MaxCompute

快速入门

为了无法计算的价值 | 阿里云

快速入门

当用户被添加到项目空间并被赋予建表等权限后，就可以操作 MaxCompute 了。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，我们首先要创建表、分区 。

创建/删除表有两种方式：

1.通过大数据开发套件实现，详情请参见： 创建/查看/删除表；

2.通过客户端常用命令实现，详情如下：

创建表

建表语句如下：

```
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
```

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 ,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 。

注意：

- 创建表的详细介绍请参考 创建表(CREATE TABLE) 。
- 添加分区请参考 添加及删除分区。
- 生命周期的修改请参考 修改表的生命周期属性 。

创建表示例：

```
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）。
```

这里我们介绍一个创建表的场景：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：

```
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;
```

查看表信息

当创建表成功之后，我们可以通过如下命令查看表的信息：

```
desc <table_name>;
```

例如，查看上述示例中表 test3 信息：

```
desc test3;
```

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

查看 test4 信息:

```
desc test4;
```

显示结果如下：

```
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。查看表信息的更多介绍请参考 Describe Table。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 。

删除表

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

示例，删除 test2 表：

```
drop table test2;
```

更多介绍请参考 删除表(DROP TABLE)。

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，我们需要创建分区 。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_col_value2, ...)
```

比如上面的例子，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');
```

删除分区

命令如下：

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

partition_spec:

: (partition_col1 = partition_col_value1, partition_col2 = partiton_col_value2, ...)
```

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

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

MaxCompute 提供多种数据导入导出方式：直接在客户端使用 Tunnel命令 或者通过 TUNNEL 提供的 SDK 自行编写 Java 工具，通过 Flume 及 Fluentd 插件方式导入，以及通过大数据开发套件对数据导入导出，详情请参见：数据同步简介 。

> 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

准备数据

假设您已准备本地文件 wc_example.txt，内容如下：

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

此处，您把该数据文件保存在 D:\odps\odps\bin 目录下。

创建 MaxCompute 表

您需要把上面的数据导入到 MaxCompute 的一张表中，所以需要创建一张表：

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

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：

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

执行成功后，查看表 wc_in 的记录，如下：

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

ID = 20170725030626541gmbdz6jc2
Log view:
http://logview.odps.aliyun.com/logview/?h=http://service.odps.aliyun.com/api&p=a
lian&i=20170725030626541gmbdz6jc2&token=WHYwTjkvSm0zdFBtb2hNdTVnY01Qb0VQZnRRPSxP
RFBTX09GTzoxMzI5MzgzMDA0NTQwNjUxLDE1MDE1NTY3ODcseyJTdGF0ZW1lbnQiOlt7IkFjdGlvbiI6
WyJvZHBzOlJlYWQiXSwiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnByb2pl
Y3RzL2FsaWFuL2luc3RhbmNlcy8yMDE3MDcyNTAzMDYyNjU0MWdtYmR6NmpjMiJdfV0sIlZlcnNpb24i
OiIxIn0=
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.
```

注意：

- 有关 Tunnel 命令的更多详细介绍，例如：如何将数据导入分区表，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。

Tunnel SDK

关于如何利用 tunnel SDK 进行上传数据，下面也将通过场景介绍。

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”。 操作步骤如下：

1. 创建表，添加分区：

```
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
```

创建 UploadSample 的工程目录结构，如下：

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

```
|---UploadSample.java
```

UploadSample : tunnel 源文件；pom.xml : maven 工程文件。

编写 UploadSample 程序；

程序如下：

```
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();
}
}
}
```

备注：这里省略了 accessId 和 accesskey 的配置，实际运行时请换上您自己的 accessId 以及 accessKey 。

配置 pom.xml 文件；

```
<?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>
```

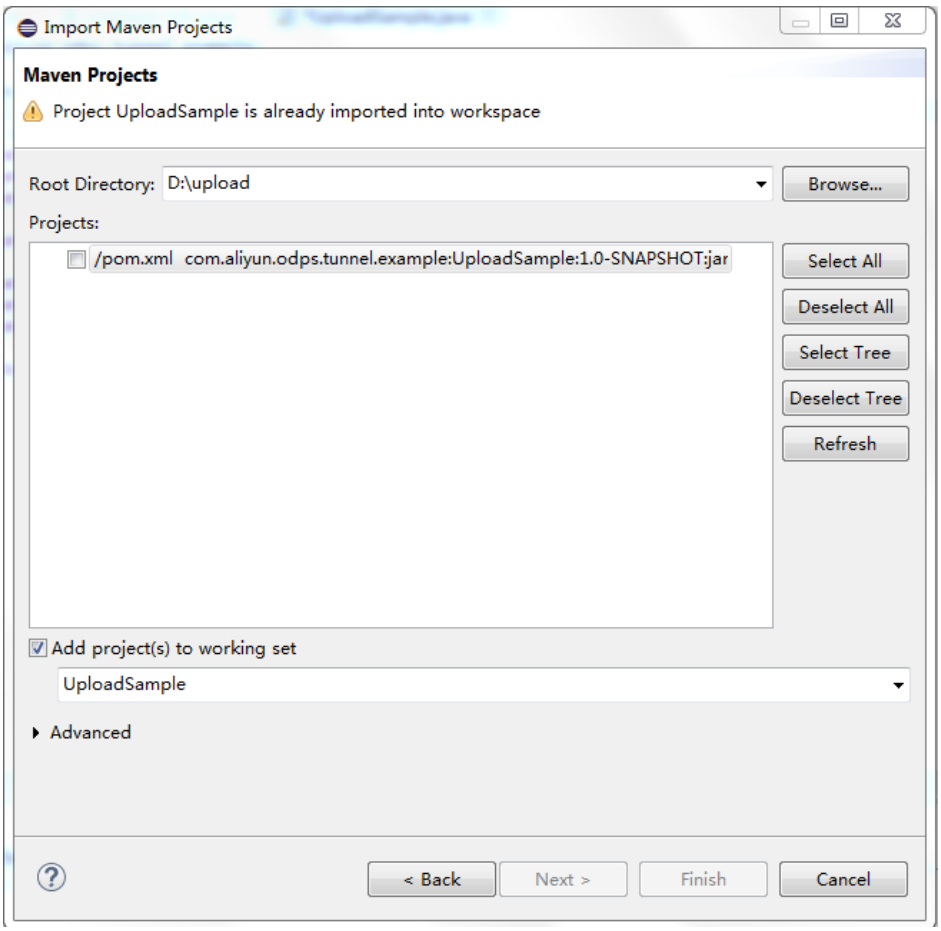
编译与运行；

编译 UploadSample 工程：

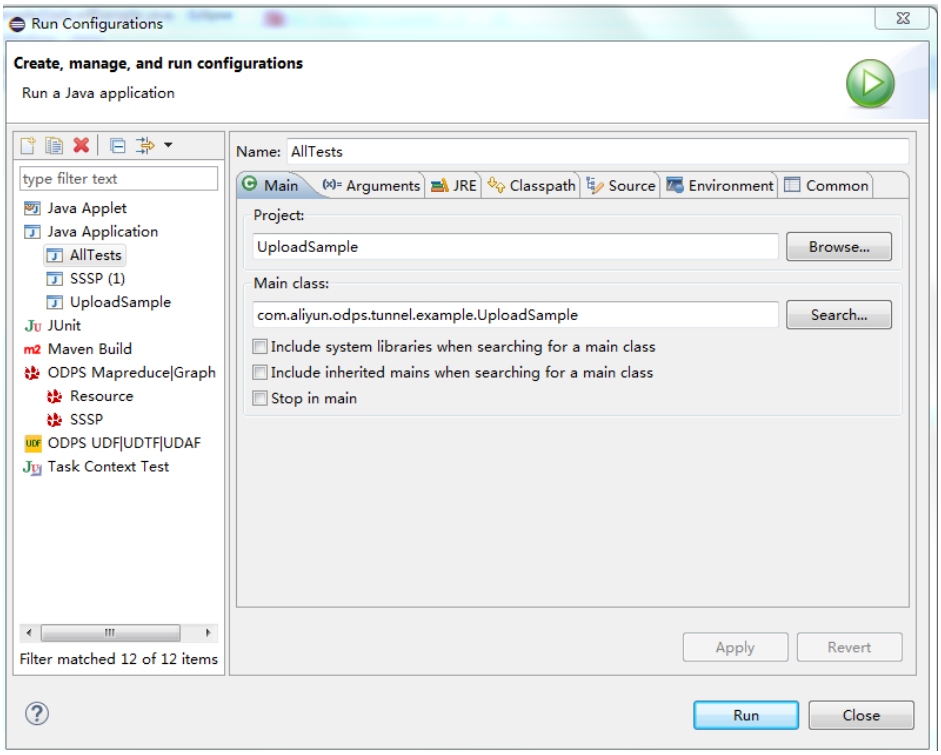
```
mvn package
```

运行 UploadSample 程序，这里使用 eclipse 导入 maven project：

- 右击 **java 工程** 并单击 **Import->Maven->Existing Maven Projects** 设置如下：



- 右击 UploadSample.java 并单击 Run As->Run Configurations，如下所示：



- 单击 **Run** 运行成功，控制台显示：

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

查看运行结果；

在客户端输入：

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

备注：

- 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 。当用户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 阿里云内网地址：http://odps-ext.aliyun-inc.com/api
- MaxCompute 公网地址：http://service.odps.aliyun.com/api

其他方式导入

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，具体介绍请参见：数据上传下载-工具介绍。

大多数用户对SQL的语法并不陌生，简单地说，MaxCompute SQL就是用于查询和分析MaxCompute中的大

规模数据。目前SQL的主要功能可以概括如下：

- 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字句过滤表中的记录。
- 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。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。

这里我们只简要介绍MaxCompute SQL使用中需要注意的问题。不再做操作示例。

注意：

- 需要注意的是，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，用户无法将其他数据库中得SQL语句无缝迁移到 MaxCompute 上来，更多与主流SQL语法差异请点击进入查看。

  此外，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的操作详细示例，请参考SQL。

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。关于这部分的介绍，请参考创建删除表。

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；
```

有这样的限制是因为，在通常的SQL解析中，group by的操作是先于select操作的，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 联用；

- 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 col2 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。
```

有这样的限制是因为，在通常的SQL解析中， order by/sort by/distribute by是后于select操作的，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6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。

另外需要注意的是，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

- ODPS 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；

SQL优化实例

Join语句中where条件的位置

当两个表进行join操作的时候，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条件里，建议写在ON条件或者子查询。主表的分区限制条件可以写在WHERE条件里(最好先用子查询过滤）。

参考下面几个sql:

```
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2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

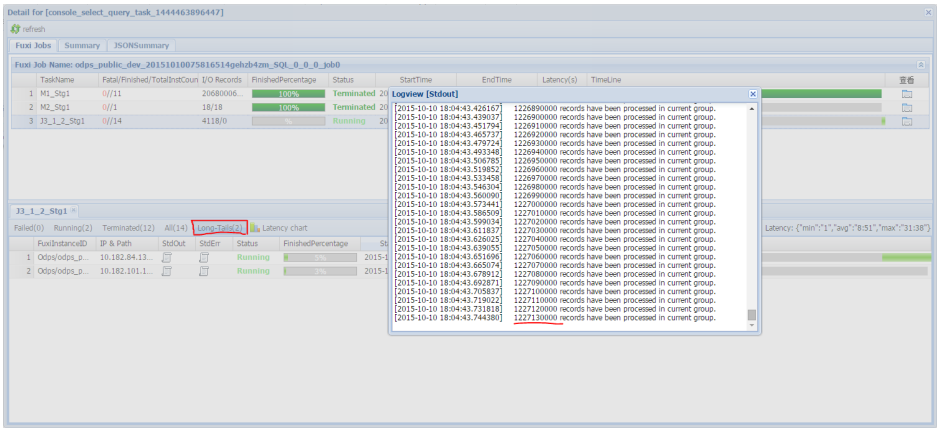
产生数据倾斜的根本原因是：有少数Worker处理的数据量远远超过其他Worker处理的数据量，从而导致少数Worker的运行时长远远超过其他的平均运行时长，进而导致整个任务运行时间超长，造成任务延迟。

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。 假设还是上面的例子,现在将大表A跟一张小表B进行join操作，运行如下语句：

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

此时我们拷贝logview的链接并打开webcosole页面，双击执行join操作的fuxi job可以看见此时在[Long-tails]区域有长尾，表示数据已经倾斜了。如下图所示：



关于如何进行优化，此时我们可以通过如下办法：

由于表B是个小表并且没有超过512MB,我们将上面的语句优化成mapjoin语句再执行，语句如下：

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

或者将倾斜的key用单独的逻辑来处理，例如经常发生两边的key里有大量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。比如：

```
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，用户往往知道数据倾斜了，但无法获取导致数据倾斜的key信息。在此有个通用的方案可以查看数据倾斜的办法：

例如：select * from a join b on a.key=b.key;产生数据倾斜。

用户可以执行：

```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，我们运行下面一条简单的语句:

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

当表中的数据足够大的时候，我们一样会在webcosole页面看见长尾。

如何解决这个问题，我们一般在执行SQL前设置防倾斜的参数： 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odps中会默认增加一个reduce，用来将相同分区的数据合并在一起。这样做的好处有：

减少 MaxCompute 系统产生的小文件，使后续处理更快；

避免一个Worker输出文件很多时占用内存过大。

但是也正是因为这个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。

例如：

```
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';
```

这种情况完全没必要使用动态分区。原来的语句可以改成：

```
 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

如果我们的SQL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，窗口函数是有可优化空间的。首选，窗口函数“over”后面要完全相同，相同的分组和排序条件；其次，多个窗口函数在同一层SQL执行。符合这两个条件的窗口函数会合并为一个Reduce执行。（如下这种SQL）：

```
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：

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

当此语句中table_b这个子查询返回的col1的个数超过1000个，系统将会报错如：records returned from subquery exceeded limit of 1000。此时可以使用Join语句来代替，如：

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

注意：

- 如果没用DISTINCT，而子查询c返回的结果里有相同的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。

MaxCompute 的 UDF 包括：UDF，UDAF，UDTF 三种函数 。通常情况下，此三种函数被统称为 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：

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

备注：

- groupId, artifactId, version信息请以在Maven库中查询到的信息为准 。
- UDF 目前只支持 Java 语言接口，用户如果想编写 UDF程序，可以通过 添加资源 的方式将 UDF 代码上传到项目空间中，使用 注册函数 语句创建 UDF；
- 本章节中会分别给出 UDF，UDAF，UDTF 的代码示例，运行 UDF 的示例请参见：UDF 开发插件介绍；
- 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

下面我们将给出一个完整的开发 UDF 流程示例，例如，实现一个字符小写转换功能的 UDF，需要经过以下几个步骤：

- 代码编写：按照 MaxCompute UDF 框架的规定，实现函数功能，并进行编译 。下面给出一个简单的代码实现：

```
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();
}
}
```

将这个 jar 包命名为“my_lower.jar” 。

- 添加资源：在运行 UDF 之前，必须指定引用的 UDF 代码 。用户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 。 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
```

- 注册 UDF 函数：用户的 jar 包被上传后，使得 MaxCompute 有条件自动获取用户代码并运行 。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 。因此需要用户在 MaxCompute 中注册一个唯一的函数名，并指定这个函数名与哪个 jar 资源的哪个函数对应 。关于如何注册 UDF，请参见： 注册函数 。

运行命令：

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

在 sql 中使用此函数：

```
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 。下面是一个计算平均值的 UDAF 的代码示例：

```
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 示例

UDTF 的注册和使用方式与 UDF 相同 。代码示例：

```
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用户的计算需求，同时用户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 。

本文章节的目的是介绍在安装好 MaxCompute 客户端后，如何快速运行 MapReduce WordCount 示例程序 。使用 Maven 的用户可以从Maven库中搜索”odps-sdk-mapred”获取不同版本的 Java SDK，相关配置信息：

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

备注：

- 编译、运行 MapReduce 需要安装 JDK1.6 版本；
- MaxCompute 客户端的快速部署请参阅 快速开始 。更多关于 MaxCompute 客户端的使用，请参考 MaxCompute 客户端参考手册 ；

1.创建输入输出表，创建表的语句请参阅 创建表(CREATE TABLE) ：

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

2.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：

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

您也可以用 sql 语句直接插入数据，比如：

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

3.编写 MapReduce 程序并编译

MaxCompute 为用户提供了便捷的 Eclipse 开发插件，方便用户快速开发 MapReduce 程序，并提供了本地调试 MapReduce 的功能 。

用户需要先在 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）导出并上传至 MaxCompute 。详细介绍请参见 MapReduce 开发插件介绍 。

4.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;
```

5.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;
```

6.在 MaxCompute 客户端查看结果：

```
select * from wc_out;
```

备注：如果在 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 Jar命令介绍 。

Graph 作业的提交方式与 MapReduce 基本相同 。下面，以 SSSP算法 为例，说明如何提交 Graph 作业 。使用 Maven 的用户可以从Maven库中搜索”odps-sdk-graph”获取不同版本的 Java SDK，相关配置信息：

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

下面将以运行示例程序单源最短距离（Single Source Shortest Path，简写：SSSP）为例， 帮助您快速掌握如何运行 Graph 作业 。

1 进入 console 并运行 odpscmd 。

2 创建输入输出表 。

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

备注：创建 Table 语句请参考 SQL Create描述。

3 上传数据

本地数据内容如下：

```
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 。本示例中假定代码被打包为 odps-graph-example-sssp.jar 。

备注：请注意，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
```

备注：创建资源介绍请参考 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 。

7 输出示例如下:

```
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
```

注意：如果用户需要使用 Graph 功能，直接开通提交图计算作业即可 。