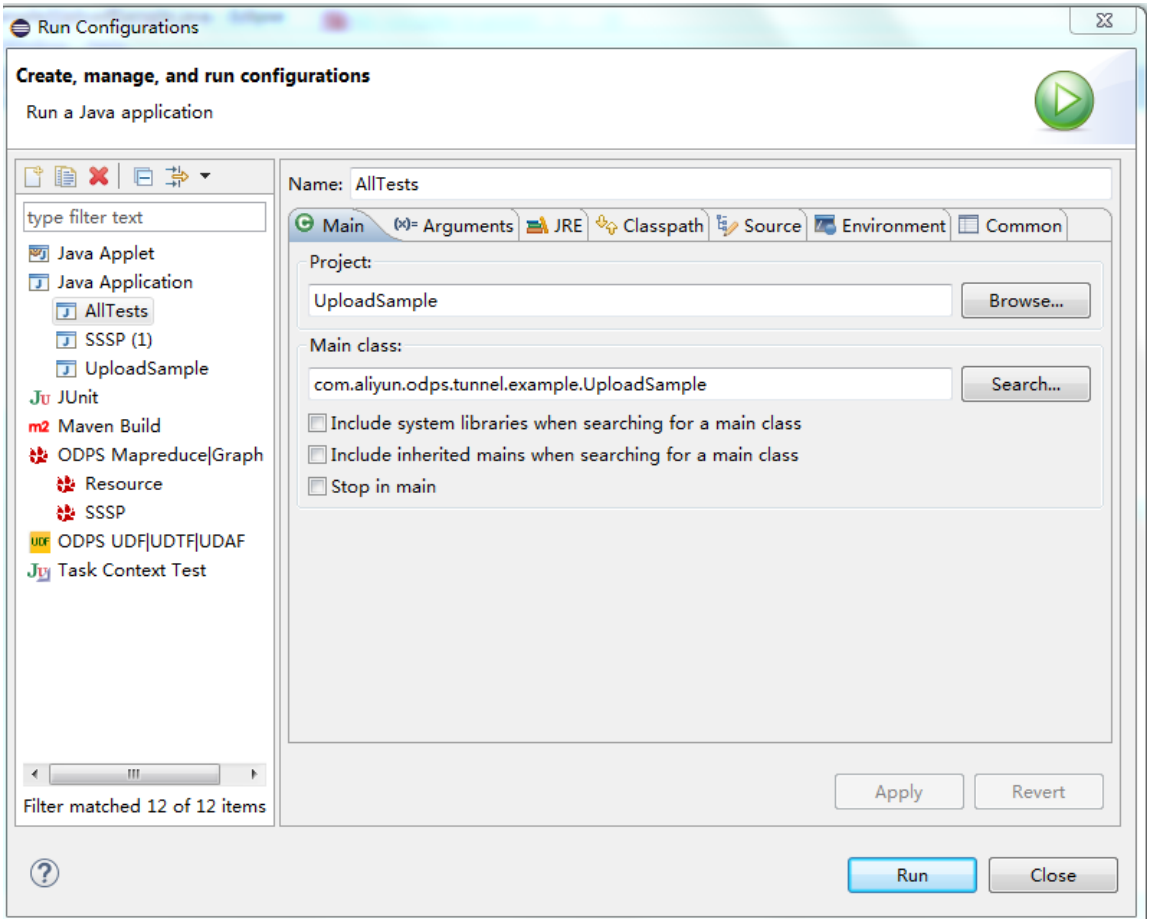
   ```
   mvn package
   ```

   運行UploadSample程式，此處使用eclipse匯入maven project。

   1. 按右鍵**java**工程並導航至**Import** > **Maven** > **Existing Maven Projects**， 設定如下：



2. 按右鍵**UploadSample.java** 並單擊**Run As** > **Run Configurations**，如下所示：



3. 單擊**Run** 運行成功，控制台顯示如下：

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

6. 查看運行結果。

您在用戶端輸入如下語句，即可查看運行結果。

```
select * from tunnel_sample_test;
```

顯示結果如下：

```
+----+------+----+----+
| 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作為MaxCompute中一個獨立的服務，有專屬的訪問連接埠提供給大家。當您在阿里雲內網環境中，使用Tunnel內網串連下載資料時，MaxCompute不會將該操作產生的流量計入計費。此外內網地址僅對上海域的雲產品有效。
- MaxCompute阿里雲內網和公網訪問地址請參見########## 。

其他匯入方式

除了通過用戶端及Tunnel Java SDK匯入資料，阿里雲數加Data Integration、開源的Sqoop、Fluentd、Flume、LogStash 等工具都可以進行資料匯入到MaxCompute，詳情請參見#########-####。

3 運行SQL

大多數使用者對SQL的文法並不陌生，簡單來說，MaxCompute SQL是用於查詢和分析MaxCompute中的大規模資料。目前SQL的主要功能如下所示。

- 支援各類運算子。
- 通過DDL語句對錶、分區以及視圖進行管理。
- 通過Select語句查詢表中的記錄，通過Where語句過濾表中的記錄。
- 通過Insert語句插入資料、更新資料。
- 通過等值串連Join操作，支援兩張表的關聯，並支援多張小表的Mapjoin。
- 支援通過內建函數和自訂函數來進行計算。
- 支援Regex。

本文將為您簡單介紹MaxCompute SQL使用中需要注意的問題，不再做操作樣本。

 说明：

- MaxCompute SQL不支援事務、索引及Update/Delete等操作，同時MaxCompute的SQL文法與Oracle，MySQL有一定差別，您無法將其他資料庫中的SQL語句無縫遷移到MaxCompute上來。
- 在使用方式上，MaxCompute作業提交後會有幾十秒到數分鐘不等的排隊調度，所以適合處理跑批作業，一次作業批量處理海量資料，不適合直接對接需要每秒處理幾千至數萬筆事務的前台業務系統。
- 關於SQL操作的詳細樣本，請參見*SQL##*。

DDL語句

簡單的DDL操作包括建立表、添加分區、查看錶和分區資訊、修改表、刪除表和分區，更多詳情請參見##/##/###。

Select語句

- group by語句的key可以是輸入表的列名，也可以是由輸入表的列構成的運算式，不可以是Select語句的輸出資料行。

```
select substr(col2, 2) from tbl group by substr(col2, 2);-- 可以，group by的key可以是輸入表的列構成的運算式；
 select col2 from tbl group by substr(col2, 2); -- 不可以，group by的key不在select語句的列中；
```

```
 select substr(col2, 2) as c from tbl group by c; -- 不可以，group by
的key 不可以是列的別名，即select語句的輸出資料行；
```

之所以有這樣的限制，是因為在通常的SQL解析中，group by的操作先於Select操作，因此group by只能接受輸入表的列或運算式為key。

- order by必須與limit連用。
- sort by前必須加distribute by。
- order by/sort by/distribute by的key必須是Select語句的輸出資料行，即列的別名。如下所示：

```
select col2 as c from tbl order by col2 limit 100 --不可以，order by的
key不是select語句的輸出資料行，即列的別名
  select col2 from tbl order by col2 limit 100; -- 可以，當select語句
的輸出資料行沒有別名時，使用列名作為別名。
```

之所以有這樣的限制，是因為在通常的SQL解析中，order by/sort by/distribute by後於Select操作，因此它們只能接受Select語句的輸出資料行為key。

Insert語句

- 向某個分區插入資料時，分區列不可以出現在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;
  -- 報錯返回，sale_date，region為分區列，不可以出現在靜態分區的insert語句
中。
```

- 動態分區插入時，動態分區列必須在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;
--失敗返回，動態分區插入時，動態分區列必須在select列表中。
```

Join操作

- MaxCompute SQL支援的Join操作類型包括{LEFT OUTER|RIGHT OUTER|FULL OUTER|INNER} JOIN。
- 目前最多支援16個並發Join操作。
- 在Mapjoin中，最多支援8張小表的Mapjoin。

Union All

Union All可以把多個Select操作返回的結果，聯合成一個資料集。它會返回所有的結果，但是不會執行去重。MaxCompute不支援直接對頂級的兩個查詢結果進行Union操作，需要寫成子查詢的形式。

> 说明：
>
> - Union All串連的兩個Select查詢語句，兩個Select的列個數、列名稱、列類型必須嚴格一致。
> - 如果原名稱不一致，可以通過別名設定成相同的名稱。

其他

- MaxCompute SQL目前最多支援128個並發Union操作。
- 最多支援128個並發insert overwrite/into操作。

MaxCompute SQL的更多限制請參見*SQL#####*。

SQL最佳化樣本

- **Join**語句中**Where**條件的位置

  當兩個表進行Join操作時，主表的Where限制可以寫在最後，但從表分區限制條件不要寫在Where條件中，建議寫在ON條件或者子查詢中。主表的分區限制條件可以寫在Where條件中（最好先用子查詢過濾）。樣本如下：

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

  第二個語句會先Join，後進行分區裁剪，資料量變大，效能下降。在實際使用過程中，應該盡量避免第二種用法。

- 資料扭曲

  產生資料扭曲的根本原因是有少數Worker處理的資料量遠遠超過其他Worker處理的資料量，從而導致少數Worker的運行時間長度遠遠超過其他的平均運行時間長度，從而導致整個任務已耗用時間超長，造成任務延遲。

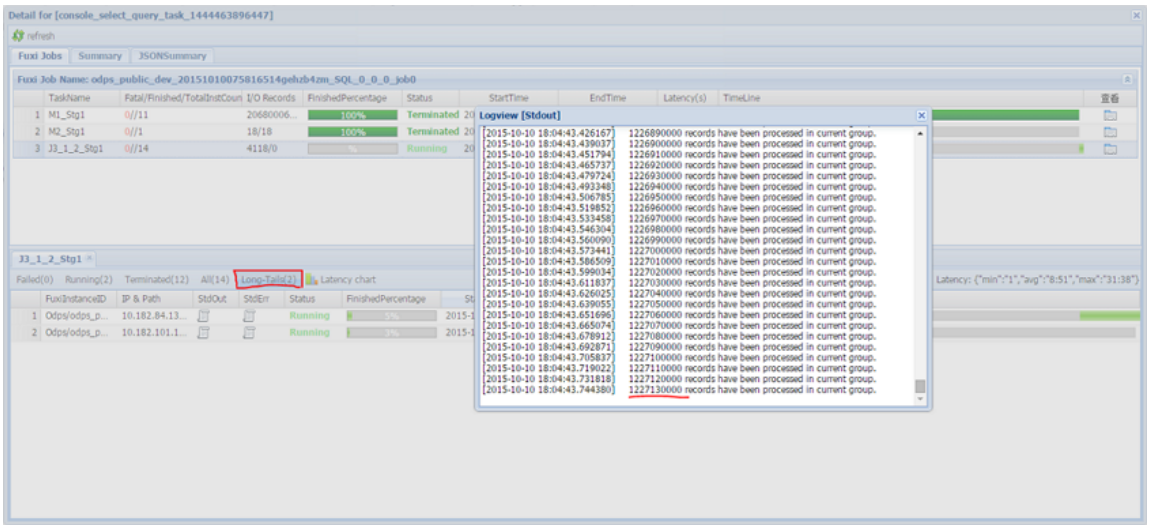
  更多資料扭曲最佳化的詳情請參見######。

  — Join造成的資料扭曲

  造成Join資料扭曲的原因是Join on的key分布不均勻。假設還是上述樣本語句，現在將大表A和小表B進行Join操作，運行如下語句。

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

  此時，複製logview的連結並開啟webcosole頁面，雙擊執行Join操作的fuxi job，可以看到此時在[Long-tails]地區有長尾，表示資料已經傾斜了。



此時您可通過如下方法進行最佳化。

- 由於表B是個小表並且沒有超過512MB，您可將上述語句最佳化為mapjoin語句再執行，語句如下。

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

- 您也可將傾斜的key用單獨的邏輯來處理，例如經常發生兩邊的key中有大量null資料導致了傾斜。則需要在Join前先過濾掉null的資料或者補上隨機數，然後再進行Join，樣本如下。

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

在實際情境中，如果您知道資料扭曲了，但無法擷取導致資料扭曲的key資訊，那麼可以使用一個通用的方案，查看資料扭曲，如下所示。

```
例如：select * from a join b on a.key=b.key; 產生資料扭曲。
您可以執行：
```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，可以判斷a join b時是否會有資料扭曲。

- **group by**傾斜

  造成group by傾斜的原因是group by的key分布不均勻。
```

假設表A內有兩個欄位（key，value），表內的資料量足夠大，並且key的值分布不均，運行語句如下所示：

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

當表中的資料足夠大時，您會在webcosole頁面看見長尾。若想解決這個問題，您需要在執行SQL前設定防傾斜的參數，設定語句為`set odps.sql.groupby.skewindata=true`。

- 錯誤使用動態分區造成的資料扭曲

  動態分區的SQL，在MaxCompute中會預設增加一個Reduce，用來將相同分區的資料合併在一起。這樣做的好處，如下所示。

  - 可減少MaxCompute系統產生的小檔案，使後續處理更快速。
  - 可避免一個Worker輸出檔案很多時佔用記憶體過大。

  但也正是因為這個Reduce的引入，導致分區資料如果有傾斜的話，會發生長尾。因為相同的資料最多隻會有10個Worker處理，所以資料量大，則會發生長尾，樣本如下。

```
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';
```

  這種情況下，沒有必要使用動態分區，所以可以改為如下語句：

```
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';
```

- 視窗函數的最佳化

  如果您的SQL語句中用到了視窗函數，一般情況下每個視窗函數會形成一個Reduce作業。如果視窗函數略多，那麼就會消耗資源。在某些特定情境下，視窗函數是可以進行最佳化的。

  - 視窗函數over後面要完全相同，相同的分組和排序條件。
  - 多個視窗函數在同一層SQL執行。

  符合上述兩個條件的視窗函數會合并為一個Reduce執行。SQL樣本如下所示。

```
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;
```

- 子查詢改**Join**

  例如有一個子查詢，如下所示。

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

  當此語句中的table_b子查詢返回的col1的個數超過1000個時，系統會報錯為`records returned from subquery exceeded limit of 1000`。此時您可以使用Join語句來代替，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：

  - 如果沒用Distinct，而子查詢c返回的結果中有相同的col1的值，可能會導致a表的結果數變多。
  - 因為Distinct子句會導致查詢全落到一個Worker裡，如果子查詢資料量比較大的話，可能會導致查詢比較慢。
  - 如果已經從業務上控制了子查詢裡的col1不可能會重複，比如查的是主鍵欄位，為了提高效能，可以把Distinct去掉。

4 編寫MapReduce

本文將為您介紹安裝好MaxCompute用戶端後，如何快速運行MapReduce WordCount樣本程式。

说明：

如果您使用Maven，可以從*Maven #*中搜尋odps-sdk-mapred擷取不同版本的Java SDK。相關配置資訊如下所示：

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

前提條件

- 編譯、運行MapReduce時，需要首先安裝JDK1.6或以上版本。
- 請參見########對MaxCompute用戶端進行部署。更多關於MaxCompute用戶端的使用，請參見*MaxCompute###*。

操作步驟

1. 安裝並配置好用戶端後，開啟odpscmd.bat，進入相應項目空間中。
2. 輸入建表語句，建立輸入和輸出表。如下所示：

   ```
   CREATE TABLE wc_in (key STRING, value STRING);
   CREATE TABLE wc_out (key STRING, cnt BIGINT);
   -- 建立輸入、輸出表
   ```

   更多建立表的語句請參見###。
3. 上傳資料。

   您可以通過以下兩種方式上傳資料。

   - 使用Tunnel命令上傳資料。

     ```
     tunnel upload kv.txt wc_in
     -- 上傳樣本資料
     ```

     kv.txt檔案中的資料如下所示：

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

```
186,val_86
```

- 您也可以通過SQL語句直接插入資料，樣本如下：

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

4. 編寫MapReduce程式並編譯。

   MaxCompute為您提供了便捷的Eclipse開發外掛程式，方便您快速開發MapReduce程式，並提供了本地調試MapReduce的功能。

   您需要先在Eclipse中建立一個項目工程，而後在此工程中編寫MapReduce程式。本地調試通過後，將編譯好的程式（Jar 包，如*Word-count-1.0.jar*）匯出並上傳至MaxCompute。詳情請參見*MapReduce########*。

5. 添加Jar包到project資源（比如這裡的Jar包名為word-count-1.0.jar）。

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

6. 在MaxCompute用戶端運行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用戶端查看結果。

```
select * from wc_out;
```

说明：

如果您在 Java 程式中使用了任何資源，請務必將此資源加入**-resources**參數。Jar命令的詳細介紹請參見*#####*

5 JAVA UDF開發

MaxCompute的UDF包括UDF、UDAF和UDTF三種函數。通常情況下，這三種函數被統稱為UDF。

實現JAVA UDF使用Maven的使用者可以從*Maven#*中搜尋**odps-sdk-udf**擷取不同版本的Java SDK，相關配置資訊如下所示：

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

通常情況下，JAVA UDF的開發可以通過以下幾種方式：

- 使用*MaxCompute Studio##JAVA UDF######*。
- 使用*Eclipse#########JAVA UDF*，匯出Jar包，然後通過命令或者DataWorks####後再####。

本文中會分別給出UDF、UDAF、UDTF的程式碼範例，並通過兩種方式給出開發UDF完整流程步驟樣本（UDAF、UDTF操作步驟與UDF操作步驟一樣）。

说明：

- 關於自訂函數註冊和登出、查看函數列表的相關命令語句請參見####。
- Java和MaxCompute的資料類型對應關係，請參見########。

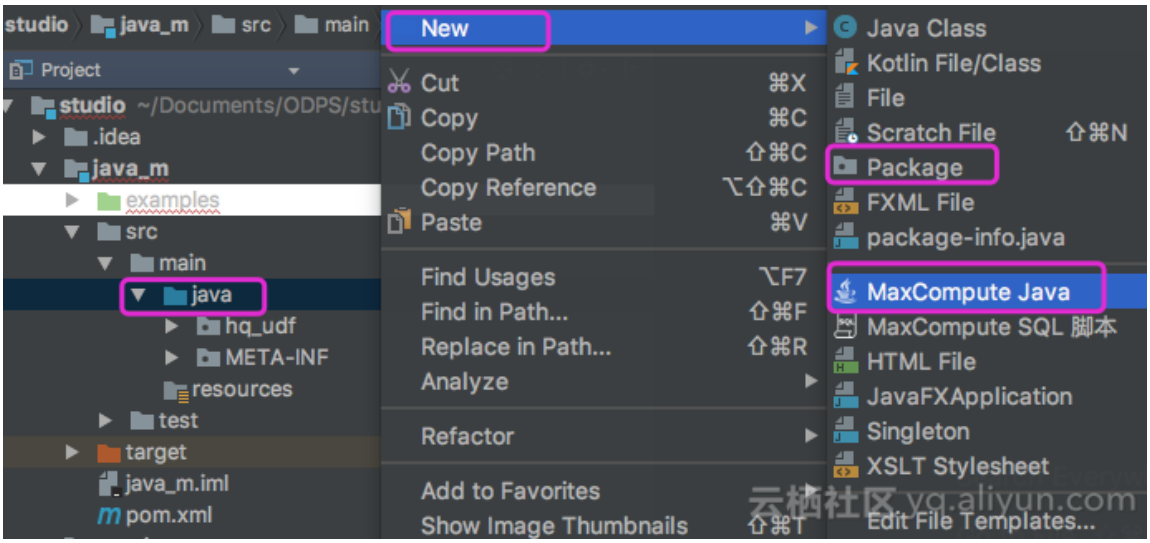
UDF樣本

下面將為您介紹一個字元小寫轉換功能的UDF實現樣本。

- 使用**MaxCompute Studio**開發
  1. 準備工具環境並建立**Java Module**。

     這裡假設已經完成環境準備，包括*##Studio*並在Studio上*##MaxCompute####*以及*##MaxCompute Java Module*。
  2. 編寫代碼。

     在配置好的Java Module下建立Java檔案。



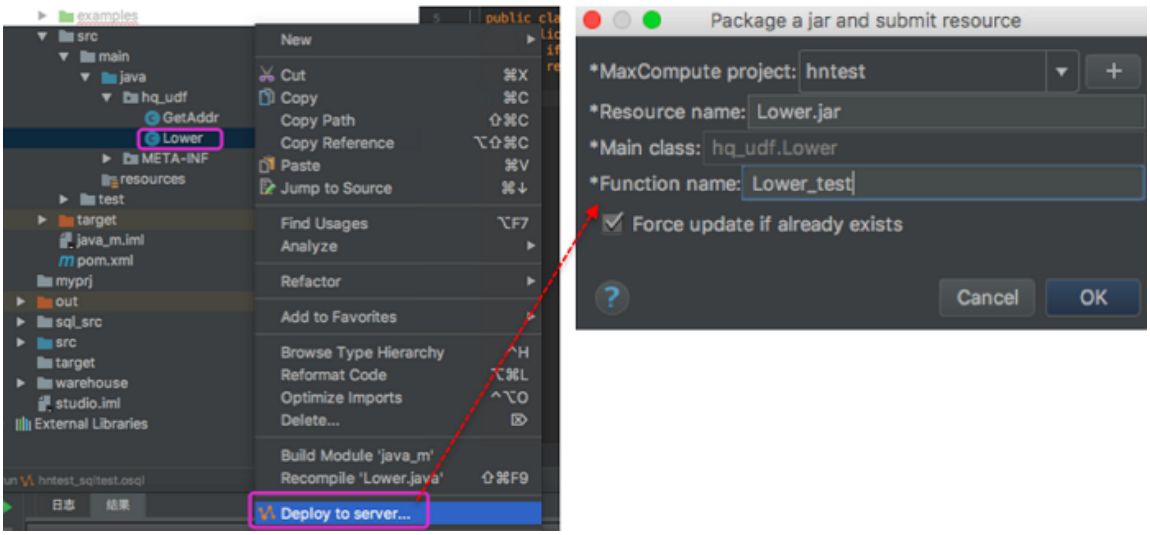
直接選擇MaxCompute Java，然後在name一欄裡輸入package名稱.檔案名稱，Kind選擇UDF。 之後編輯如下代碼：

```
package <package名稱>;
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：
> 若需本地調試java udf，請參見*#####UDF*

3. 註冊**MaxCompute UDF**。

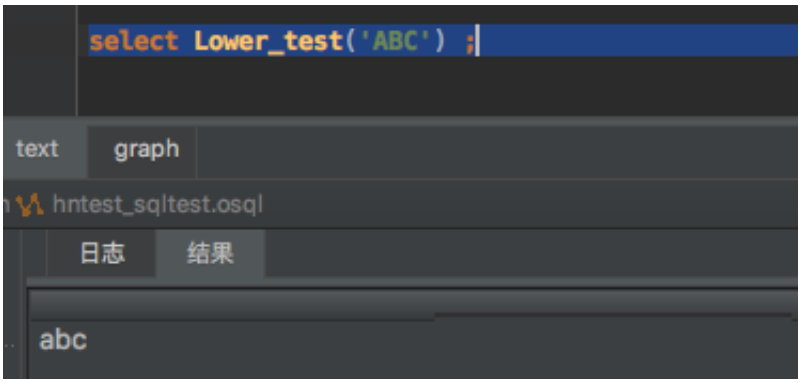
如下圖所示，按右鍵UDF的Java檔案，選擇**Deploy to server**，彈框裡選擇註冊到那個MaxCompute project，輸入`function name`，Resource name也可以修改。



填寫好後，單擊**OK**即可。註冊成功後會有提示。

4. 試用**UDF**。

   開啟SQL指令碼，執行代碼如`select Lower_test('ABC');`結果如下圖所示：



> 说明：
> Studio中編寫SQL指令碼請參見*##SQL###*。

- 使用**Eclipse**外掛程式開發

  1. 建立工程

     此處假設已經在Eclipse外掛程式建立好一個MaxCompute（原名ODPS）工程，詳情請參見*##MaxCompute##*。

  2. 編寫代碼

     按照MaxCompute UDF架構的規定，實現函數功能，並進行編譯。樣本如下：

```
package <package名稱>;
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();
}
}
```

將這個Jar包命名為**my_lower.jar**。

说明：

- 更詳細的開發調試代碼的介紹請參見*UDF########*。
- SDK的使用資訊請參見*UDF SDK* 。

3. 添加資源

在運行UDF之前，必須指定引用的UDF代碼。代碼通過資源的形式添加到MaxCompute中。Java UDF必須被打成Jar包，以Jar資源添加到MaxCompute中，UDF架構會自動載入Jar包，運行使用者自訂的UDF。

说明：

MaxCompute MapReduce也用到了資源這一特有概念，*MapReduce*文檔中對資源的使用也有闡述。

執行如下命令：

```
add jar my_lower.jar;
-- 如果存在同名的資源請將這個jar包重新命名
-- 並注意修改下面樣本命令中相關jar包的名字
-- 又或者直接使用-f選項覆蓋原有的jar資源
```

4. 註冊**UDF**函數

Jar包被上傳後，使得MaxCompute有條件自動擷取代碼並運行。但此時仍然無法使用這個UDF，因為MaxCompute中並沒有關於這個UDF的任何資訊。因此需要在MaxCompute中註冊一個唯一的函數名，並指定這個函數名與哪個jar資源的哪個類對應。

執行如下命令：

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

5. 在SQL中使用此函數進行驗證：

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

UDAF樣本

UDAF的註冊方式與UDF基本相同，使用方式與內建函數中的####相同。計算平均值的UDAF的程式碼範例，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樣本

UDTF的註冊和使用方式與UDF相同。程式碼範例如下：

```
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提供了很多內建函數來滿足您的計算需求，同時您還可以通過建立自訂函數來滿足不同的計算需求。詳情請參見######。

6 編寫Graph

本文將以*SSSP###*為例，為您介紹如何提交Graph作業。

*Graph*作業的提交方式與*MapReduce*基本相同。如果您使用Maven，可以從*Maven#*中搜尋`odps-sdk-graph`擷取不同版本的Java SDK，相關配置資訊如下所示：

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

操作步驟

1. 進入console並運行odpscmd。
2. 建立輸入表和輸出表 。

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

   建立表的更多語句請參見*###*。
3. 上傳資料。

   本機資料的內容如下：

   ```
   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
   ```

   以空格鍵做兩列的分隔字元，執行Tunnel命令上傳資料。

   ```
   tunnel u -fd " " sssp.txt sssp_in;
   ```
4. 編寫SSSP樣本。

   根據*Graph######*的介紹，本地編譯、調試*SSSP#####*。本樣本中假設代碼被打包為*odps-graph-example-sssp.jar*。

   > 说明：
   > 僅需要將SSSP代碼打包即可，不需要同時將SDK打入*odps-graph-example-sssp.jar*中。

5. 添加Jar資源。

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

> 说明：
> 建立資源的介紹請參見####。

6. 運行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命令用於運行MaxCompute Graph作業，用法與*MapReduce*作業的運行命令完全一致。

Graph作業執行時命令列會列印工作執行個體ID，執行進度，結果Summary等。

輸出樣本如下所示：

```
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，直接開通提交圖計算作業即可。